

# High-resolution ecogeographical variables for species distribution modelling describing Latvia, 2024

Andris Avotiņš      Jekaterīna Butkeviča      Betija Rubene      Solvita Rūsiņa

Rūta Starka      Vita Šakele      Vineta Vērpēja      Ivo Vinogradovs  
Ainārs Auniņš

2025-12-03



# Contents

<b>Preface</b>	<b>19</b>
About this material . . . . .	19
Outline . . . . .	20
<b>1 Terminology and acronyms</b>	<b>21</b>
<b>2 Utilities</b>	<b>25</b>
2.1 R package egyptools . . . . .	25
2.2 Other utility functions . . . . .	26
<b>3 Templates files</b>	<b>29</b>
3.1 Vector data . . . . .	29
3.2 Raster data . . . . .	30
<b>4 Raw geodata</b>	<b>33</b>
4.1 State Forest Service’s State Forest Register . . . . .	33
4.2 Rural Support Service’s information on declared fields . . . . .	33
4.3 Melioration Cadaster . . . . .	34
4.4 Topographic Map . . . . .	59
4.5 Corine Land Cover 2018 . . . . .	64
4.6 Publicly available LVM data . . . . .	64
4.7 Soil data . . . . .	65
4.8 Dynamic World data . . . . .	69
4.9 The Global Forest Watch . . . . .	70
4.10 Palsar . . . . .	71
4.11 CHELSA v2.1 . . . . .	72
4.12 HydroClim data . . . . .	73
4.13 Sentinel-2 indices . . . . .	74
4.14 Waste and garbage disposal sites, landfills . . . . .	75
4.15 Digital elevation/terrain models . . . . .	76
4.16 Latvian Exclusive Economic Zone polygon . . . . .	77
4.17 Bogs and Mires: EDI . . . . .	77

<b>5 Geodata products</b>	<b>79</b>
5.1 Terrain products . . . . .	79
5.2 Soil texture product . . . . .	83
5.3 Landscape classification . . . . .	84
5.4 Landscape diversity . . . . .	101
<b>6 Ecogeographical variables</b>	<b>109</b>
6.1 Climate_CHELSAv2.1-bio1_cell . . . . .	110
6.2 Climate_CHELSAv2.1-bio10_cell . . . . .	111
6.3 Climate_CHELSAv2.1-bio11_cell . . . . .	111
6.4 Climate_CHELSAv2.1-bio12_cell . . . . .	112
6.5 Climate_CHELSAv2.1-bio13_cell . . . . .	113
6.6 Climate_CHELSAv2.1-bio14_cell . . . . .	114
6.7 Climate_CHELSAv2.1-bio15_cell . . . . .	115
6.8 Climate_CHELSAv2.1-bio16_cell . . . . .	116
6.9 Climate_CHELSAv2.1-bio17_cell . . . . .	117
6.10 Climate_CHELSAv2.1-bio18_cell . . . . .	117
6.11 Climate_CHELSAv2.1-bio19_cell . . . . .	118
6.12 Climate_CHELSAv2.1-bio2_cell . . . . .	119
6.13 Climate_CHELSAv2.1-bio3_cell . . . . .	120
6.14 Climate_CHELSAv2.1-bio4_cell . . . . .	121
6.15 Climate_CHELSAv2.1-bio5_cell . . . . .	122
6.16 Climate_CHELSAv2.1-bio6_cell . . . . .	123
6.17 Climate_CHELSAv2.1-bio7_cell . . . . .	124
6.18 Climate_CHELSAv2.1-bio8_cell . . . . .	124
6.19 Climate_CHELSAv2.1-bio9_cell . . . . .	125
6.20 Climate_CHELSAv2.1-clt-max_cell . . . . .	126
6.21 Climate_CHELSAv2.1-clt-mean_cell . . . . .	127
6.22 Climate_CHELSAv2.1-clt-min_cell . . . . .	128
6.23 Climate_CHELSAv2.1-clt-range_cell . . . . .	129
6.24 Climate_CHELSAv2.1-cmi-max_cell . . . . .	130
6.25 Climate_CHELSAv2.1-cmi-mean_cell . . . . .	130
6.26 Climate_CHELSAv2.1-cmi-min_cell . . . . .	131
6.27 Climate_CHELSAv2.1-cmi-range_cell . . . . .	132
6.28 Climate_CHELSAv2.1-fcf_cell . . . . .	133
6.29 Climate_CHELSAv2.1-fgd_cell . . . . .	134
6.30 Climate_CHELSAv2.1-gdd0_cell . . . . .	135
6.31 Climate_CHELSAv2.1-gdd10_cell . . . . .	136
6.32 Climate_CHELSAv2.1-gdd5_cell . . . . .	136
6.33 Climate_CHELSAv2.1-gddlgd0_cell . . . . .	137
6.34 Climate_CHELSAv2.1-gddlgd10_cell . . . . .	138

6.35 Climate_CHELSAv2.1-gddlgd5_cell . . . . .	139
6.36 Climate_CHELSAv2.1-gdgfd0_cell . . . . .	140
6.37 Climate_CHELSAv2.1-gdgfd10_cell . . . . .	141
6.38 Climate_CHELSAv2.1-gdgfd5_cell . . . . .	142
6.39 Climate_CHELSAv2.1-gsl_cell . . . . .	142
6.40 Climate_CHELSAv2.1-gsp_cell . . . . .	143
6.41 Climate_CHELSAv2.1-gst_cell . . . . .	144
6.42 Climate_CHELSAv2.1-hurs-max_cell . . . . .	145
6.43 Climate_CHELSAv2.1-hurs-mean_cell . . . . .	146
6.44 Climate_CHELSAv2.1-hurs-min_cell . . . . .	147
6.45 Climate_CHELSAv2.1-hurs-range_cell . . . . .	148
6.46 Climate_CHELSAv2.1-lgd_cell . . . . .	148
6.47 Climate_CHELSAv2.1-ngd0_cell . . . . .	149
6.48 Climate_CHELSAv2.1-ngd10_cell . . . . .	150
6.49 Climate_CHELSAv2.1-ngd5_cell . . . . .	151
6.50 Climate_CHELSAv2.1-npp_cell . . . . .	152
6.51 Climate_CHELSAv2.1-pet-penman-max_cell . . . . .	153
6.52 Climate_CHELSAv2.1-pet-penman-mean_cell . . . . .	154
6.53 Climate_CHELSAv2.1-pet-penman-min_cell . . . . .	154
6.54 Climate_CHELSAv2.1-pet-penman-range_cell . . . . .	155
6.55 Climate_CHELSAv2.1-rsds-max_cell . . . . .	156
6.56 Climate_CHELSAv2.1-rsds-mean_cell . . . . .	157
6.57 Climate_CHELSAv2.1-rsds-min_cell . . . . .	158
6.58 Climate_CHELSAv2.1-rsds-range_cell . . . . .	159
6.59 Climate_CHELSAv2.1-scd_cell . . . . .	160
6.60 Climate_CHELSAv2.1-sfcWind-max_cell . . . . .	161
6.61 Climate_CHELSAv2.1-sfcWind-mean_cell . . . . .	161
6.62 Climate_CHELSAv2.1-sfcWind-min_cell . . . . .	162
6.63 Climate_CHELSAv2.1-sfcWind-range_cell . . . . .	163
6.64 Climate_CHELSAv2.1-swb_cell . . . . .	164
6.65 Climate_CHELSAv2.1-swe_cell . . . . .	165
6.66 Climate_CHELSAv2.1-vpd-max_cell . . . . .	166
6.67 Climate_CHELSAv2.1-vpd-mean_cell . . . . .	167
6.68 Climate_CHELSAv2.1-vpd-min_cell . . . . .	167
6.69 Climate_CHELSAv2.1-vpd-range_cell . . . . .	168
6.70 HydroClim_01-max_cell . . . . .	169
6.71 HydroClim_02-max_cell . . . . .	171
6.72 HydroClim_03-max_cell . . . . .	172
6.73 HydroClim_04-max_cell . . . . .	173
6.74 HydroClim_05-max_cell . . . . .	175
6.75 HydroClim_06-min_cell . . . . .	176

6.76	HydroClim_07-max_cell	178
6.77	HydroClim_08-max_cell	179
6.78	HydroClim_09-min_cell	181
6.79	HydroClim_10-max_cell	182
6.80	HydroClim_11-min_cell	183
6.81	HydroClim_12-max_cell	185
6.82	HydroClim_13-max_cell	186
6.83	HydroClim_14-max_cell	188
6.84	HydroClim_15-max_cell	189
6.85	HydroClim_16-max_cell	190
6.86	HydroClim_17-max_cell	192
6.87	HydroClim_18-max_cell	193
6.88	HydroClim_19-max_cell	195
6.89	Distance_Builtup_cell	196
6.90	Distance_ForestInside_cell	197
6.91	Distance_GrasslandPermanent_cell	198
6.92	Distance_Landfill_cell	199
6.93	Distance_Sea_cell	200
6.94	Distance_Trees_cell	201
6.95	Distance_Waste_cell	202
6.96	Distance_Water_cell	203
6.97	Distance_WaterInside_cell	204
6.98	Diversity_Farmland_r500	205
6.99	Diversity_Farmland_r1250	206
6.100	Diversity_Farmland_r3000	207
6.101	Diversity_Farmland_r10000	208
6.102	Diversity_Forest_r500	209
6.103	Diversity_Forest_r1250	210
6.104	Diversity_Forest_r3000	211
6.105	Diversity_Forest_r10000	212
6.106	Diversity_Total_r500	213
6.107	Diversity_Total_r1250	214
6.108	Diversity_Total_r3000	215
6.109	Diversity_Total_r10000	216
6.110	Edges_Bogs-Trees_cell	217
6.111	Edges_Bogs-Trees_r500	219
6.112	Edges_Bogs-Trees_r1250	220
6.113	Edges_Bogs-Trees_r3000	221
6.114	Edges_Bogs-Trees_r10000	222
6.115	Edges_Bogs-Water_cell	223
6.116	Edges_Bogs-Water_r500	224

6.117Edges_Bogs-Water_r1250 . . . . .	225
6.118Edges_Bogs-Water_r3000 . . . . .	226
6.119Edges_Bogs-Water_r10000 . . . . .	227
6.120Edges_Farmland-Builtup_cell . . . . .	228
6.121Edges_Farmland-Builtup_r500 . . . . .	230
6.122Edges_Farmland-Builtup_r1250 . . . . .	231
6.123Edges_Farmland-Builtup_r3000 . . . . .	232
6.124Edges_Farmland-Builtup_r10000 . . . . .	233
6.125Edges_Trees-Builtup_cell . . . . .	234
6.126Edges_Trees-Builtup_r500 . . . . .	235
6.127Edges_Trees-Builtup_r1250 . . . . .	236
6.128Edges_Trees-Builtup_r3000 . . . . .	237
6.129Edges_Trees-Builtup_r10000 . . . . .	239
6.130Edges_CropsFallow_cell . . . . .	240
6.131Edges_CropsFallow_r500 . . . . .	241
6.132Edges_CropsFallow_r1250 . . . . .	242
6.133Edges_CropsFallow_r3000 . . . . .	243
6.134Edges_CropsFallow_r10000 . . . . .	244
6.135Edges_FarmlandShrubs-Trees_cell . . . . .	245
6.136Edges_FarmlandShrubs-Trees_r500 . . . . .	246
6.137Edges_FarmlandShrubs-Trees_r1250 . . . . .	248
6.138Edges_FarmlandShrubs-Trees_r3000 . . . . .	249
6.139Edges_FarmlandShrubs-Trees_r10000 . . . . .	250
6.140Edges_Grasslands_cell . . . . .	251
6.141Edges_Grasslands_r500 . . . . .	252
6.142Edges_Grasslands_r1250 . . . . .	253
6.143Edges_Grasslands_r3000 . . . . .	254
6.144Edges_Grasslands_r10000 . . . . .	255
6.145Edges_OldForests_cell . . . . .	256
6.146Edges_OldForests_r500 . . . . .	258
6.147Edges_OldForests_r1250 . . . . .	259
6.148Edges_OldForests_r3000 . . . . .	260
6.149Edges_OldForests_r10000 . . . . .	261
6.150Edges_Roads_cell . . . . .	262
6.151Edges_Roads_r500 . . . . .	263
6.152Edges_Roads_r1250 . . . . .	264
6.153Edges_Roads_r3000 . . . . .	265
6.154Edges_Roads_r10000 . . . . .	266
6.155Edges_Trees_cell . . . . .	267
6.156Edges_Trees_r500 . . . . .	269
6.157Edges_Trees_r1250 . . . . .	270

6.158Edges_Trees_r3000 . . . . .	271
6.159Edges_Trees_r10000 . . . . .	272
6.160Edges_Water_cell . . . . .	273
6.161Edges_Water_r500 . . . . .	274
6.162Edges_Water_r1250 . . . . .	275
6.163Edges_Water_r3000 . . . . .	276
6.164Edges_Water_r10000 . . . . .	277
6.165Edges_Water-Farmland_cell . . . . .	278
6.166Edges_Water-Farmland_r500 . . . . .	280
6.167Edges_Water-Farmland_r1250 . . . . .	281
6.168Edges_Water-Farmland_r3000 . . . . .	282
6.169Edges_Water-Farmland_r10000 . . . . .	283
6.170Edges_Water-Grassland_cell . . . . .	284
6.171Edges_Water-Grassland_r500 . . . . .	285
6.172Edges_Water-Grassland_r1250 . . . . .	286
6.173Edges_Water-Grassland_r3000 . . . . .	287
6.174Edges_Water-Grassland_r10000 . . . . .	288
6.175Edges_ReedSedgeRushBeds-Water_cell . . . . .	289
6.176Edges_ReedSedgeRushBeds-Water_r500 . . . . .	291
6.177Edges_ReedSedgeRushBeds-Water_r1250 . . . . .	292
6.178Edges_ReedSedgeRushBeds-Water_r3000 . . . . .	293
6.179Edges_ReedSedgeRushBeds-Water_r10000 . . . . .	294
6.180FarmlandCrops_CropsAll_cell . . . . .	295
6.181FarmlandCrops_CropsAll_r500 . . . . .	297
6.182FarmlandCrops_CropsAll_r1250 . . . . .	298
6.183FarmlandCrops_CropsAll_r3000 . . . . .	299
6.184FarmlandCrops_CropsAll_r10000 . . . . .	300
6.185FarmlandCrops_CropsHoed_cell . . . . .	301
6.186FarmlandCrops_CropsHoed_r500 . . . . .	302
6.187FarmlandCrops_CropsHoed_r1250 . . . . .	303
6.188FarmlandCrops_CropsHoed_r3000 . . . . .	304
6.189FarmlandCrops_CropsHoed_r10000 . . . . .	305
6.190FarmlandCrops_CropsOther_cell . . . . .	306
6.191FarmlandCrops_CropsOther_r500 . . . . .	308
6.192FarmlandCrops_CropsOther_r1250 . . . . .	309
6.193FarmlandCrops_CropsOther_r3000 . . . . .	310
6.194FarmlandCrops_CropsOther_r10000 . . . . .	311
6.195FarmlandCrops_CropsSpring_cell . . . . .	312
6.196FarmlandCrops_CropsSpring_r500 . . . . .	314
6.197FarmlandCrops_CropsSpring_r1250 . . . . .	315
6.198FarmlandCrops_CropsSpring_r3000 . . . . .	316

6.199FarmlandCrops_CropsSpring_r10000 . . . . .	317
6.200FarmlandCrops_CropsWinter_cell . . . . .	318
6.201FarmlandCrops_CropsWinter_r500 . . . . .	319
6.202FarmlandCrops_CropsWinter_r1250 . . . . .	320
6.203FarmlandCrops_CropsWinter_r3000 . . . . .	321
6.204FarmlandCrops_CropsWinter_r10000 . . . . .	323
6.205FarmlandCrops_RapeseedsSpring_cell . . . . .	324
6.206FarmlandCrops_RapeseedsSpring_r500 . . . . .	325
6.207FarmlandCrops_RapeseedsSpring_r1250 . . . . .	326
6.208FarmlandCrops_RapeseedsSpring_r3000 . . . . .	327
6.209FarmlandCrops_RapeseedsSpring_r10000 . . . . .	328
6.210FarmlandCrops_RapeseedsWinter_cell . . . . .	329
6.211FarmlandCrops_RapeseedsWinter_r500 . . . . .	331
6.212FarmlandCrops_RapeseedsWinter_r1250 . . . . .	332
6.213FarmlandCrops_RapeseedsWinter_r3000 . . . . .	333
6.214FarmlandCrops_RapeseedsWinter_r10000 . . . . .	334
6.215FarmlandGrassland_GrasslandsAbandoned_cell . . . . .	335
6.216FarmlandGrassland_GrasslandsAbandoned_r500 . . . . .	336
6.217FarmlandGrassland_GrasslandsAbandoned_r1250 . . . . .	338
6.218FarmlandGrassland_GrasslandsAbandoned_r3000 . . . . .	339
6.219FarmlandGrassland_GrasslandsAbandoned_r10000 . . . . .	340
6.220FarmlandGrassland_GrasslandsAll_cell . . . . .	341
6.221FarmlandGrassland_GrasslandsAll_r500 . . . . .	342
6.222FarmlandGrassland_GrasslandsAll_r1250 . . . . .	343
6.223FarmlandGrassland_GrasslandsAll_r3000 . . . . .	344
6.224FarmlandGrassland_GrasslandsAll_r10000 . . . . .	345
6.225FarmlandGrassland_GrasslandsPermanent_cell . . . . .	346
6.226FarmlandGrassland_GrasslandsPermanent_r500 . . . . .	348
6.227FarmlandGrassland_GrasslandsPermanent_r1250 . . . . .	349
6.228FarmlandGrassland_GrasslandsPermanent_r3000 . . . . .	350
6.229FarmlandGrassland_GrasslandsPermanent_r10000 . . . . .	351
6.230FarmlandGrassland_GrasslandsTemporary_cell . . . . .	352
6.231FarmlandGrassland_GrasslandsTemporary_r500 . . . . .	353
6.232FarmlandGrassland_GrasslandsTemporary_r1250 . . . . .	355
6.233FarmlandGrassland_GrasslandsTemporary_r3000 . . . . .	356
6.234FarmlandGrassland_GrasslandsTemporary_r10000 . . . . .	357
6.235FarmlandParcels_FieldsActive_cell . . . . .	358
6.236FarmlandParcels_FieldsActive_r500 . . . . .	359
6.237FarmlandParcels_FieldsActive_r1250 . . . . .	360
6.238FarmlandParcels_FieldsActive_r3000 . . . . .	361
6.239FarmlandParcels_FieldsActive_r10000 . . . . .	362

6.240FarmlandPloughed_CropsFallow_cell . . . . .	363
6.241FarmlandPloughed_CropsFallow_r500 . . . . .	365
6.242FarmlandPloughed_CropsFallow_r1250 . . . . .	366
6.243FarmlandPloughed_CropsFallow_r3000 . . . . .	367
6.244FarmlandPloughed_CropsFallow_r10000 . . . . .	368
6.245FarmlandPloughed_CropsFallowTempGrass_cell . . . . .	369
6.246FarmlandPloughed_CropsFallowTempGrass_r500 . . . . .	371
6.247FarmlandPloughed_CropsFallowTempGrass_r1250 . . . . .	372
6.248FarmlandPloughed_CropsFallowTempGrass_r3000 . . . . .	373
6.249FarmlandPloughed_CropsFallowTempGrass_r10000 . . . . .	374
6.250FarmlandPloughed_Fallow_cell . . . . .	375
6.251FarmlandPloughed_Fallow_r500 . . . . .	377
6.252FarmlandPloughed_Fallow_r1250 . . . . .	378
6.253FarmlandPloughed_Fallow_r3000 . . . . .	379
6.254FarmlandPloughed_Fallow_r10000 . . . . .	380
6.255FarmlandSubsidies_BiologicalSubsidies_cell . . . . .	381
6.256FarmlandSubsidies_BiologicalSubsidies_r500 . . . . .	382
6.257FarmlandSubsidies_BiologicalSubsidies_r1250 . . . . .	383
6.258FarmlandSubsidies_BiologicalSubsidies_r3000 . . . . .	384
6.259FarmlandSubsidies_BiologicalSubsidies_r10000 . . . . .	385
6.260FarmlandTrees_PermanentCrops_cell . . . . .	386
6.261FarmlandTrees_PermanentCrops_r500 . . . . .	388
6.262FarmlandTrees_PermanentCrops_r1250 . . . . .	389
6.263FarmlandTrees_PermanentCrops_r3000 . . . . .	390
6.264FarmlandTrees_PermanentCrops_r10000 . . . . .	391
6.265FarmlandTrees_ShortRotationCoppice_cell . . . . .	392
6.266FarmlandTrees_ShortRotationCoppice_r500 . . . . .	394
6.267FarmlandTrees_ShortRotationCoppice_r1250 . . . . .	395
6.268FarmlandTrees_ShortRotationCoppice_r3000 . . . . .	396
6.269FarmlandTrees_ShortRotationCoppice_r10000 . . . . .	397
6.270ForestsAge_ClearcutsLowStands_cell . . . . .	398
6.271ForestsAge_ClearcutsLowStands_r500 . . . . .	400
6.272ForestsAge_ClearcutsLowStands_r1250 . . . . .	401
6.273ForestsAge_ClearcutsLowStands_r3000 . . . . .	402
6.274ForestsAge_ClearcutsLowStands_r10000 . . . . .	403
6.275ForestsAge_Middle_cell . . . . .	404
6.276ForestsAge_Middle_r500 . . . . .	406
6.277ForestsAge_Middle_r1250 . . . . .	407
6.278ForestsAge_Middle_r3000 . . . . .	408
6.279ForestsAge_Middle_r10000 . . . . .	409
6.280ForestsAge_Old_cell . . . . .	410

6.281ForestsAge_Old_r500 . . . . .	412
6.282ForestsAge_Old_r1250 . . . . .	413
6.283ForestsAge_Old_r3000 . . . . .	414
6.284ForestsAge_Old_r10000 . . . . .	415
6.285ForestsAge_YoungTallStandsShrubs_cell . . . . .	416
6.286ForestsAge_YoungTallStandsShrubs_r500 . . . . .	418
6.287ForestsAge_YoungTallStandsShrubs_r1250 . . . . .	419
6.288ForestsAge_YoungTallStandsShrubs_r3000 . . . . .	420
6.289ForestsAge_YoungTallStandsShrubs_r10000 . . . . .	421
6.290ForestsQuant_AgeProp-average_cell . . . . .	422
6.291ForestsQuant_DominantDiameter-max_cell . . . . .	425
6.292ForestsQuant_LargestDiameter-max_cell . . . . .	427
6.293ForestsQuant_TimeSinceDisturbance-average_cell . . . . .	430
6.294ForestsQuant_VolumeAspen-sum_cell . . . . .	432
6.295ForestsQuant_VolumeBirch-sum_cell . . . . .	435
6.296ForestsQuant_VolumeBlackAlder-sum_cell . . . . .	437
6.297ForestsQuant_VolumeBorealDeciduousOther-sum_cell . . . . .	440
6.298ForestsQuant_VolumeBorealDeciduousTotal-sum_cell . . . . .	442
6.299ForestsQuant_VolumeConiferous-sum_cell . . . . .	445
6.300ForestsQuant_VolumeOak-sum_cell . . . . .	447
6.301ForestsQuant_VolumeOakMaple-sum_cell . . . . .	450
6.302ForestsQuant_VolumePine-sum_cell . . . . .	452
6.303ForestsQuant_VolumeSpruce-sum_cell . . . . .	455
6.304ForestsQuant_VolumeTemperateDeciduousTotal-sum_cell . . . . .	457
6.305ForestsQuant_VolumeTemperateWithoutOak-sum_cell . . . . .	460
6.306ForestsQuant_VolumeTemperateWithoutOakMaple-sum_cell . . . . .	462
6.307ForestsQuant_VolumeTotal-sum_cell . . . . .	465
6.308ForestsSoil_EutrophicDrained_cell . . . . .	467
6.309ForestsSoil_EutrophicDrained_r500 . . . . .	469
6.310ForestsSoil_EutrophicDrained_r1250 . . . . .	470
6.311ForestsSoil_EutrophicDrained_r3000 . . . . .	471
6.312ForestsSoil_EutrophicDrained_r10000 . . . . .	472
6.313ForestsSoil_EutrophicMineral_cell . . . . .	473
6.314ForestsSoil_EutrophicMineral_r500 . . . . .	474
6.315ForestsSoil_EutrophicMineral_r1250 . . . . .	475
6.316ForestsSoil_EutrophicMineral_r3000 . . . . .	476
6.317ForestsSoil_EutrophicMineral_r10000 . . . . .	477
6.318ForestsSoil_EutrophicOrganic_cell . . . . .	478
6.319ForestsSoil_EutrophicOrganic_r500 . . . . .	480
6.320ForestsSoil_EutrophicOrganic_r1250 . . . . .	481
6.321ForestsSoil_EutrophicOrganic_r3000 . . . . .	482

6.322ForestsSoil_EutrophicOrganic_r10000 . . . . .	483
6.323ForestsSoil_MesotrophicMineral_cell . . . . .	484
6.324ForestsSoil_MesotrophicMineral_r500 . . . . .	485
6.325ForestsSoil_MesotrophicMineral_r1250 . . . . .	486
6.326ForestsSoil_MesotrophicMineral_r3000 . . . . .	487
6.327ForestsSoil_MesotrophicMineral_r10000 . . . . .	488
6.328ForestsSoil_OligotrophicDrained_cell . . . . .	490
6.329ForestsSoil_OligotrophicDrained_r500 . . . . .	491
6.330ForestsSoil_OligotrophicDrained_r1250 . . . . .	492
6.331ForestsSoil_OligotrophicDrained_r3000 . . . . .	493
6.332ForestsSoil_OligotrophicDrained_r10000 . . . . .	494
6.333ForestsSoil_OligotrophicMineral_cell . . . . .	495
6.334ForestsSoil_OligotrophicMineral_r500 . . . . .	497
6.335ForestsSoil_OligotrophicMineral_r1250 . . . . .	498
6.336ForestsSoil_OligotrophicMineral_r3000 . . . . .	499
6.337ForestsSoil_OligotrophicMineral_r10000 . . . . .	500
6.338ForestsSoil_OligotrophicOrganic_cell . . . . .	501
6.339ForestsSoil_OligotrophicOrganic_r500 . . . . .	502
6.340ForestsSoil_OligotrophicOrganic_r1250 . . . . .	503
6.341ForestsSoil_OligotrophicOrganic_r3000 . . . . .	504
6.342ForestsSoil_OligotrophicOrganic_r10000 . . . . .	505
6.343ForestsTreesAge_BorealDeciduousOld_cell . . . . .	506
6.344ForestsTreesAge_BorealDeciduousOld_r500 . . . . .	509
6.345ForestsTreesAge_BorealDeciduousOld_r1250 . . . . .	510
6.346ForestsTreesAge_BorealDeciduousOld_r3000 . . . . .	511
6.347ForestsTreesAge_BorealDeciduousOld_r10000 . . . . .	512
6.348ForestsTreesAge_BorealDeciduousYoung_cell . . . . .	513
6.349ForestsTreesAge_BorealDeciduousYoung_r500 . . . . .	516
6.350ForestsTreesAge_BorealDeciduousYoung_r1250 . . . . .	517
6.351ForestsTreesAge_BorealDeciduousYoung_r3000 . . . . .	518
6.352ForestsTreesAge_BorealDeciduousYoung_r10000 . . . . .	519
6.353ForestsTreesAge_ConiferousOld_cell . . . . .	520
6.354ForestsTreesAge_ConiferousOld_r500 . . . . .	522
6.355ForestsTreesAge_ConiferousOld_r1250 . . . . .	523
6.356ForestsTreesAge_ConiferousOld_r3000 . . . . .	524
6.357ForestsTreesAge_ConiferousOld_r10000 . . . . .	525
6.358ForestsTreesAge_ConiferousYoung_cell . . . . .	526
6.359ForestsTreesAge_ConiferousYoung_r500 . . . . .	529
6.360ForestsTreesAge_ConiferousYoung_r1250 . . . . .	530
6.361ForestsTreesAge_ConiferousYoung_r3000 . . . . .	531
6.362ForestsTreesAge_ConiferousYoung_r10000 . . . . .	532

6.363ForestsTreesAge_MixedOld_cell . . . . .	533
6.364ForestsTreesAge_MixedOld_r500 . . . . .	535
6.365ForestsTreesAge_MixedOld_r1250 . . . . .	536
6.366ForestsTreesAge_MixedOld_r3000 . . . . .	537
6.367ForestsTreesAge_MixedOld_r10000 . . . . .	538
6.368ForestsTreesAge_MixedYoung_cell . . . . .	540
6.369ForestsTreesAge_MixedYoung_r500 . . . . .	542
6.370ForestsTreesAge_MixedYoung_r1250 . . . . .	543
6.371ForestsTreesAge_MixedYoung_r3000 . . . . .	544
6.372ForestsTreesAge_MixedYoung_r10000 . . . . .	545
6.373ForestsTreesAge_TemperateDeciduousOld_cell . . . . .	546
6.374ForestsTreesAge_TemperateDeciduousOld_r500 . . . . .	548
6.375ForestsTreesAge_TemperateDeciduousOld_r1250 . . . . .	550
6.376ForestsTreesAge_TemperateDeciduousOld_r3000 . . . . .	551
6.377ForestsTreesAge_TemperateDeciduousOld_r10000 . . . . .	552
6.378ForestsTreesAge_TemperateDeciduousYoung_cell . . . . .	553
6.379ForestsTreesAge_TemperateDeciduousYoung_r500 . . . . .	555
6.380ForestsTreesAge_TemperateDeciduousYoung_r1250 . . . . .	556
6.381ForestsTreesAge_TemperateDeciduousYoung_r3000 . . . . .	557
6.382ForestsTreesAge_TemperateDeciduousYoung_r10000 . . . . .	558
6.383ForestsTrees_BorealDeciduous_cell . . . . .	559
6.384ForestsTrees_BorealDeciduous_r500 . . . . .	562
6.385ForestsTrees_BorealDeciduous_r1250 . . . . .	563
6.386ForestsTrees_BorealDeciduous_r3000 . . . . .	564
6.387ForestsTrees_BorealDeciduous_r10000 . . . . .	565
6.388ForestsTrees_Coniferous_cell . . . . .	566
6.389ForestsTrees_Coniferous_r500 . . . . .	568
6.390ForestsTrees_Coniferous_r1250 . . . . .	569
6.391ForestsTrees_Coniferous_r3000 . . . . .	570
6.392ForestsTrees_Coniferous_r10000 . . . . .	571
6.393ForestsTrees_Mixed_cell . . . . .	573
6.394ForestsTrees_Mixed_r500 . . . . .	575
6.395ForestsTrees_Mixed_r1250 . . . . .	576
6.396ForestsTrees_Mixed_r3000 . . . . .	577
6.397ForestsTrees_Mixed_r10000 . . . . .	578
6.398ForestsTrees_TemperateDeciduous_cell . . . . .	579
6.399ForestsTrees_TemperateDeciduous_r500 . . . . .	581
6.400ForestsTrees_TemperateDeciduous_r1250 . . . . .	582
6.401ForestsTrees_TemperateDeciduous_r3000 . . . . .	583
6.402ForestsTrees_TemperateDeciduous_r10000 . . . . .	585
6.403General_AllotmentGardens_cell . . . . .	586

6.404General_AllotmentGardens_r500 . . . . .	587
6.405General_AllotmentGardens_r1250 . . . . .	588
6.406General_AllotmentGardens_r3000 . . . . .	589
6.407General_AllotmentGardens_r10000 . . . . .	590
6.408General_BareSoilQuarry_cell . . . . .	591
6.409General_BareSoilQuarry_r500 . . . . .	592
6.410General_BareSoilQuarry_r1250 . . . . .	593
6.411General_BareSoilQuarry_r3000 . . . . .	594
6.412General_BareSoilQuarry_r10000 . . . . .	595
6.413General_Builtup_cell . . . . .	596
6.414General_Builtup_r500 . . . . .	597
6.415General_Builtup_r1250 . . . . .	598
6.416General_Builtup_r3000 . . . . .	599
6.417General_Builtup_r10000 . . . . .	600
6.418General_Farmland_cell . . . . .	601
6.419General_Farmland_r500 . . . . .	603
6.420General_Farmland_r1250 . . . . .	604
6.421General_Farmland_r3000 . . . . .	605
6.422General_Farmland_r10000 . . . . .	606
6.423General_ForestsWithoutInventory_cell . . . . .	607
6.424General_ForestsWithoutInventory_r500 . . . . .	608
6.425General_ForestsWithoutInventory_r1250 . . . . .	609
6.426General_ForestsWithoutInventory_r3000 . . . . .	610
6.427General_ForestsWithoutInventory_r10000 . . . . .	611
6.428General_GardensOrchards_cell . . . . .	612
6.429General_GardensOrchards_r500 . . . . .	613
6.430General_GardensOrchards_r1250 . . . . .	614
6.431General_GardensOrchards_r3000 . . . . .	615
6.432General_GardensOrchards_r10000 . . . . .	616
6.433General_Roads_cell . . . . .	617
6.434General_ShrubsOrchards_cell . . . . .	618
6.435General_ShrubsOrchards_r500 . . . . .	620
6.436General_ShrubsOrchards_r1250 . . . . .	621
6.437General_ShrubsOrchards_r3000 . . . . .	622
6.438General_ShrubsOrchards_r10000 . . . . .	623
6.439General_ShrubsOrchardsGardens_cell . . . . .	624
6.440General_ShrubsOrchardsGardens_r500 . . . . .	626
6.441General_ShrubsOrchardsGardens_r1250 . . . . .	627
6.442General_ShrubsOrchardsGardens_r3000 . . . . .	628
6.443General_ShrubsOrchardsGardens_r10000 . . . . .	629
6.444General_SwampsMiresBogsHelophytes_cell . . . . .	630

6.445General_SwampsMiresBogsHelophytes_r500 . . . . .	631
6.446General_SwampsMiresBogsHelophytes_r1250 . . . . .	632
6.447General_SwampsMiresBogsHelophytes_r3000 . . . . .	633
6.448General_SwampsMiresBogsHelophytes_r10000 . . . . .	634
6.449General_Trees_cell . . . . .	635
6.450General_Trees_r500 . . . . .	636
6.451General_Trees_r1250 . . . . .	637
6.452General_Trees_r3000 . . . . .	638
6.453General_Trees_r10000 . . . . .	639
6.454General_TreesOutsideForests_cell . . . . .	641
6.455General_TreesOutsideForests_r500 . . . . .	642
6.456General_TreesOutsideForests_r1250 . . . . .	643
6.457General_TreesOutsideForests_r3000 . . . . .	644
6.458General_TreesOutsideForests_r10000 . . . . .	645
6.459General_Water_cell . . . . .	646
6.460General_Water_r500 . . . . .	647
6.461General_Water_r1250 . . . . .	648
6.462General_Water_r3000 . . . . .	649
6.463General_Water_r10000 . . . . .	650
6.464Wetlands_Bogs_cell . . . . .	651
6.465Wetlands_Bogs_r500 . . . . .	652
6.466Wetlands_Bogs_r1250 . . . . .	653
6.467Wetlands_Bogs_r3000 . . . . .	654
6.468Wetlands_Bogs_r10000 . . . . .	655
6.469Wetlands_Mires_cell . . . . .	656
6.470Wetlands_Mires_r500 . . . . .	657
6.471Wetlands_Mires_r1250 . . . . .	658
6.472Wetlands_Mires_r3000 . . . . .	659
6.473Wetlands_Mires_r10000 . . . . .	660
6.474Wetlands_ReedSedgeRushBeds_cell . . . . .	661
6.475Wetlands_ReedSedgeRushBeds_r500 . . . . .	662
6.476Wetlands_ReedSedgeRushBeds_r1250 . . . . .	663
6.477Wetlands_ReedSedgeRushBeds_r3000 . . . . .	665
6.478Wetlands_ReedSedgeRushBeds_r10000 . . . . .	666
6.479EO_NDMI-LYmed-average_cell . . . . .	667
6.480EO_NDMI-LYmedian-iqr_cell . . . . .	667
6.481EO_NDMI-STiqr-median_cell . . . . .	668
6.482EO_NDMI-STmedian-average_cell . . . . .	669
6.483EO_NDMI-STmedian-iqr_cell . . . . .	670
6.484EO_NDMI-STp25-min_cell . . . . .	671
6.485EO_NDMI-STp75-max_cell . . . . .	672

6.486EO_NDVI-LYmedian-average_cell . . . . .	673
6.487EO_NDVI-LYmedian-iqr_cell . . . . .	674
6.488EO_NDVI-STiqr-median_cell . . . . .	675
6.489EO_NDVI-STmedian-average_cell . . . . .	676
6.490EO_NDVI-STmedian-iqr_cell . . . . .	676
6.491EO_NDVI-STp25-min_cell . . . . .	678
6.492EO_NDVI-STp75-max_cell . . . . .	678
6.493EO_NDWI-LYmedian-average_cell . . . . .	679
6.494EO_NDWI-LYmedian-iqr_cell . . . . .	680
6.495EO_NDWI-STiqr-median_cell . . . . .	681
6.496EO_NDWI-STmedian-average_cell . . . . .	682
6.497EO_NDWI-STmedian-iqr_cell . . . . .	683
6.498EO_NDWI-STp25-min_cell . . . . .	684
6.499EO_NDWI-STp75-max_cell . . . . .	685
6.500SoilChemistry_ESDAC-CN_cell . . . . .	685
6.501SoilChemistry_ESDAC-CaCo3_cell . . . . .	686
6.502SoilChemistry_ESDAC-K_cell . . . . .	687
6.503SoilChemistry_ESDAC-N_cell . . . . .	688
6.504SoilChemistry_ESDAC-P_cell . . . . .	689
6.505SoilChemistry_ESDAC-phH2O_cell . . . . .	690
6.506SoilTexture_Clay_cell . . . . .	690
6.507SoilTexture_Clay_r500 . . . . .	691
6.508SoilTexture_Clay_r1250 . . . . .	692
6.509SoilTexture_Clay_r3000 . . . . .	693
6.510SoilTexture_Clay_r10000 . . . . .	694
6.511SoilTexture_Organic_cell . . . . .	695
6.512SoilTexture_Organic_r500 . . . . .	696
6.513SoilTexture_Organic_r1250 . . . . .	697
6.514SoilTexture_Organic_r3000 . . . . .	698
6.515SoilTexture_Organic_r10000 . . . . .	699
6.516SoilTexture_Sand_cell . . . . .	700
6.517SoilTexture_Sand_r500 . . . . .	701
6.518SoilTexture_Sand_r1250 . . . . .	702
6.519SoilTexture_Sand_r3000 . . . . .	703
6.520SoilTexture_Sand_r10000 . . . . .	704
6.521SoilTexture_Silt_cell . . . . .	705
6.522SoilTexture_Silt_r500 . . . . .	706
6.523SoilTexture_Silt_r1250 . . . . .	707
6.524SoilTexture_Silt_r3000 . . . . .	708
6.525SoilTexture_Silt_r10000 . . . . .	709
6.526Terrain_ASL-average_cell . . . . .	710

6.527Terrain_Aspect-average_cell . . . . .	711
6.528Terrain_Aspect-iqr_cell . . . . .	712
6.529Terrain_DiS-area_cell . . . . .	713
6.530Terrain_DiS-area_r500 . . . . .	714
6.531Terrain_DiS-area_r1250 . . . . .	715
6.532Terrain_DiS-area_r3000 . . . . .	716
6.533Terrain_DiS-area_r10000 . . . . .	717
6.534Terrain_DiS-max_cell . . . . .	718
6.535Terrain_DiS-mean_cell . . . . .	719
6.536Terrain_Slope-average_cell . . . . .	720
6.537Terrain_Slope-iqr_cell . . . . .	721
6.538Terrain_TWI-average_cell . . . . .	722
<b>7 Data access</b>	<b>723</b>
<b>References</b>	<b>725</b>



# Preface

Welcome! This book documents the geodata and processing workflows used to create ecogeographical variables (EGVs) for species distribution modelling in Latvia (2024).

This material presents the results of three University of Latvia projects deeply rooted in species distribution modelling and, more importantly, explains the workflow and decisions made to ensure their repeatability and reproducibility. These projects are:

- The project “Preparation of a geospatial data layer covering existing protected areas for the implementation of the EU Biodiversity Strategy 2030” (No. 1-08/73/2023), funded by the Administrations of the Latvian Environmental Protection Fund;
- Scientific research service project commissioned by the Joint Stock Company “Latvijas valsts meži” (Latvian State Forests) “Improvement of the monitoring of the northern goshawk *Accipiter gentilis* and creation of a spatial model of habitat suitability” (Latvian State Forests document No. 5-5.5.1\_000r\_101\_23\_27\_6);
- State research program “Development of research specified in the Biodiversity Priority Action Program” project “High-resolution quantification of biodiversity for nature conservation and management: HiQBioDiv” (VPP-VARAM-DABA-2024/1-0002).

The material was developed in R using `{bookdown}`. The data processing and analysis described in the content was mainly performed in R, and one of the main reasons for creating this material was to transfer the information necessary for reproducing the work using verified command lines. A desirable side effect is to promote openness and reproducibility in scientific practice and practical science.

- Home repository of this material: [aavotins/HiQBioDiv\\_EGVs](#)
- Cite as needed using `book/book.bib`.

## About this material

This material **is not**:

- *an introduction to R or other programming language.* On the contrary, it will be most useful to those who already understand how to use command lines. However, it will also be informative for other users regarding the approaches used;
- *a tutorial on geoprocessing.* This material summarizes the approaches that, at the time of its development, were known to the authors as the most effective (in terms of processing time, RAM and hard disk space, performance guarantees, and reliability), but they are certainly not the only ones possible;
- *copy/paste ready product.* Although the use and publication of command lines tends to be intended for these purposes, in a situation where large amounts of data and, at least in part, restricted access data are used for the work, this is simply not possible. However, by ensuring data availability and placement in accordance with the file structure of this project (available at `root/Data` or by forking [template repository](#)), the command lines will be repeatable without changes and will produce the same results.

This material **has been** prepared to provide a reproducible workflow, describing the decisions made and solutions implemented in the preparation of ecogeographical variables for species distribution (habitat suitability) modelling for biodiversity conservation planning.

For the most part, this material consists of:

- *explanatory text*, which is recognizable as text;
- *command lines*, which are hidden by default to make the text easier to read. The locations of the command lines can be identified by the “|> Code” visible on the left side of the page, just below this paragraph. Clicking on it will open the code area, where the text on a grey background is command lines, for example:

```
object=function(arguments1,arguments2,  
path=".~/path/file/tree/object.extension")  
# comment
```

In the example above, the first line creates an object (“object”) that is the result of a function (“function()”). The function has three arguments (“arguments1”, “arguments2” and “path”) separated by commas (as with all function arguments in R). The third argument is the path in the file tree. It is on “a new line” but is a continuation of the function on the previous line, because the parentheses are not closed. Note the beginning “./”, which indicates a relative path - the location in the file tree is relative to the project location.

The second line of the example above is a comment - everything after “#” is a comment. Anything in a command line before “#” must be an executable function or object. A comment can contain anything and be on the same line as an executable function (at the end of it).

Command lines are the most important part of this material for reproducibility. However, the person using them must ensure the availability of input data and maintain correct paths in the file tree.

In this material code chunks are formatted as individual pieces to better pinpoint commands used for a job described in the text around. However, in practical setting the creation of ecogeographical variables will be much faster, if they will be combined in loops or other batch processing setup. Command lines used in practice are available in the [home repository](#) of this material at

Data/RScripts\_final, they can be executed in an alphanumeric order, if not specified differently. We performed parts of the compute on the University of Latvia Institute of Numerical Modelling HPC cluster with the same file tree as in this material. Shell scripts used to run R commands are available in the [home repository](#) of this material at Data/hpc\_io/Jobs\_shell/2024/EGVs.

Sometimes we will refer to R packages in the text, we will put them in curly brackets, for example, {package}.

- *graphics* - occasional diagrams that describe the workflow or data characteristics and maps;
- *links to other resources*, especially to higher-level products and results created within the project, as well as any publicly available data. The results are intended for practical use.

Within reason, the material describes all data sets used and provides metadata related to ensuring reproducibility. Since not all data sets are freely available, they are not published as such, but in all cases information is provided on how they were obtained for the development of this project.

## Outline

1. [Terminology and acronyms](#)
2. [Utilities](#)
3. [Template files](#)
4. [Raw geodata](#)
5. [Geodata products](#)
6. [Ecogeographical variables](#)
7. [Data access](#)

# Chapter 1

## Terminology and acronyms

Athough all georeferenced data can be considered *geodata*, in this material we use the following terms in the order listed below in our workflows:

- **raw geodata** - considered as raw data obtained for a harmonised description of the environment. This may include tables with coordinates, raster or vector data. It can be anything that has been or can be used to create *ecogeographical variables*, with or without slight processing.
- **geodata product** - processed *raw geodata* that have undergone heavy modifications, e.g. spatial overlays and combinations of different sets of *raw geodata*, and are used as *input data*. In this document, *geodata products* are categorical raster layers that match the *CRS* and the pixel locations of *input data*. When split by categories, they become *input data*. The processing step of creating *geodata products* is necessary when decisions about the order of spatial overlays are important. For example, in a high-resolution pixel, there can only be water or forest, if the edge between water and forest need to be calculated.
- **input data or input layers** - very-high resolution (multiple times higher than that used for *ecogeographical variables*) raster data that are the direct input for the creation of most of the *ecogeographical variables*. The creation of such layers is particularly useful alongside *geodata products*, as dealing with border misalignment or decisions regarding the order of spatial overlays, as well as simple geo-processing, is much faster with raster data.
- **ecogeographical variables (EGVs)** - this is the final product of the workflow describing environment for statistical analysis (e.g. *species distribution modelling*). They are suitable also for publishing due to standardisation of the values. In other words, these are standardised landscape ecological variables in the form of high-resolution raster layers (we use 1 ha cells). Each layer contains values representing the environment within the cell footprint or a summary of focal neighbours. In our case, each layer is of quantitative data describing a natural quantity (e.g. timber volume, mean annual temperature), or quantified information of categories (e.g. the fraction of class's area in an analysis cell or some neighbourhood, the number of pixels creating an edge of a certain class or between two classes in the analysis cell or some neighbourhood). The values of each layer are standardised: for each cell, the layer mean is subtracted and the result is divided by the root mean square error. Therefore, the values are more suitable for modelling, and the layers can be made publicly available as they do not directly provide exact sensitive information.

In this material, we use the term *species distribution modelling (SDM)* as a more broadly used term, that is synonymous with *ecological niche analysis* and *ecological niche modelling*.

Tree species groups:

**coniferous** - following species (codes) as used in the national forest stand-level-inventory database:

- pines (1, 14, 22)
- spruces (3, 15)
- larch (13)

- firs (23, 28)

**boreal deciduous** - following species (codes) as used in the national forest stand-level-inventory database:

- birches (4)
- black alder (6)
- aspens (8, 19, 68)
- grey alder (9)
- willows (20, 21)
- rowan (32)
- eve (35)

**temperate deciduous** - following species (codes) as used in the national forest stand-level-inventory database:

- oaks (10, 61)
- ashes (11, 64)
- lindens (12, 62)
- elms (16, 65)
- beech (17)
- hornbeam (18)
- maples (24, 63)
- cherry (25)
- apple (26)
- pear (27)
- yew (29)
- acacia (50)
- walnut (66)
- chestnut (67)
- robinia (69)

Forest stand **age groups** (*vgr*) as used in the national forest stand-level-inventory database:

- young stands (*vgr* = 1) in coniferous trees, ashes and oaks - until 40 years, in grey alder - until 10 years, in other tree species - until 20 years;
- medium aged stands (*vgr* = 2 or *vgr* = 3) are between young stands (*vgr* = 1) and legal rotation age;
- old stands (*vgr* = 4 or *vgr* = 5) are stands exceeding legal rotation age. This is defined in [by law](#) based on tree species and site quality class (bonity). Generally for oaks, pines and larches it is 101 or 121 years, for spruces, ashes, limes, elms and maples it is 81 years, for birches it is 71 or 51 years, for black alder it is 71 years, for aspens it is 41 years. Currently, there is no minimum rotation age in grey alder. We used 35 years, as it is the age of the youngest stand registered as “full grown” in the database. This was necessary for the harmonization of EGVs throughout forests.

Acronyms:

**CRS** - coordinate reference system

**DW** - [Dynamic World](#)

**EDI** - Institute of Electronics and Computer Sciences

**EGV** - ecogeographical variables

**GEE** - Google Earth Engine

**SDM** - species distribution modelling

**SDMs** - species distribution models

**LAD** - Rural Support Service

**LGIA** - Latvian Geospatial Information Agency

**LULC** - Land use and land cover

**LU** - University of Latvia

**LU GZZF** - University of Latvia Faculty of Geography and Earth Sciences

**LVM** - state owned Joint Stock Company “Latvia’s State Forests”

**LVMI Silava** - Latvian State Forest Research Institute “Silava”

**NDMI** - normalized difference moisture index

**NDVI** - normalized difference vegetation index

**NDWI** - normalized difference water index

**MVR** - State Forest Service’s stand level inventory database “Forest State Registry”

**VMD** - State Forest Service



# Chapter 2

## Utilities

This chapter provides a brief description of the utility functions used in this material. Most of these functions are available in the R package `{egvtools}`, which was created specifically for this work.

### 2.1 R package `egvtools`

`{egvtools}` provides a coherent set of wrappers and utilities that facilitate the reproducible and efficient creation of large-scale EGVs on real datasets. The package relies on robust building blocks — `{terra}`, `{sf}`, `{sfarrow}`, `{exactextractr}` and `{whitebox}` — and standardises input/output data, naming conventions and multi-scale zonal statistics, ensuring that the pipelines are repeatable across machines and projects.

The package was developed for the project ‘HiQBioDiv: High-resolution quantification of biodiversity for conservation and management’, which was funded by the Latvian Council of Science (Ref. No. VPP-VARAM-DABA-2024/1-0002), to simplify our work and to facilitate the reproduction of our results. Five of the functions are strictly for replication, while others are useful for a wider audience.

Package can be installed from [GitHub](#) with:

```
# install.packages("pak")
pak::pak("aavotins/egvtools")
```

or obtained as a [Docker container](#) with all the necessary system and software dependencies.

#### 2.1.1 Reproduction only functions

These functions are small wrappers, that help to recreate our working environments - template files and their locations in the file tree.

These functions are:

- [`download\_raster\_templates\(\)`](#) — fetch template rasters from Zenodo repository and place them in a user specified location on the disk, or by default - the place we used. By default this function links to the [version 2.0.0](#) of the dataset;
- [`download\_vector\_templates\(\)`](#) - fetch template vector grids/points from Zenodo repository and place them in a user specified location on the disk, or by default - the place we used. By default this function links to the [version 1.0.1](#) of the dataset;
- [`radius\_function\(\)`](#) — extracts summary statistics from raster layers using buffered polygon zones of multiple radii and rasterises them onto a common template grid. Internally hard coded to use filenames (first and second part in the result of tiling functions) as used in this project. If the filenames are kept, function can easily be used for other projects, regions etc. Function can be used to run sequentially, however much faster compute will be with parallel computing. If fast swap disk is available, this function needs only c.a. 5 GiBs of RAM per worker to perform tasks in this project. However, if the swap disk is not available, at least 20 GiBs of RAM per worker need to be assigned.

## 2.1.2 General purpose functions

Each of those functions are small workflows themselves that can be combined into larger workflows and used more widely than for Latvia.

- [`tile\_vector\_grid\(\)`](#) — tile template (vector) grid for chunked processing. The function internally is linked to our file naming convention. As long as it is maintained, function can be used to create tiled grid from any {sfarrow} parquet grid file;
- [`tiled\_buffers\(\)`](#) — precompute buffered tiles for multiple radii around points. The function internally is linked to our file naming convention. As long as it is maintained, function can be used to create tiled polygons with buffers around points from any {sfarrow} parquet grid file. There are three buffering modes: **dense** (buffers the best-matching pts100\*.parquet (prefers pts100\_sauzeme.parquet) for each tile by radii\_dense (default: 500, 1250, 3000, 10000 m ensuring that every analysis grid cell has desired buffer. Computationally heavy in the following workflows), **sparse** (uses a file to radius mapping and is highly generalizable), and **specified** (the same as sparse, but with one single point file). **In our workflows we used the sparse mode with default mapping**;
- [`create\_backgrounds\(\)`](#) — a wrapper around `terra::ifel()` to build consistent background rasters. This function better guards coordinate reference system and how it is stored, while also guarding spatial cover, resolution, coordinate reference system, exact pixel matching, etc. Creation of layers with default background values is faster than recreating them several times in workflows preparing EGVs;
- [`polygon2input\(\)`](#) — rasterise polygons to input layers. Handles only polygon data, other geometry types need to be buffered. Rasterizes polygon/multipolygon sf data to a raster aligned to a template GeoTIFF. Rasterization targets a raster::RasterLayer built from the template (so grids normally match). Projection is optional (project\_mode). Missing values are counted only over valid template cells. User may optionally restrict the result with a raster mask (restrict\_to) using numeric values or bracketed range strings (e.g., “(0,5]”, “[10,)”). Remaining NA cells can be filled by covering with a background raster (background\_raster) or a constant (background\_value). For large rasters, heavy steps (projection/mask/cover) can stream to disk via terra\_todisk=TRUE.
- [`input2egv\(\)`](#) — normalize/align a fine-resolution input raster to a (coarser) EGV template, optionally cover missing values and/or fill gaps (IDW via Whitebox), and write the result to disk. Designed for large runs: fast gap counting (inside template footprint only), optional filling, tuned GDAL write options, and controlled terra memory/temp behavior.
- [`downscale2egv\(\)`](#) — downscale coarse rasters to a template grid (CRS, resolution, extent), masks to the template footprint, and optionally: (1) fills NoData gaps using WhiteboxTools’ IDW-based fill\_missing\_data, and (2) applies IDW smoothing to reduce blockiness from low-resolution inputs.
- [`distance2egv\(\)`](#) — computes Euclidean distance (in map units) from cells matching a set of class values in an input raster to all cells of an EGV template grid, then writes a Float32 GeoTIFF aligned to the template. Designed to work with rasters produced by `polygon2input()`.
- [`landscape\_function\(\)`](#) — computes a {landscapemetrics} metric (default “lsm\_l\_shdi”), optionally with extra lm\_args, that yields one value per zone and per input layer. Runs tile-by-tile (by tile\_field), writes per-tile rasters, merges to final per-layer GeoTIFF(s), then performs gap analysis (NA count within the template footprint and optional maximum gap width) and optional IDW gap filling via WhiteboxTools. Returns a compact `data.frame` with per-layer stats and timing. Function can be used to run sequentially, however much faster compute will be with parallel computing. If fast swap disk is available, this function needs only 3 GiBs of RAM per worker to perform tasks in this project. However, if the swap disk is not available, at least 20 GiBs of RAM per worker need to be assigned.

## 2.2 Other utility functions

Other handy functions repeatedly used, not included in {egvtools} are stored in `egvs02.02_UtilityFunctions.R` file, located in `Data/RScripts_final`.

- `ensure_multipolygons()` - rather aggressive function to create MULTIPOLYGON geometries from GEOMETRYCOLLECTION

```
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(gdalUtilities)) {install.packages("gdalUtilities"); require(gdalUtilities)}

ensure_multipolygons <- function(X) {
  library(sf)
  library(gdalUtilities)

  tmp1 <- tempfile(fileext = ".gpkg")
  tmp2 <- tempfile(fileext = ".gpkg")
  st_write(X, tmp1)
  ogr2ogr(tmp1, tmp2, f = "GPKG", nlt = "MULTIPOLYGON")
  Y <- st_read(tmp2)
  st_sf(st_drop_geometry(X), geom = st_geometry(Y))
}
```



# Chapter 3

## Templates files

This chapter defines template files. They define the analysis space and ensure harmonisation of georeferenced data creation, and facilitate connection with other Latvian geodata.

### 3.1 Vector data

Baseline template (or reference) vector grid and point files are publicly available in the [HiQBioDiv's Zenodo repository](#). The command lines and data used to create these files are documented in the HiQBioDiv main code repository's [file](#).

The easiest way to obtain these files is to run determined function `download_vector_templates()` from `{egvtools}`.

```
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

download_vector_templates(
  url = "https://zenodo.org/api/records/14277114/files-archive",
  grid_dir = "./Templates/TemplateGrids",
  points_dir = "./Templates/TemplateGridPoints",
  gpkg_dir = "./Templates",
  overwrite = FALSE,
  quiet = FALSE
)
```

Once template vector data are downloaded and unarchived, they need to be tiled:

1. Analysis grid is tiled in `tks50km` pages

```
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

tile_vector_grid(
  grid_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  out_dir = "./Templates/TemplateGrids/tiles",
  tile_field = "tks50km",
  chunk_size = 50000L,
  overwrite = FALSE,
  quiet = FALSE
)
```

Expect to see warning: This is an initial implementation of Parquet/Feather file support and geo metadata. This is tracking version 0.1.0 of the metadata (<https://github.com/geopandas/geo-arrow-spec>). This metadata specification may change and does not yet make stability promises. We do not yet recommend using this in a production setting unless you are able to rewrite your Parquet/Feather files.

2. Point files are tiled and buffered. In the workflows creating EGVs described in this document, we used a “sparse” grid:

- 500m buffers around every 100m grid cell’s centre;
- 1250m buffers around every 100m grid cell’s centre;
- 3000m buffers around every 300m grid cell’s centre (to speed up neighbourhood analysis ~9 times, while loosing <0.001% of precision);
- 10000m buffers around every 1000m grid cell’s centre (to speed up neighbourhood analysis ~100 times, while loosing <0.001% of precision)

```
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

tiled_buffers(
  in_dir = "./Templates/TemplateGridPoints",
  out_dir = "./Templates/TemplateGridPoints/tiles",
  buffer_mode = "sparse",
  mapping_sparse = list("pts100_sauzeme.parquet" = c(500, 1250),
                        "pts300_sauzeme.parquet" = 3000,
                        "pts1000_sauzeme.parquet" = 10000),
  split_field = "tks50km",
  n_workers = 4,
  future_max_mem_gb = 4,
  overwrite = FALSE,
  quiet = FALSE
)
```

Expect to see warning: This is an initial implementation of Parquet/Feather file support and geo metadata. This is tracking version 0.1.0 of the metadata (<https://github.com/geopandas/geo-arrow-spec>). This metadata specification may change and does not yet make stability promises. We do not yet recommend using this in a production setting unless you are able to rewrite your Parquet/Feather files.

Appearance of file pts300\_r3000\_NA.parquet, i.e. without a tile number, is expected, due to slight mismatch of 300 m grid with the 50 km one.

## 3.2 Raster data

Baseline template (or reference) raster grid and point files are publically available in the [HiQBioDiv’s Zenodo repository](#). The command lines and data used to create these files are documented in the HiQBioDiv main code repository’s [file](#).

The easiest way to obtain these files is to run determined function `download_raster_templates()` from `{egv-tools}`.

```
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)

download_raster_templates(
  url = "https://zenodo.org/api/records/14497070/files-archive",
  out_dir = "./Templates/TemplateRasters",
  overwrite = TRUE,
  quiet = FALSE
)
```

During EGV creation, background filling to handle missing values may be necessary. For all EGVs described in this document where such an exercise might be required, the variables can be considered quantities of ratio scale, therefore backgrounds with value 0 are created.

```
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
```

```
create_backgrounds(in_dir="./Templates/TemplateRasters/",  
                  out_dir = "./Templates/TemplateRasters/",  
                  background_value = 0,  
                  out_prefix = "nulls_",  
                  overwrite=TRUE)
```



# Chapter 4

## Raw geodata

This chapter describes raw geodata used and the preliminary processing conducted on them.

### 4.1 State Forest Service’s State Forest Register

The State Forest Service’s State Forest Register database (ESRI file geodatabase), which compiles indicators and spatial data characterizing forest compartments (stand level inventory database), was received by the University of Latvia on January 7, 2024, to support study and research processes. The structure of the received database version corresponds to the [State Forest Register Forest Inventory File Structure](#), but lowercase letters are used in field names.

After downloading, the CRS is guarded, geometries are checked and saved in GeoParquet format.

Files are stored at `Geodata/2024/MVR/`.

```
# libs
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(arrow)) {install.packages("arrow"); require(arrow)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(gdalUtilities)) {install.packages("gdalUtilities"); require(gdalUtilities)}

# database
nog=read_sf("./Geodata/2024/MVR/VMD.gdb/", layer="Nogabali_pilna_datubaze")

# ensuring geometries
source("./RScripts_final/egvs02.02.UtilityFunctions.R")
nogabali <- ensure_multipolygons(nog)

# securing geometries
nogabali2 = nogabali[!st_is_empty(nogabali),,drop=FALSE] # 108 tukšas géometrijas
validity=st_is_valid(nogabali2)
table(validity) # 1733 invalid géometrijas
nogabali3=st_make_valid(nogabali2)

# transforming CRS
nogabali4=st_transform(nogabali3, crs=3059)

# saving
sfarrow::st_write_parquet(nogabali4, "./Geodata/2024/MVR/nogabali_2024janv.parquet")
```

### 4.2 Rural Support Service’s information on declared fields

The Rural Support Service maintains [regularly updated information on their open data portal](#). An archive (since 2016) is also available there, and the data of interest contain the keyword “deklarētās platības”.

After downloading files to Geodata/2024/LAD/downloads/, they are unzipped and read into R. Files are checked, empty geometries are deleted and the rest are validated. Then, all individual files are combined into one, which is saved in GeoPackage and GeoParquet formats at Geodata/2024/LAD/. At the end, downloaded files are unlinked.

```
# libs
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(arrow)) {install.packages("arrow"); require(arrow)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(gdalUtilities)) {install.packages("gdalUtilities"); require(gdalUtilities)}

# reading all files
faili=data.frame(celi=list.files("./Geodata/2024/LAD/downloads", full.names = TRUE))
dati=st_read(faili$celi[1])
for(i in 2:length(faili$celi)){
  nakosais=st_read(faili$celi[i])
  dati=bind_rows(dati,nakosais)
  print(nrow(dati))
}

# ensuring geometries
source("./RScripts_final/egvs02.02.UtilityFunctions.R")
nogabali <- ensure_multipolygons(nog)
dati2 <- ensure_multipolygons(dati)
dati3 = dati2[!st_is_empty(dati2),,drop=FALSE] # viss kārtībā
table(st_is_valid(dati3))
dati4=st_make_valid(dati3)
table(st_is_valid(dati4))
dati5 <- ensure_multipolygons(dati4)
table(st_is_valid(dati5))

# saving output
st_write(dati5,"./Geodata/2024/LAD/Lauki_2024.gpkg",append = FALSE)
sfarrow::st_write_parquet(dati5,"./Geodata/2024/LAD/Lauki_2024.parquet")

# unlinking downloads
for(i in seq_along(faili$celi)){
  unlink(faili$celi[i])
}
rm(list=ls())
```

## 4.3 Melioration Cadaster

The Land Improvement Cadastre Information System database was downloaded layer by layer from Geoserver. Geometries were tested and validated for each layer, and layers were all combined into a single GeoPackage file stored at Geodata/2024/MKIS/.

Initially, no additional processing was performed on this data. It was used to prepare [Geodata products](#) - both [Terrain products](#) and [Landscape classification](#).

```
# libs
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(httr)) {install.packages("httr"); require(httr)}
if(!require(ows4R)) {install.packages("ows4R"); require(ows4R)}

# basis information ----
link="https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
url=parse_url(link)
url$query <- list(service = "wfs",
                   #version = "2.0.0", # facultative
                   request = "GetCapabilities")
```

```

request <- build_url(url)
request
bwk_client <- WFSClient$new(link,
                             serviceVersion = "2.0.0")
bwk_client
bwk_client$getFeatureTypes(pretty = TRUE)

# dams ----

bwk_client$getFeatureTypes(pretty = TRUE)
url$query <- list(service = "wfs",
                  request = "GetFeature",
                  srsName="EPSG:3059",
                  typename = "zmni:zmni_dam")
request <- build_url(url)
aizsargdambji <- read_sf(request)
aizsargdambji = aizsargdambji %>% st_set_crs(st_crs(3059))
aizsargdambji=st_cast(aizsargdambji,"MULTILINESTRING")

ggplot(aizsargdambji)+geom_sf()

table(st_is_valid(aizsargdambji))

write_sf(aizsargdambji,
         "./Geodata/2024/MKIS/MKIS_2025.gpkg",
         layer="Aizsargdambji",
         append=FALSE)
rm(aizsargdambji)

# watercourses ----

bwk_client$getFeatureTypes(pretty = TRUE)
url$query <- list(service = "wfs",
                  request = "GetFeature",
                  srsName="EPSG:3059",
                  typename = "zmni:zmni_watercourses")
request <- build_url(url)

DabiskasUdensteces <- read_sf(request)
DabiskasUdensteces = DabiskasUdensteces %>% st_set_crs(st_crs(3059))
DabiskasUdensteces=st_cast(DabiskasUdensteces,"MULTILINESTRING")

ggplot(DabiskasUdensteces)+geom_sf()

table(st_is_valid(DabiskasUdensteces))

write_sf(DabiskasUdensteces,
         "./Geodata/2024/MKIS/MKIS_2025.gpkg",
         layer="DabiskasUdensteces",
         append=FALSE)
rm(DabiskasUdensteces)

# dam pickets ----

bwk_client$getFeatureTypes(pretty = TRUE)
url$query <- list(service = "wfs",
                  request = "GetFeature",
                  srsName="EPSG:3059",
                  typename = "zmni:zmni_dampicket")
request <- build_url(url)

DambjuPiketi <- read_sf(request)

```

```

DambjuPiketi = DambjuPiketi %>% st_set_crs(st_crs(3059))
DambjuPiketi=st_cast(DambjuPiketi,"POINT")

ggplot(DambjuPiketi)+geom_sf()

table(st_is_valid(DambjuPiketi))

write_sf(DambjuPiketi,
        "./Geodata/2024/MKIS/MKIS_2025.gpkg",
        layer="DambjuPiketi",
        append=FALSE)
rm(DambjuPiketi)

# drainpipes ----

bwk_client$getFeatureTypes(pretty = TRUE)

base_url <- "https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
type_name <- "zmni:zmni_drainpipes"
crs_code <- 3059
chunk_size <- 100000
gpkg_path <- "./Geodata/2024/MKIS/temp_MKIS_2025.gpkg"
layer_name <- "temp_Drenas"
i <- 0

repeat {
  message("Fetching features ", i * chunk_size + 1, " to ", (i + 1) * chunk_size, "...")

  query <- list(
    service = "WFS",
    version = "2.0.0",
    request = "GetFeature",
    typename = type_name,
    srsName = paste0("EPSG:", crs_code),
    count = chunk_size,
    startIndex = i * chunk_size
  )

  req_url <- modify_url(base_url, query = query)

  try({
    chunk <- read_sf(req_url)
    if (nrow(chunk) == 0) break

    # Set CRS and cast to MULTILINESTRING
    chunk <- chunk %>%
      st_set_crs(st_crs(crs_code)) %>%
      st_cast("MULTILINESTRING")

    # Write chunk to GeoPackage (append mode after first)
    st_write(
      chunk,
      dsn = gpkg_path,
      layer = layer_name,
      append = i != 0,
      quiet = FALSE
    )

    i <- i + 1
  }, silent = TRUE)
}

message("All chunks written to ", gpkg_path, " in layer ", layer_name)

Drenas_all=st_read("./Geodata/2024/MKIS/temp_MKIS_2025.gpkg",

```

```

        layer="temp_Drenas")
Drenas_all2 = Drenas_all[!st_is_empty(Drenas_all),,drop=FALSE] # 1
table(st_is_valid(Drenas_all2))

write_sf(Drenas_all2,
        "./Geodata/2024/MKIS/MKIS_2025.gpkg",
        layer="Drenas",
        append=FALSE)
rm(list=ls())

# drain collectors ----

bwk_client$getFeatureTypes(pretty = TRUE)

# geoms
bwk_client$getFeatureTypes(pretty = TRUE)
url$query <- list(service = "wfs",
                   request = "GetFeature",
                   srsName="EPSG:3059",
                   typename = "zmni:zmni_draincollectors",
                   count=1)
request <- build_url(url)

geometrijam <- read_sf(request)
geometrijam

# count
url$query <- list(service = "wfs",
                   request = "GetFeature",
                   srsName="EPSG:3059",
                   typename = "zmni:zmni_draincollectors",
                   resultType="hits")
request <- build_url(url)
result <- GET(request)
parsed <- xml2::as_list(content(result, "parsed"))
n_features <- attr(parsed$FeatureCollection, "numberMatched")
n_features

# download
base_url <- "https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
type_name <- "zmni:zmni_draincollectors"
crs_code <- 3059
chunk_size <- 100000
gpkg_path <- "./Geodata/2024/MKIS/temp_MKIS_2025.gpkg"
layer_name <- "temp_DrenuKolektori"
i <- 0

repeat {
  message("Fetching features ", i * chunk_size + 1, " to ", (i + 1) * chunk_size, "...")
  query <- list(
    service = "WFS",
    version = "2.0.0",
    request = "GetFeature",
    typename = type_name,
    srsName = paste0("EPSG:", crs_code),
    count = chunk_size,
    startIndex = i * chunk_size
  )
}

```

```

req_url <- modify_url(base_url, query = query)

try({
  chunk <- read_sf(req_url)
  if (nrow(chunk) == 0) break

  # Set CRS and cast to MULTILINESTRING
  chunk <- chunk %>%
    st_set_crs(st_crs(crs_code)) %>%
    st_cast("MULTILINESTRING")

  # Write chunk to GeoPackage (append mode after first)
  st_write(
    chunk,
    dsn = gpkg_path,
    layer = layer_name,
    append = i != 0,
    quiet = FALSE
  )

  i <- i + 1
}, silent = TRUE)
}

message("All chunks written to ", gpkg_path, " in layer ", layer_name)

DrenuKolektori_all=st_read("./Geodata/2024/MKIS/temp_MKIS_2025.gpkg",
                           layer="temp_DrenuKolektori")
DrenuKolektori_all2 = DrenuKolektori_all[!st_is_empty(DrenuKolektori_all),,drop=FALSE] # 1
table(st_is_valid(DrenuKolektori_all2))

write_sf(DrenuKolektori_all2,
        "./Geodata/2024/MKIS/MKIS_2025.gpkg",
        layer="DrenuKolektori",
        append=FALSE)
rm(list=ls())


# drainage network structures ----

link="https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
url=parse_url(link)

url$query <- list(service = "wfs", request = "GetCapabilities")
request <- build_url(url)
bwk_client <- WFSClient$new(link, serviceVersion = "2.0.0")

bwk_client$getFeatureTypes(pretty = TRUE)

# geoms

url$query <- list(service = "wfs",
                    request = "GetFeature",
                    srsName="EPSG:3059",
                    typename = "zmni:zmni_networkstructures",
                    count=1)
request <- build_url(url)

geometrijam <- read_sf(request)
geometrijam

```

```

# download
base_url <- "https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
type_name <- "zmni:zmni_networkstructures"
crs_code <- 3059
chunk_size <- 100000
gpkg_path <- "./Geodata/2024/MKIS/temp_MKIS_2025.gpkg"
layer_name <- "temp_DrenazasTiklaBuvues"
i <- 0

repeat {
  message("Fetching features ", i * chunk_size + 1, " to ", (i + 1) * chunk_size, "...")

  query <- list(
    service = "WFS",
    version = "2.0.0",
    request = "GetFeature",
    typename = type_name,
    srsName = paste0("EPSG:", crs_code),
    count = chunk_size,
    startIndex = i * chunk_size
  )

  req_url <- modify_url(base_url, query = query)

  try({
    chunk <- read_sf(req_url)
    if (nrow(chunk) == 0) break

    # Set CRS and cast to MULTILINESTRING
    chunk <- chunk %>%
      st_set_crs(st_crs(crs_code)) %>%
      st_cast("POINT")

    # Write chunk to GeoPackage (append mode after first)
    st_write(
      chunk,
      dsn = gpkg_path,
      layer = layer_name,
      append = i != 0,
      quiet = FALSE
    )

    i <- i + 1
  }, silent = TRUE)
}

message("All chunks written to ", gpkg_path, " in layer ", layer_name)

DrenazasTiklaBuvues_all<-st_read("./Geodata/2024/MKIS/temp_MKIS_2025.gpkg",
                                    layer="temp_DrenazasTiklaBuvues")
DrenazasTiklaBuvues_all2 =
  DrenazasTiklaBuvues_all[!st_is_empty(DrenazasTiklaBuvues_all),,drop=FALSE] # 0
table(st_is_valid(DrenazasTiklaBuvues_all2))

write_sf(DrenazasTiklaBuvues_all2,
        "./Geodata/2024/MKIS/MKIS_2025.gpkg",
        layer="DrenazasTiklaBuvues",
        append=FALSE)
rm(list=ls())

# dithces -----

```

```

link="https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
url=parse_url(link)

url$query <- list(service = "wfs", request = "GetCapabilities")
request <- build_url(url)
bwk_client <- WFSClient$new(link, serviceVersion = "2.0.0")

bwk_client$getFeatureTypes(pretty = TRUE)

# geoms

url$query <- list(service = "wfs",
                     request = "GetFeature",
                     srsName="EPSG:3059",
                     typename = "zmni:zmni_ditches",
                     count=100)
request <- build_url(url)

geometrijam <- read_sf(request)
geometrijam

# download
base_url <- "https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
type_name <- "zmni:zmni_ditches"
crs_code <- 3059
chunk_size <- 100000
gpkg_path <- "./Geodata/2024/MKIS/temp_MKIS_2025.gpkg"
layer_name <- "temp_Gravji"
i <- 0

repeat {
  message("Fetching features ", i * chunk_size + 1, " to ", (i + 1) * chunk_size, "...")

  query <- list(
    service = "WFS",
    version = "2.0.0",
    request = "GetFeature",
    typename = type_name,
    srsName = paste0("EPSG:", crs_code),
    count = chunk_size,
    startIndex = i * chunk_size
  )

  req_url <- modify_url(base_url, query = query)

  try({
    chunk <- read_sf(req_url)
    if (nrow(chunk) == 0) break

    # Set CRS and cast to MULTILINESTRING
    chunk <- chunk %>%
      st_set_crs(st_crs(crs_code)) %>%
      st_cast("MULTILINESTRING")

    # Write chunk to GeoPackage (append mode after first)
    st_write(
      chunk,
      dsn = gpkg_path,
      layer = layer_name,
      append = i != 0,
      quiet = FALSE
    )
  })
}

```

```

    i <- i + 1
  }, silent = TRUE)
}

message("All chunks written to ", gpkg_path, " in layer ", layer_name)

Gravji_all<-st_read("./Geodata/2024/MKIS/temp_MKIS_2025.gpkg",
                      layer="temp_Gravji")
Gravji_all2 = Gravji_all[!st_is_empty(Gravji_all),,drop=FALSE] # 0
table(st_is_valid(Gravji_all2))

write_sf(Gravji_all2,
         "./Geodata/2024/MKIS/MKIS_2025.gpkg",
         layer="Gravji",
         append=FALSE)
rm(list=ls())


# hydrometric posts ----

link="https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
url=parse_url(link)

url$query <- list(service = "wfs",request = "GetCapabilities")
request <- build_url(url)
bwk_client <- WFSClient$new(link,serviceVersion = "2.0.0")

bwk_client$getFeatureTypes(pretty = TRUE)

# geoms

url$query <- list(service = "wfs",
                    request = "GetFeature",
                    srsName="EPSG:3059",
                    typename = "zmni:zmni_hydropost",
                    count=100)
request <- build_url(url)

geometrijam <- read_sf(request)
geometrijam


# download
base_url <- "https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
type_name <- "zmni:zmni_hydropost"
crs_code <- 3059
chunk_size <- 100000
gpkg_path <- "./IevadesDati/MKIS/temp_MKIS_2025.gpkg"
layer_name <- "temp_HidrometriskiePosteni"
i <- 0

repeat {
  message("Fetching features ", i * chunk_size + 1, " to ", (i + 1) * chunk_size, "...")

  query <- list(
    service = "WFS",
    version = "2.0.0",
    request = "GetFeature",
    typename = type_name,
    srsName = paste0("EPSG:", crs_code),
    count = chunk_size,
    
```

```

        startIndex = i * chunk_size
    )

req_url <- modify_url(base_url, query = query)

try({
  chunk <- read_sf(req_url)
  if (nrow(chunk) == 0) break

  # Set CRS and cast to MULTILINESTRING, POINT, MULTIPOLYGON
  chunk <- chunk %>%
    st_set_crs(st_crs(crs_code)) %>%
    st_cast("POINT")

  # Write chunk to GeoPackage (append mode after first)
  st_write(
    chunk,
    dsn = gpkg_path,
    layer = layer_name,
    append = i != 0,
    quiet = FALSE
  )

  i <- i + 1
}, silent = TRUE)
}

message("All chunks written to ", gpkg_path, " in layer ", layer_name)

HidrometriskiePosteni_all=st_read("./Geodata/2024/MKIS/temp_MKIS_2025.gpkg",
                                    layer="temp_HidrometriskiePosteni")
HidrometriskiePosteni_all2 =
  ↵ HidrometriskiePosteni_all[!st_is_empty(HidrometriskiePosteni_all),,drop=FALSE] # 0
table(st_is_valid(HidrometriskiePosteni_all2))

write_sf(HidrometriskiePosteni_all2,
        "./Geodata/2024/MKIS/MKIS_2025.gpkg",
        layer="HidrometriskiePosteni",
        append=FALSE)
rm(list=ls())

# large diameter drain collectors ----

link="https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
url=parse_url(link)

url$query <- list(service = "wfs",request = "GetCapabilities")
request <- build_url(url)
bwk_client <- WFSClient$new(link,serviceVersion = "2.0.0")

bwk_client$getFeatureTypes(pretty = TRUE)

# geoms

url$query <- list(service = "wfs",
                    request = "GetFeature",
                    srsName="EPSG:3059",
                    typename = "zmni:zmni_bigdraincollectors",
                    count=100)
request <- build_url(url)

geometrijam <- read_sf(request)
geometrijam

```

```

# download
base_url <- "https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
type_name <- "zmni:zmni_bigdraincollectors"
crs_code <- 3059
chunk_size <- 100000
gpkg_path <- "./Geodata/2024/MKIS/temp_MKIS_2025.gpkg"
layer_name <- "temp_LielaDiametraKolektori"
i <- 0

repeat {
  message("Fetching features ", i * chunk_size + 1, " to ", (i + 1) * chunk_size, "...")

  query <- list(
    service = "WFS",
    version = "2.0.0",
    request = "GetFeature",
    typename = type_name,
    srsName = paste0("EPSG:", crs_code),
    count = chunk_size,
    startIndex = i * chunk_size
  )

  req_url <- modify_url(base_url, query = query)

  try({
    chunk <- read_sf(req_url)
    if (nrow(chunk) == 0) break

    # Set CRS and cast to MULTILINESTRING, POINT, MULTIPOLYGON
    chunk <- chunk %>%
      st_set_crs(st_crs(crs_code)) %>%
      st_cast("MULTILINESTRING")

    # Write chunk to GeoPackage (append mode after first)
    st_write(
      chunk,
      dsn = gpkg_path,
      layer = layer_name,
      append = i != 0,
      quiet = FALSE
    )

    i <- i + 1
  }, silent = TRUE)
}

message("All chunks written to ", gpkg_path, " in layer ", layer_name)

LielaDiametraKolektori_all<-st_read("./Geodata/2024/MKIS/temp_MKIS_2025.gpkg",
                                      layer="temp_LielaDiametraKolektori")
LielaDiametraKolektori_all2 =
  ↵ LielaDiametraKolektori_all[!st_is_empty(LielaDiametraKolektori_all),,drop=FALSE] # 0
table(st_is_valid(LielaDiametraKolektori_all2))

write_sf(LielaDiametraKolektori_all2,
        "./Geodata/2024/MKIS/MKIS_2025.gpkg",
        layer="LielaDiametraKolektori",
        append=FALSE)
rm(list=ls())

```

```

# river packets ----

link="https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
url=parse_url(link)

url$query <- list(service = "wfs", request = "GetCapabilities")
request <- build_url(url)
bwk_client <- WFSClient$new(link, serviceVersion = "2.0.0")

bwk_client$getFeatureTypes(pretty = TRUE)

# geoms

url$query <- list(service = "wfs",
                     request = "GetFeature",
                     srsName="EPSG:3059",
                     typename = "zmni:zmni_stateriverspickets",
                     count=100)
request <- build_url(url)

geometrijam <- read_sf(request)
geometrijam

# download

base_url <- "https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
type_name <- "zmni:zmni_stateriverspickets"
crs_code <- 3059
chunk_size <- 100000
gpkg_path <- "./Geodata/2024/MKIS/temp_MKIS_2025.gpkg"
layer_name <- "temp_Piketi"
i <- 0

repeat {
  message("Fetching features ", i * chunk_size + 1, " to ", (i + 1) * chunk_size, "...")

  query <- list(
    service = "WFS",
    version = "2.0.0",
    request = "GetFeature",
    typename = type_name,
    srsName = paste0("EPSG:", crs_code),
    count = chunk_size,
    startIndex = i * chunk_size
  )

  req_url <- modify_url(base_url, query = query)

  try({
    chunk <- read_sf(req_url)
    if (nrow(chunk) == 0) break

    # Set CRS and cast to MULTILINESTRING, POINT, MULTIPOLYGON
    chunk <- chunk %>%
      st_set_crs(st_crs(crs_code)) %>%
      st_cast("POINT")

    # Write chunk to GeoPackage (append mode after first)
    st_write(
      chunk,
      dsn = gpkg_path,
      layer = layer_name,

```

```

        append = i != 0,
        quiet = FALSE
    )

    i <- i + 1
}, silent = TRUE)
}

message("All chunks written to ", gpkg_path, " in layer ", layer_name)

Piketi_all=st_read("./Geodata/2024/MKIS/temp_MKIS_2025.gpkg",
                    layer="temp_Piketi")
Piketi_all2 = Piketi_all[!st_is_empty(Piketi_all),,drop=FALSE] # 0
table(st_is_valid(Piketi_all2))

write_sf(Piketi_all2,
        "./Geodata/2024/MKIS/MKIS_2025.gpkg",
        layer="Piketi",
        append=FALSE)
rm(list=ls())

# polder pumping stations ----

link="https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
url=parse_url(link)

url$query <- list(service = "wfs",request = "GetCapabilities")
request <- build_url(url)
bwk_client <- WFSClient$new(link,serviceVersion = "2.0.0")

bwk_client$getFeatureTypes(pretty = TRUE)

# geoms

url$query <- list(service = "wfs",
                     request = "GetFeature",
                     srsName="EPSG:3059",
                     typename = "zmni:zmni_polderpumpingstation",
                     count=100)
request <- build_url(url)

geometrijam <- read_sf(request)
geometrijam

# download
base_url <- "https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
type_name <- "zmni:zmni_polderpumpingstation"
crs_code <- 3059
chunk_size <- 100000
gpkg_path <- "./Geodata/2024/MKIS/temp_MKIS_2025.gpkg"
layer_name <- "temp_PolderuSuknuStacijas"
i <- 0

repeat {
  message("Fetching features ", i * chunk_size + 1, " to ", (i + 1) * chunk_size, "....")

  query <- list(
    service = "WFS",
    version = "2.0.0",
    request = "GetFeature",
    typename = type_name,

```

```

    srsName = paste0("EPSG:", crs_code),
    count = chunk_size,
    startIndex = i * chunk_size
  )

req_url <- modify_url(base_url, query = query)

try({
  chunk <- read_sf(req_url)
  if (nrow(chunk) == 0) break

  # Set CRS and cast to MULTILINESTRING, POINT, MULTIPOLYGON
  chunk <- chunk %>%
    st_set_crs(st_crs(crs_code)) %>%
    st_cast("POINT")

  # Write chunk to GeoPackage (append mode after first)
  st_write(
    chunk,
    dsn = gpkg_path,
    layer = layer_name,
    append = i != 0,
    quiet = FALSE
  )

  i <- i + 1
}, silent = TRUE)
}

message("All chunks written to ", gpkg_path, " in layer ", layer_name)

PolderuSuknuStacijas_all=st_read("./Geodata/2024/MKIS/temp_MKIS_2025.gpkg",
                                   layer="temp_PolderuSuknuStacijas")
PolderuSuknuStacijas_all2 =
  PolderuSuknuStacijas_all[!st_is_empty(PolderuSuknuStacijas_all),,drop=FALSE] # 0
table(st_is_valid(PolderuSuknuStacijas_all2))

write_sf(PolderuSuknuStacijas_all2,
         "./Geodata/2024/MKIS/MKIS_2025.gpkg",
         layer="PolderuSuknuStacijas",
         append=FALSE)
rm(list=ls())


# polders ----

link="https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
url=parse_url(link)

url$query <- list(service = "wfs", request = "GetCapabilities")
request <- build_url(url)
bwk_client <- WFSClient$new(link, serviceVersion = "2.0.0")

bwk_client$getFeatureTypes(pretty = TRUE)

# geoms

url$query <- list(service = "wfs",
                    request = "GetFeature",
                    srsName="EPSG:3059",
                    typename = "zmni:zmni_polderterritory",
                    count=100)
request <- build_url(url)

```

```

geometrijam <- read_sf(request)
geometrijam

geometrijas=st_set_crs(geometrijam,st_crs(3059))

library(gdalUtilities)
ensure_multipolygons <- function(X) {
  tmp1 <- tempfile(fileext = ".gpkg")
  tmp2 <- tempfile(fileext = ".gpkg")
  st_write(X, tmp1)
  ogr2ogr(tmp1, tmp2, f = "GPKG", nlt = "MULTIPOLYGON")
  Y <- st_read(tmp2)
  st_sf(st_drop_geometry(X), geom = st_geometry(Y))
}
poligoni <- ensure_multipolygons(geometrijas)
PolderuTeritorijas_all2 = poligoni[!st_is_empty(poligoni),,drop=FALSE] # 0
table(st_is_valid(PolderuTeritorijas_all2))

# download
base_url <- "https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
type_name <- "zmni:zmni_polderterritory"
crs_code <- 3059
chunk_size <- 100000
gpkg_path <- "./Geodata/2024/MKIS/temp_MKIS_2025.gpkg"
layer_name <- "temp_PolderuTeritorijas"
i <- 0

repeat {
  message("Fetching features ", i * chunk_size + 1, " to ", (i + 1) * chunk_size, "...")

  query <- list(
    service = "WFS",
    version = "2.0.0",
    request = "GetFeature",
    typename = type_name,
    srsName = paste0("EPSG:", crs_code),
    count = chunk_size,
    startIndex = i * chunk_size
  )

  req_url <- modify_url(base_url, query = query)

  try({
    chunk <- read_sf(req_url)
    if (nrow(chunk) == 0) break

    # Set CRS and cast to MULTILINESTRING, POINT, MULTIPOLYGON
    chunk <- chunk %>%
      st_set_crs(st_crs(crs_code)) %>%
      st_cast("MULTIPOLYGON")

    # Write chunk to GeoPackage (append mode after first)
    st_write(
      chunk,
      dsn = gpkg_path,
      layer = layer_name,
      append = i != 0,
      quiet = FALSE
    )

    i <- i + 1
  }, silent = TRUE)
  Sys.sleep(0.5)
}

```

```

message("All chunks written to ", gpkg_path, " in layer ", layer_name)

PolderuTeritorijas_all<-st_read("./Geodata/2024/MKIS/temp_MKIS_2025.gpkg",
                                layer="temp_PolderuTeritorijas")
PolderuTeritorijas_all2 =
  ↵ PolderuTeritorijas_all[!st_is_empty(PolderuTeritorijas_all),,drop=FALSE] # 0
  table(st_is_valid(PolderuTeritorijas_all2))

write_sf(PolderuTeritorijas_all2,
         "./Geodata/2024/MKIS/MKIS_2025.gpkg",
         layer="PolderuTeritorijas",
         append=FALSE)
rm(list=ls())

# catchment basins ----

link="https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
url=parse_url(link)

url$query <- list(service = "wfs",request = "GetCapabilities")
request <- build_url(url)
bwk_client <- WFSClient$new(link,serviceVersion = "2.0.0")

bwk_client$getFeatureTypes(pretty = TRUE)

# geoms

url$query <- list(service = "wfs",
                    request = "GetFeature",
                    srsName="EPSG:3059",
                    typename = "zmni:zmni_catchment",
                    count=100)
request <- build_url(url)

geometrijam <- read_sf(request)
geometrijam

# download
base_url <- "https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
type_name <- "zmni:zmni_catchment"
crs_code <- 3059
chunk_size <- 100000
gpkg_path <- "./Geodata/2024/MKIS/temp_MKIS_2025.gpkg"
layer_name <- "temp_SatecesBaseini"
i <- 0

repeat {
  message("Fetching features ", i * chunk_size + 1, " to ", (i + 1) * chunk_size, "...")
  query <- list(
    service = "WFS",
    version = "2.0.0",
    request = "GetFeature",
    typename = type_name,
    srsName = paste0("EPSG:", crs_code),
    count = chunk_size,
    startIndex = i * chunk_size
  )
}

```

```

req_url <- modify_url(base_url, query = query)

try({
  chunk <- read_sf(req_url)
  if (nrow(chunk) == 0) break

  # Set CRS and cast to MULTILINESTRING, POINT, MULTIPOLYGON
  chunk <- chunk %>%
    st_set_crs(st_crs(crs_code))

  ensure_multipolygons <- function(X) {
    tmp1 <- tempfile(fileext = ".gpkg")
    tmp2 <- tempfile(fileext = ".gpkg")
    st_write(X, tmp1)
    ogr2ogr(tmp1, tmp2, f = "GPKG", nlt = "MULTIPOLYGON")
    Y <- st_read(tmp2)
    st_sf(st_drop_geometry(X), geom = st_geometry(Y))
  }
  chunk <- ensure_multipolygons(chunk)

  # Write chunk to GeoPackage (append mode after first)
  st_write(
    chunk,
    dsn = gpkg_path,
    layer = layer_name,
    append = i != 0,
    quiet = FALSE
  )

  i <- i + 1
}, silent = TRUE)
Sys.sleep(0.5)
}

message("All chunks written to ", gpkg_path, " in layer ", layer_name)

SatecesBaseini_all<-st_read("./Geodata/2024/MKIS/temp_MKIS_2025.gpkg",
                           layer="temp_SatecesBaseini")
SatecesBaseini_all2 = SatecesBaseini_all[!st_is_empty(SatecesBaseini_all),,drop=FALSE] # 0
table(st_is_valid(SatecesBaseini_all2))

SatecesBaseini_all3<-st_make_valid(SatecesBaseini_all2)
table(st_is_valid(SatecesBaseini_all3))
SatecesBaseini_all3

write_sf(SatecesBaseini_all3,
        "./Geodata/2024/MKIS/MKIS_2025.gpkg",
        layer="SatecesBaseini",
        append=FALSE)
rm(list=ls())

# drainage connection points ----

link="https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
url=parse_url(link)

url$query <- list(service = "wfs", request = "GetCapabilities")
request <- build_url(url)
bwk_client <- WFSClient$new(link, serviceVersion = "2.0.0")

bwk_client$getFeatureTypes(pretty = TRUE)

```

```

# geoms

url$query <- list(service = "wfs",
                   request = "GetFeature",
                   srsName="EPSG:3059",
                   typename = "zmni:zmni_connectionpoints",
                   count=100)
request <- build_url(url)

geometrijam <- read_sf(request)
geometrijam

# download
base_url <- "https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
type_name <- "zmni:zmni_connectionpoints"
crs_code <- 3059
chunk_size <- 100000
gpkg_path <- "./Geodata/2024/MKIS/temp_MKIS_2025.gpkg"
layer_name <- "temp_Savienojumi"
i <- 0

repeat {
  message("Fetching features ", i * chunk_size + 1, " to ", (i + 1) * chunk_size, "...")

  query <- list(
    service = "WFS",
    version = "2.0.0",
    request = "GetFeature",
    typename = type_name,
    srsName = paste0("EPSG:", crs_code),
    count = chunk_size,
    startIndex = i * chunk_size
  )

  req_url <- modify_url(base_url, query = query)

  try({
    chunk <- read_sf(req_url)
    if (nrow(chunk) == 0) break

    # Set CRS and cast to MULTILINESTRING, POINT, MULTIPOLYGON
    chunk <- chunk %>%
      st_set_crs(st_crs(crs_code))

    chunk=st_cast(chunk,"POINT")

    # Write chunk to GeoPackage (append mode after first)
    st_write(
      chunk,
      dsn = gpkg_path,
      layer = layer_name,
      append = i != 0,
      quiet = FALSE
    )

    i <- i + 1
  }, silent = TRUE)
  Sys.sleep(0.5)
}

message("All chunks written to ", gpkg_path, " in layer ", layer_name)

Savienojumi_all=st_read("./Geodata/2024/MKIS/temp_MKIS_2025.gpkg",

```

```

            layer="temp_Savienojumi")
Savienojumi_all2 = Savienojumi_all[!st_is_empty(Savienojumi_all),,drop=FALSE] # 0
table(st_is_valid(Savienojumi_all2))

write_sf(Savienojumi_all2,
        "./Geodata/2024/MKIS/MKIS_2025.gpkg",
        layer="Savienojumi",
        append=FALSE)
rm(list=ls())

# state controlled rivers ----

link="https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
url=parse_url(link)

url$query <- list(service = "wfs",request = "GetCapabilities")
request <- build_url(url)
bwk_client <- WFSClient$new(link,serviceVersion = "2.0.0")

bwk_client$getFeatureTypes(pretty = TRUE)

# geoms

url$query <- list(service = "wfs",
                    request = "GetFeature",
                    srsName="EPSG:3059",
                    typename = "zmni:zmni_statecontrolleddrivers",
                    count=100)
request <- build_url(url)

geometrijam <- read_sf(request)
geometrijam

# download
base_url <- "https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
type_name <- "zmni:zmni_statecontrolleddrivers"
crs_code <- 3059
chunk_size <- 100000
gpkg_path <- "./Geodata/2024/MKIS/temp_MKIS_2025.gpkg"
layer_name <- "temp_ValstsNozimesUdensnotekas"
i <- 0

repeat {
  message("Fetching features ", i * chunk_size + 1, " to ", (i + 1) * chunk_size, "...")

  query <- list(
    service = "WFS",
    version = "2.0.0",
    request = "GetFeature",
    typename = type_name,
    srsName = paste0("EPSG:", crs_code),
    count = chunk_size,
    startIndex = i * chunk_size
  )

  req_url <- modify_url(base_url, query = query)

  try({
    chunk <- read_sf(req_url)
    if (nrow(chunk) == 0) break
  })
}

```

```

# Set CRS and cast to MULTILINESTRING, POINT, MULTIPOLYGON
chunk <- chunk %>%
  st_set_crs(st_crs(crs_code))

chunk=st_cast(chunk,"MULTILINESTRING")

# Write chunk to GeoPackage (append mode after first)
st_write(
  chunk,
  dsn = gpkg_path,
  layer = layer_name,
  append = i != 0,
  quiet = FALSE
)

i <- i + 1
}, silent = TRUE)
Sys.sleep(0.5)
}

message("All chunks written to ", gpkg_path, " in layer ", layer_name)

ValstsNozimesUdensnotekas_all=st_read("./Geodata/2024/MKIS/temp_MKIS_2025.gpkg",
                                       layer="temp_ValstsNozimesUdensnotekas")
ValstsNozimesUdensnotekas_all2 =
  ValstsNozimesUdensnotekas_all[!st_is_empty(ValstsNozimesUdensnotekas_all),,drop=FALSE] #
  0
table(st_is_valid(ValstsNozimesUdensnotekas_all2))

write_sf(ValstsNozimesUdensnotekas_all2,
         "./Geodata/2024/MKIS/MKIS_2025.gpkg",
         layer="ValstsNozimesUdensnotekas",
         append=FALSE)
rm(list=ls())

# zmni regions ----

link="https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
url=parse_url(link)

url$query <- list(service = "wfs", request = "GetCapabilities")
request <- build_url(url)
bwk_client <- WFSClient$new(link,serviceVersion = "2.0.0")

bwk_client$getFeatureTypes(pretty = TRUE)

# geoms

url$query <- list(service = "wfs",
                    request = "GetFeature",
                    srsName="EPSG:3059",
                    typename = "zmni:zmni_zmniregion",
                    count=100)
request <- build_url(url)

geometrijam <- read_sf(request)
geometrijam

library(gdalUtilities)

```

```

# download
base_url <- "https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
type_name <- "zmni:zmni_zmniregion"
crs_code <- 3059
chunk_size <- 100000
gpkg_path <- "./Geodata/2024/MKIS/temp_MKIS_2025.gpkg"
layer_name <- "temp_ZMNIRegions"
i <- 0

repeat {
  message("Fetching features ", i * chunk_size + 1, " to ", (i + 1) * chunk_size, "...")

  query <- list(
    service = "WFS",
    version = "2.0.0",
    request = "GetFeature",
    typename = type_name,
    srsName = paste0("EPSG:", crs_code),
    count = chunk_size,
    startIndex = i * chunk_size
  )

  req_url <- modify_url(base_url, query = query)

  try({
    chunk <- read_sf(req_url)
    if (nrow(chunk) == 0) break

    # Set CRS and cast to MULTILINESTRING, POINT, MULTIPOLYGON
    chunk <- chunk %>%
      st_set_crs(st_crs(crs_code))

    ensure_multipolygons <- function(X) {
      tmp1 <- tempfile(fileext = ".gpkg")
      tmp2 <- tempfile(fileext = ".gpkg")
      st_write(X, tmp1)
      ogr2ogr(tmp1, tmp2, f = "GPKG", nlt = "MULTIPOLYGON")
      Y <- st_read(tmp2)
      st_sf(st_drop_geometry(X), geom = st_geometry(Y))
    }
    chunk <- ensure_multipolygons(chunk)

    # Write chunk to GeoPackage (append mode after first)
    st_write(
      chunk,
      dsn = gpkg_path,
      layer = layer_name,
      append = i != 0,
      quiet = FALSE
    )

    i <- i + 1
  }, silent = TRUE)
  Sys.sleep(0.5)
}

message("All chunks written to ", gpkg_path, " in layer ", layer_name)

ZMNIRegions_all<-st_read("./Geodata/2024/MKIS/temp_MKIS_2025.gpkg",
                         layer="temp_ZMNIRegions")
ZMNIRegions_all2 = ZMNIRegions_all[!st_is_empty(ZMNIRegions_all)],,drop=FALSE] # 0
table(st_is_valid(ZMNIRegions_all2))

```

```

write_sf(ZMNIRegions_all2,
        "./Geodata/2024/MKIS/MKIS_2025.gpkg",
        layer="ZMNIRegions",
        append=FALSE)
rm(list=ls())


# water drainage ditches ----

link="https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
url=parse_url(link)

url$query <- list(service = "wfs", request = "GetCapabilities")
request <- build_url(url)
bwk_client <- WFSClient$new(link, serviceVersion = "2.0.0")

bwk_client$getFeatureTypes(pretty = TRUE)

# geoms

url$query <- list(service = "wfs",
                    request = "GetFeature",
                    srsName="EPSG:3059",
                    typename = "zmni:zmni_waterdrainditches",
                    count=100)
request <- build_url(url)

geometrijam <- read_sf(request)
geometrijam


# download
base_url <- "https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
type_name <- "zmni:zmni_waterdrainditches"
crs_code <- 3059
chunk_size <- 100000
gpkg_path <- "./Geodata/2024/MKIS/temp_MKIS_2025.gpkg"
layer_name <- "temp_UdensnotekasNovadgravji"
i <- 0

repeat {
  message("Fetching features ", i * chunk_size + 1, " to ", (i + 1) * chunk_size, "...")

  query <- list(
    service = "WFS",
    version = "2.0.0",
    request = "GetFeature",
    typename = type_name,
    srsName = paste0("EPSG:", crs_code),
    count = chunk_size,
    startIndex = i * chunk_size
  )

  req_url <- modify_url(base_url, query = query)

  try({
    chunk <- read_sf(req_url)
    if (nrow(chunk) == 0) break

    # Set CRS and cast to MULTILINESTRING, POINT, MULTIPOLYGON
    chunk <- chunk %>%
      st_set_crs(st_crs(crs_code))
  })
}

```

```

chunk=st_cast(chunk,"MULTILINESTRING")

# Write chunk to GeoPackage (append mode after first)
st_write(
  chunk,
  dsn = gpkg_path,
  layer = layer_name,
  append = i != 0,
  quiet = FALSE
)

i <- i + 1
}, silent = TRUE)
Sys.sleep(0.5)
}

message("All chunks written to ", gpkg_path, " in layer ", layer_name)

UdensnotekasNovadgravji_all=st_read("./Geodata/2024/MKIS/temp_MKIS_2025.gpkg",
  layer="temp_UdensnotekasNovadgravji")
UdensnotekasNovadgravji_all2 =
  ↵ UdensnotekasNovadgravji_all[!st_is_empty(UdensnotekasNovadgravji_all),,drop=FALSE] # 0
table(st_is_valid(UdensnotekasNovadgravji_all2))

write_sf(UdensnotekasNovadgravji_all2,
  "./Geodata/2024/MKIS/MKIS_2025.gpkg",
  layer="UdensnotekasNovadgravji",
  append=FALSE)
rm(list=ls())


# ditch pickets ----

link="https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
url=parse_url(link)

url$query <- list(service = "wfs",request = "GetCapabilities")
request <- build_url(url)
bwk_client <- WFSClient$new(link,serviceVersion = "2.0.0")

bwk_client$getFeatureTypes(pretty = TRUE)

# geoms

url$query <- list(service = "wfs",
  request = "GetFeature",
  srsName="EPSG:3059",
  typename = "zmni:zmni_ditchpicket",
  count=100)
request <- build_url(url)

geometrijam <- read_sf(request)
geometrijam


# download
base_url <- "https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
type_name <- "zmni:zmni_ditchpicket"

```

```

crs_code <- 3059
chunk_size <- 100000
gpkg_path <- "./Geodata/2024/MKIS/temp_MKIS_2025.gpkg"
layer_name <- "temp_UdensnotekuNovadgravjuPiketi"
i <- 0

repeat {
  message("Fetching features ", i * chunk_size + 1, " to ", (i + 1) * chunk_size, "...")

  query <- list(
    service = "WFS",
    version = "2.0.0",
    request = "GetFeature",
    typename = type_name,
    srsName = paste0("EPSG:", crs_code),
    count = chunk_size,
    startIndex = i * chunk_size
  )

  req_url <- modify_url(base_url, query = query)

  try({
    chunk <- read_sf(req_url)
    if (nrow(chunk) == 0) break

    # Set CRS and cast to MULTILINESTRING, POINT, MULTIPOLYGON
    chunk <- chunk %>%
      st_set_crs(st_crs(crs_code)) %>%
      st_cast("POINT")

    # Write chunk to GeoPackage (append mode after first)
    st_write(
      chunk,
      dsn = gpkg_path,
      layer = layer_name,
      append = i != 0,
      quiet = FALSE
    )

    i <- i + 1
  }, silent = TRUE)
}

message("All chunks written to ", gpkg_path, " in layer ", layer_name)

UdensnotekuNovadgravjuPiketi_all<-st_read("./Geodata/2024/MKIS/temp_MKIS_2025.gpkg",
                                             layer="temp_UdensnotekuNovadgravjuPiketi")
UdensnotekuNovadgravjuPiketi_all2 = UdensnotekuNovadgravjuPiketi_all[!st_is_empty]
  ↵ (UdensnotekuNovadgravjuPiketi_all),,drop=FALSE] # 0
table(st_is_valid(UdensnotekuNovadgravjuPiketi_all2))

write_sf(UdensnotekuNovadgravjuPiketi_all2,
         "./Geodata/2024/MKIS/MKIS_2025.gpkg",
         layer="UdensnotekuNovadgravjuPiketi",
         append=FALSE)
rm(list=ls())

# state river axis ----

link="https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
url=parse_url(link)

```

```

url$query <- list(service = "wfs", request = "GetCapabilities")
request <- build_url(url)
bwk_client <- WFSClient$new(link, serviceVersion = "2.0.0")

bwk_client$getFeatureTypes(pretty = TRUE)

# geoms

url$query <- list(service = "wfs",
  request = "GetFeature",
  srsName="EPSG:3059",
  typename = "zmni:zmni_stateriversline",
  count=100)
request <- build_url(url)

geometrijam <- read_sf(request)
geometrijam

# download
base_url <- "https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
type_name <- "zmni:zmni_stateriversline"
crs_code <- 3059
chunk_size <- 100000
gpkg_path <- "./Geodata/2024/MKIS/temp_MKIS_2025.gpkg"
layer_name <- "temp_UdenstecuAsis"
i <- 0

repeat {
  message("Fetching features ", i * chunk_size + 1, " to ", (i + 1) * chunk_size, "...")

  query <- list(
    service = "WFS",
    version = "2.0.0",
    request = "GetFeature",
    typename = type_name,
    srsName = paste0("EPSG:", crs_code),
    count = chunk_size,
    startIndex = i * chunk_size
  )

  req_url <- modify_url(base_url, query = query)

  try({
    chunk <- read_sf(req_url)
    if (nrow(chunk) == 0) break

    # Set CRS and cast to MULTILINESTRING, POINT, MULTIPOLYGON
    chunk <- chunk %>%
      st_set_crs(st_crs(crs_code))

    chunk=st_cast(chunk,"MULTILINESTRING")

    # Write chunk to GeoPackage (append mode after first)
    st_write(
      chunk,
      dsn = gpkg_path,
      layer = layer_name,
      append = i != 0,
      quiet = FALSE
    )
    i <- i + 1
  }, silent = TRUE)
  Sys.sleep(0.5)
}

```

```

}

message("All chunks written to ", gpkg_path, " in layer ", layer_name)

UdenstecuAsis_all<-st_read("./Geodata/2024/MKIS/temp_MKIS_2025.gpkg",
                           layer="temp_UdenstecuAsis")
UdenstecuAsis_all2 = UdenstecuAsis_all[!st_is_empty(UdenstecuAsis_all),,drop=FALSE] # 0
table(st_is_valid(UdenstecuAsis_all2))

write_sf(UdenstecuAsis_all2,
         "./Geodata/2024/MKIS/MKIS_2025.gpkg",
         layer="UdenstecuAsis",
         append=FALSE)
rm(list=ls())


# river surface ----

link="https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
url=parse_url(link)

url$query <- list(service = "wfs",request = "GetCapabilities")
request <- build_url(url)
bwk_client <- WFSClient$new(link,serviceVersion = "2.0.0")

bwk_client$getFeatureTypes(pretty = TRUE)

# geoms

url$query <- list(service = "wfs",
                    request = "GetFeature",
                    srsName="EPSG:3059",
                    typename = "zmni:zmni_stateriverspolygon",
                    count=100)
request <- build_url(url)

geometrijam <- read_sf(request)
geometrijam


# download
base_url <- "https://lvmgeoserver.lvm.lv/geoserver/zmni/ows?"
type_name <- "zmni:zmni_stateriverspolygon"
crs_code <- 3059
chunk_size <- 100000
gpkg_path <- "./Geodata/2024/MKIS/temp_MKIS_2025.gpkg"
layer_name <- "temp_UdenstecuVirsmasLaukumi"
i <- 0

repeat {
  message("Fetching features ", i * chunk_size + 1, " to ", (i + 1) * chunk_size, "...")
  query <- list(
    service = "WFS",
    version = "2.0.0",
    request = "GetFeature",
    typename = type_name,
    srsName = paste0("EPSG:", crs_code),
    count = chunk_size,
    startIndex = i * chunk_size
  )
}

```

```

req_url <- modify_url(base_url, query = query)

try({
  chunk <- read_sf(req_url)
  if (nrow(chunk) == 0) break

  # Set CRS and cast to MULTILINESTRING, POINT, MULTIPOLYGON
  chunk <- chunk %>%
    st_set_crs(st_crs(crs_code))

  ensure_multipolygons <- function(X) {
    tmp1 <- tempfile(fileext = ".gpkg")
    tmp2 <- tempfile(fileext = ".gpkg")
    st_write(X, tmp1)
    ogr2ogr(tmp1, tmp2, f = "GPKG", nlt = "MULTIPOLYGON")
    Y <- st_read(tmp2)
    st_sf(st_drop_geometry(X), geom = st_geometry(Y))
  }
  chunk <- ensure_multipolygons(chunk)

  # Write chunk to GeoPackage (append mode after first)
  st_write(
    chunk,
    dsn = gpkg_path,
    layer = layer_name,
    append = i != 0,
    quiet = FALSE
  )

  i <- i + 1
}, silent = TRUE)
Sys.sleep(0.5)
}

message("All chunks written to ", gpkg_path, " in layer ", layer_name)

UdenstecuVirsmasLaukumi_all<-st_read("./Geodata/2024/MKIS/temp_MKIS_2025.gpkg",
                                         layer="temp_UdenstecuVirsmasLaukumi")
UdenstecuVirsmasLaukumi_all2 =
  ↵ UdenstecuVirsmasLaukumi_all[!st_is_empty(UdenstecuVirsmasLaukumi_all),,drop=FALSE] # 0
table(st_is_valid(UdenstecuVirsmasLaukumi_all2))

UdenstecuVirsmasLaukumi_all3<-st_make_valid(UdenstecuVirsmasLaukumi_all2)
table(st_is_valid(UdenstecuVirsmasLaukumi_all3))

write_sf(UdenstecuVirsmasLaukumi_all3,
         "./Geodata/2024/MKIS/MKIS_2025.gpkg",
         layer="UdenstecuVirsmasLaukumi",
         append=FALSE)
rm(list=ls())

```

## 4.4 Topographic Map

To support research process at the University of Latvia, the third (completed by January 1, 2018) and fourth (unfinished) versions of the Latvian Geospatial Information Agency's topographic map M:10000 vector geodatabase were received. The most recent version is available for [public viewing](#), but access to the vector data is restricted.

For the purposes of this project, the ESRI geodatabase has been converted to a GeoPackage file. As part of the file format change, geometries (empty, their validity checked and corrected where necessary) and coordinate system have been checked.

Files were stored at Geodata/2024/TopographicMap/.

After processing each database separately, we combined the layers used in this project, selecting the most recent layer per map sheet. These layers are:

- bridge\_L, describing bridges as lines;
- bridge\_P, describing bridges as points;
- hidro\_A, describing waterbodies as polygons;
- hidro\_L, describing ditches and small rivers as lines;
- landus\_A, describing LULC as polygons;
- road\_A, describing larger roads as polygons;
- road\_L, including very small or disused ones, as lines;
- swamp\_A, describing bogs as polygons;
- flora\_L, describing linear tree and shrub formations;
- build\_A, describing types of built-up areas. Version 4 available at the University of Latvia does not include all the classes present in Version 3, therefore version 3 is used.

```
# libs ----
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(openxlsx)) {install.packages("openxlsx"); require(openxlsx)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# v4 ----
slani_v4=st_layers("./Geodata/2024/TopographicMap/Latvija_LKS92_v4_20250703.gdb/")
write.xlsx(slani_v4, "./Geodata/2024/TopographicMap/slani_v4partial.xlsx")

slani_v4$geometrijai=as.character(slani_v4$geomtype)
table(slani_v4$geometrijai)

slani_v4$geometrijai2=ifelse(slani_v4$geometrijai=="3D Point", "POINT",
                               ifelse(slani_v4$geometrijai=="Multi Polygon",
                                      "MULTIPOLYGON",
                                      ifelse(slani_v4$geometrijai=="3D Multi Line String",
                                             "MULTILINESTRING",
                                             ifelse(slani_v4$geometrijai=="3D Multi Polygon",
                                                   "MULTIPOLYGON", NA)))
                               )

slani4x=data.frame(name=slani_v4$name,
                    geometrija=slani_v4$geometrijai2)

ciklam4x=levels(factor(slani4x$name))
for(i in seq_along(ciklam4x)){
  print(i)
  sakums=Sys.time()
  nosaukums=ciklam4x[i]
  objekts=slani4x %>%
    filter(name==nosaukums)
  print(nosaukums)
  slanis=read_sf("./Geodata/2024/TopographicMap/topo10v4/Latvija_LKS92_v4_20250703.gdb/",
                 layer=nosaukums)
  slanisZM=st_zm(slanis)
  slanis2=st_cast(slanisZM,to=objekts$geometrija)
  write_sf(slanis2, "./Geodata/2024/TopographicMap/LGIAtopo10K_v4partial.gpkg",
          layer=nosaukums,
          append=FALSE)
  ilgums=Sys.time()-sakums
  print(ilgums)
}
```

```

# v3 ----
slani_v3=st_layers("./Geodata/2024/TopographicMap/Latvija_LKS92_v3_pilnais.gdb/")
write.xlsx(slani_v3,"./Geodata/2024/TopographicMap/slani_v3.xlsx")

slani_v3$geometrijai=as.character(slani_v3$geomtype)
table(slani_v3$geometrijai)

slani_v3$geometrijai2=ifelse(slani_v3$geometrijai=="3D Point","POINT",
                             ifelse(slani_v3$geometrijai=="Multi Polygon",
                                    "MULTIPOLYGON",
                                    ifelse(slani_v3$geometrijai=="3D Multi Line String",
                                           "MULTILINESTRING",
                                           ifelse(slani_v3$geometrijai=="3D Multi Polygon",
                                                 "MULTIPOLYGON",
                                                 ifelse(slani_v3$geometrijai=="Point","POINT",
                                                       ifelse(slani_v3$geometrijai=="Multi
                                                       ↳ Line String",
                                                       "MULTILINESTRING",
                                                       ifelse(slani_v3$geometrijai=="3D
                                                       ↳ Measured Point",
                                                       "POINT",
                                                       NA))))))

slani3x=data.frame(name=slani_v3$name,
                    geometrija=slani_v3$geometrijai2)

ciklam3x=levels(factor(slani3x$name))
for(i in seq_along(ciklam3x)){
  print(i)
  sakums=Sys.time()
  nosaukums=ciklam3x[i]
  objekts=slani3x %>%
    filter(name==nosaukums)
  print(nosaukums)
  slanis=read_sf("./Geodata/2024/TopographicMap/Latvija_LKS92_v3_pilnais.gdb/",
                 layer=nosaukums)
  slanisZM=st_zm(slanis)
  slanis2=st_cast(slanisZM,to=objekts$geometrija)
  write_sf(slanis2,"./Geodata/2024/TopographicMap/LGIAtopo10K_v3.gpkg",
           layer=nosaukums,
           append=FALSE)
  ilgums=Sys.time()-sakums
  print(ilgums)
}

# combination ----
st_layers("./Geodata/2024/TopographicMap/LGIAtopo10K_v3.gpkg")

pages4=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v4partial.gpkg",
               layer="Topo10_lapas")
pages4_united=st_union(pages4)
ggplot(pages4_united)+geom_sf()

# landus_A
landus_3=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v3.gpkg",
                  layer="landus_A")
landus_not4=st_difference(landus_3,pages4_united)
landus_not4=landus_not4 %>%
  dplyr::select(FNAME,FCODE)
landus_4=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v4partial.gpkg",
                  layer="landus_A")
landus_4=landus_4 %>%

```

```

dplyr::select(FNAME,FCODE)

landus_new=rbind(landus_not4,landus_4)
sfarrow::st_write_parquet(landus_new,"./Geodata/2024/TopographicMap/LandusA_COMB.parquet")

# bridge_L
data_3=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v3.gpkg",
               layer="bridge_L")
data_not4=st_difference(data_3,pages4_united)
data_not4=data_not4 %>%
  dplyr::select(FNAME,FCODE)
data_4=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v4partial.gpkg",
               layer="bridge_L")
data_4=data_4 %>%
  dplyr::select(FNAME,FCODE)

data_new=rbind(data_not4,data_4)
sfarrow::st_write_parquet(data_new,"./Geodata/2024/TopographicMap/BridgeL_COMB.parquet")

# bridge_P
data_3=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v3.gpkg",
               layer="bridge_P")
data_not4=st_difference(data_3,pages4_united)
data_not4=data_not4 %>%
  dplyr::select(FNAME,FCODE)
data_4=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v4partial.gpkg",
               layer="bridge_P")
data_4=data_4 %>%
  dplyr::select(FNAME,FCODE)

data_new=rbind(data_not4,data_4)
sfarrow::st_write_parquet(data_new,"./Geodata/2024/TopographicMap/BridgeP_COMB.parquet")

# hidro_A
data_3=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v3.gpkg",
               layer="hidro_A")
data_not4=st_difference(data_3,pages4_united)
data_not4=data_not4 %>%
  dplyr::select(FNAME,FCODE)
data_4=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v4partial.gpkg",
               layer="hidro_A")
data_4=data_4 %>%
  dplyr::select(FNAME,FCODE)

data_new=rbind(data_not4,data_4)
sfarrow::st_write_parquet(data_new,"./Geodata/2024/TopographicMap/HidroA_COMB.parquet")

# hidro_L
data_3=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v3.gpkg",
               layer="hidro_L")
data_not4=st_difference(data_3,pages4_united)
data_not4=data_not4 %>%
  dplyr::select(FNAME,FCODE)
data_4=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v4partial.gpkg",
               layer="hidro_L")
data_4=data_4 %>%
  dplyr::select(FNAME,FCODE)

data_new=rbind(data_not4,data_4)
sfarrow::st_write_parquet(data_new,"./Geodata/2024/TopographicMap/HidroL_COMB.parquet")

# road_A
data_3=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v3.gpkg",
               layer="road_A")
data_not4=st_difference(data_3,pages4_united)

```

```

data_not4=data_not4 %>%
  dplyr::select(FNAME,FCODE)
data_4=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v4partial.gpkg",
              layer="road_A")
data_4=data_4 %>%
  dplyr::select(FNAME,FCODE)

data_new=rbind(data_not4,data_4)
sfarrow::st_write_parquet(data_new,"./Geodata/2024/TopographicMap/RoadA_COMB.parquet")

# road_L
data_3=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v3.gpkg",
               layer="road_L")
data_not4=st_difference(data_3,pages4_united)
data_not4=data_not4 %>%
  dplyr::select(FNAME,FCODE)
data_4=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v4partial.gpkg",
               layer="road_L")
data_4=data_4 %>%
  dplyr::select(FNAME,FCODE)

data_new=rbind(data_not4,data_4)
sfarrow::st_write_parquet(data_new,"./Geodata/2024/TopographicMap/RoadL_COMB.parquet")

# swamp_A
data_3=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v3.gpkg",
               layer="swamp_A")
data_not4=st_difference(data_3,pages4_united)
data_not4=data_not4 %>%
  dplyr::select(FNAME,FCODE)
data_4=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v4partial.gpkg",
               layer="swamp_A")
data_4=data_4 %>%
  dplyr::select(FNAME,FCODE)

data_new=rbind(data_not4,data_4)
sfarrow::st_write_parquet(data_new,"./Geodata/2024/TopographicMap/SwampA_COMB.parquet")

# flora_L
data_3=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v3.gpkg",
               layer="flora_L")
data_not4=st_difference(data_3,pages4_united)
data_not4=data_not4 %>%
  dplyr::select(FNAME,FCODE)
data_4=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v4partial.gpkg",
               layer="flora_L")
data_4=data_4 %>%
  dplyr::select(FNAME,FCODE)

data_new=rbind(data_not4,data_4)
sfarrow::st_write_parquet(data_new,"./Geodata/2024/TopographicMap/FloraL_COMB.parquet")

# build_A
data_3=st_read("./Geodata/2024/TopographicMap/LGIAtopo10K_v3.gpkg",
               layer="build_A")
data_3=data_3 %>%
  dplyr::select(FNAME,FCODE)
sfarrow::st_write_parquet(data_3,"./Geodata/2024/TopographicMap/BuildA_v3.parquet")

```

## 4.5 Corine Land Cover 2018

Corine Land Cover is a publicly available geodata that characterizes land cover and land use (LULC) across Europe over a long period of time using a generally consistent (comparable) [methodology](#), providing results for [individual years](#) - 1990, 2000, 2006, 2012, 2018. Although the dataset has a coarse resolution – the mapping unit is 25 ha areas that are at least 100 m wide – it provides sufficient information for general use, such as workflow testing and observation filtering. This project uses data from 2018.

The downloaded data set has been transformed into the Latvian coordinate system (EPSG:3059), and the file format has been changed to GeoParquet to facilitate and speed up further work. As part of the file format change, geometries (empty, valid) have been checked.

Data are stored at Geodata/2024/CLC/.

```
# libs ----
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(arrow)) {install.packages("arrow"); require(arrow)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}

# downloaded data
clcLV=st_read("./Geodata/2024/CLC/clcLV.gpkg", layer="clcLV")

# empty geoms
clcLV2 = clcLV[!st_is_empty(clcLV), , drop=FALSE] # OK

# validation
validity=st_is_valid(clcLV2)
table(validity) # 3 non-valid
clcLV3=st_make_valid(clcLV2)

# crs
clcLV3=st_transform(clcLV3, crs=3059)

# saving
sfarrow::st_write_parquet(clcLV3, "./Geodata/2024/CLC/CLC_LV_2018.parquet")
```

## 4.6 Publicly available LVM data

[Latvian State Forests geospatial data on forest infrastructure and its description](#). The following datasets were used in the project:

- roads:
  - forest roads;
  - forest roads to be developed;
  - turning areas;
  - changeover areas;
  - driveways;
- drainage systems:
  - ditches;
  - drainage systems;
  - renovated drainage facilities.

Initially, no additional processing of this data was performed. It was used to prepare [geodata products](#) (more specifically, [Landscape classification](#)).

Data were downloaded to Geodata/2024/LVM\_OpenData

## 4.7 Soil data

Directory Geodata/2024/Soils/ contains various soil related datasets that need to be combined (soil texture) or can be used individually (soil chemistry). These datasets and their location in the file tree are documented in following subchapters.

### 4.7.1 Soil chemistry

Data on soil chemistry are obtained from [European Soil Data Centre's European Soil database](#) (Panagos et al., 2022). Dataset describing soil chemistry is derived from [LUCAS 2009/2012 topsoil data](#). There are several chemical properties available for download, however not all of them were chosen by experts for SDM:

- “P”: used;
- “N”: used;
- “K”: used;
- “CEC”: not used;
- “CN”: used;
- “pH\_CaCl”: not used;
- “ph\_H2O\_ration\_ph\_CaCl”: not used;
- “pH\_H2O”: used;
- “CaCO3”: used.

Files were downloaded to Geodata/2024/Soils/ESDAC/chemistry/ and no preprocessing was carried out.

### 4.7.2 Soil texture: Europe

Data on soil texture were obtained from [European Soil Data Centre's European Soil database](#) (Panagos et al., 2022). Dataset is available as [European Soil Database v2 Raster Library 1kmx1km](#). There are several properties available for download, TXT was used to create [soil texture product](#). Files were downloaded to Geodata/2024/Soils/ESDAC/texture/.

During the preprocessing (see code below) the layer was projected to match the 10 m template with “near” as interpolation method, value 0 substituted with NA and the result was masked and cropped to the template. Result was saved for further processing.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Template ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")

# ESDAC texture ----

sdTEXT=rast(paste0("./Geodata/2024/Soils/ESDAC/texture/SoilDatabaseV2_raster/",
                     "ESDB-Raster-Library-1k-GeoTIFF-20240507/TEXT/TEXT.tif"))

plot(sdTEXT)

sdTEXT=project(sdTEXT,template10,method="near")
plot(sdTEXT)

sdTEXT=subst(sdTEXT,0,NA)
```

```

plot(sdTEXT)

sdTEXT2=mask(sdTEXT,template10,
              filename="./RasterGrids_10m/2024/SoilTXT_ESDAC.tif",
              overwrite=TRUE)
plot(sdTEXT2)

```

#### 4.7.3 Soil texture: Farmland

Topsoil characteristics in Latvia were mapped in the mid-20th century, almost exclusively in farmlands. With time, data were digitised and combined with some other information resulting in artefacts. Therefore preprocessing was necessary. The version we used was obtained from the project “GOODWATER” C1D1\_Deliverable\_R2.

File is stored at Geodata/2024/Soils/TopSoil\_LV/.

Preprocessing included:

- reclassification based on the field GrSast:
  - sand (1): “mS”, “mSp”, “S”, “sS”, “iS”, “Gr”, “mGr”, “D”;
  - silt (2): “sM”, “sMp”, “sM2”, “sMp2”, “sM3”, “sMp3”;
  - clay (3): “M”, “M1”, “Mp”, “M2”, “sM1”, “sMp1”;
  - organic (4): “l”, “vd”, “vj”, “n”, “T”;
  - and other categories were left unclassified.
- coordinate transformation to EPSG:3059;
- investigation of the resulting layer looking for anomalies by scrolling in interactive GIS, which led to exclusion of land parcels larger than 200 ha.
- rasterisation to match the 10 m template with the highest class code prevailing.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Template ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")

# Farmland soil texture ----

augsnes=st_read("./Geodata/2024/Soils/TopSoil_LV/soil.gpkg",layer="soilunion")

# calculate parcels area
augsnes$platiba_ha=as.numeric(st_area(augsnes))/10000

# only parcels with existing information on texture
tuksas=augsnes %>%
  filter(GrSast=="") 

# classification
clay=c("M","M1","Mp","M2","sM1","sMp1")
silt=c("sM", "sMp", "sM2", "sMp2", "sM3", "sMp3")
sand=c("mS", "mSp", "S", "sS", "iS", "Gr", "mGr", "D")
peat=c("l", "vd", "vj", "n", "T")
augsnes=augsnes %>%
  mutate(grupas=case_when(GrSast %in% sand~"Sand",
                          GrSast %in% silt~"Silt",
                          GrSast %in% clay~"Clay",
                          GrSast %in% peat~"organika",

```

```

    .default=NA)) %>%
  mutate(grupas_num = case_when(GrSast %in% sand~"1",
                                GrSast %in% silt~"2",
                                GrSast %in% clay~"3",
                                GrSast %in% peat~"4",
                                .default=NA))

# crs
augsnes_3059 = st_transform(augsnes, crs=3059)

# only existing texture classification
augsnes_3059 = augsnes_3059 %>%
  filter(!is.na(grupas_num))

# parcels up to 200 ha
augsnes_3059small = augsnes_3059 %>%
  filter(!is.na(grupas_num)) %>%
  filter(platiba_ha < 200)

# rasterisation
viraugsnem2 = rasterize(augsnes_3059small, template10, field="grupas_num", fun="max",
                        filename="./RasterGrids_10m/2024/SoilTXT_topSoillLV.tif",
                        overwrite=TRUE)
plot(viraugsnem2)

```

#### 4.7.4 Soil texture: Quaternary

Data on Quaternary Geology are digitised and stored by the University of Latvia Department of Geology.

File is stored at `Geodata/2024/Soils/QuaternaryGeology_LV/`.

Preprocessing included:

- reclassification based on field `Litologija`:
  - sand (1): “smilts”, “smilts\_aleiritiska”, “smilts\_dunjaina”, “smilts\_grants”, “smilts\_grants\_oli”, “smilts\_grants\_oli\_aleirts”, “smilts\_kudraina”, “smilts\_videjgraudaina”, “malsmilts”, “smilts\_videjgraudaina”~“Sand”;
  - silt (2): “aleirits”, “aleirits\_malains”, “morena”, “smilts\_aleirits\_mals”, “smilts\_aleirits\_sapropelis”, “smilts\_malaina\_dazadgraudaina”, “malsmilts”;
  - clay (3): “mals”, “mals\_aleiritisks”;
  - organic (4): “dunjas”, “kudra”;
- coordinate transformation to EPSG:3059;
- rasterisation to match the 10 m template with the highest class code prevailing.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Template ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")

# Quaternary geology ----
kvartars=sfarrow::st_read_parquet(
  "./Geodata/2024/Soils/QuaternaryGeology_LV/Kvartargeologija.parquet")

# reclassification
kvartars=kvartars %>%

```

```

  mutate(grupas = case_when(Litologija=="aleirits"~"Silt",
                             Litologija=="aleirits_malains"~"Silt",
                             Litologija=="dunjas"~"organika",
                             Litologija=="kudra"~"organika",
                             Litologija=="mals"~"Clay",
                             Litologija=="mals_aleiritisks"~"Clay",
                             Litologija=="morena"~"Silt",
                             Litologija=="smilts"~"Sand",
                             Litologija=="smilts_aleiritiska"~"Sand",
                             Litologija=="smilts_aleirits_mals"~"Silt",
                             Litologija=="smilts_aleirits_sapropelis"~"Silt",
                             Litologija=="smilts_dunjaina"~"Sand",
                             Litologija=="smilts_grants"~"Sand",
                             Litologija=="smilts_grants_oli"~"Sand",
                             Litologija=="smilts_grants_oli_aleirits"~"Sand",
                             Litologija=="smilts_kudraina"~"Sand",
                             Litologija=="smilts_malaina_dazadraudaina, malsmilts"~"Silt",
                             Litologija=="smilts_videjgraudaina, malsmilts"~"Sand",
                             Litologija=="smilts_videjgraudaina"~"Sand",
                             .default=NA))

# numeric codes
kvartars=kvartars %>%
  mutate(grupas_num=case_when(grupas == "Sand" ~"1",
                               grupas == "Silt" ~"2",
                               grupas == "Clay" ~"3",
                               grupas == "organika" ~"4",
                               .default=NA))

# crs transformation
kvartars_3059=st_transform(kvartars,crs=3059)

# nonmissing classes
kvartars_3059=kvartars_3059 %>%
  filter(!is.na(grupas_num))

# rasterisation
apaksaugsnem=rasterize(kvartars_3059,template10,field="grupas_num",fun="max",
                        filename="./RasterGrids_10m/2024/SoilTXT_QuaternaryLV.tif",
                        overwrite=TRUE)
plot(apaksaugsnem)

```

#### 4.7.5 Organic soils: SILAVA

The distribution of organic soils was modelled under the EU LIFE Programme project “Demonstration of climate change mitigation potential of nutrients rich organic soils in Baltic States and Finland” at the scientific institue SILAVA. Results were downloaded and stored at Geodata/2024/Soils/OrganicSoils\_SILAVA/.

Even though the layer covers all of Latvia, it has visible inconsistencies, particularly stripes. These were digitised manually (as vector polygons) and masked out as a part of preprocessing.

For further soil texture analysis we saved a GeoTIFF file with only presences.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Template ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")

# Organic Soils SILAVA ----

organika_silava=rast("./Geodata/2024/Soils/OrganicSoils_SILAVA/Silava_OrgSoils.tif")
plot(organika_silava)

```

```

# visible stripes

# only 40+ cm deep
organika_silava=ifel(organika_silava==2,1,NA)
organika_silavaLV=project(organika_silava,template10)

# stripes drawn manually, rasterisation
silavas_telpai=st_read("./Geodata/2024/Soils/OrganicSoils_SILAVA/stripam.gpkg",
                        layer="stripam")
silavas_telpai=st_transform(silavas_telpai,crs=3059)
silavas_telpai$yes=1
SilavasTelpa_10=rasterize(silavas_telpai,template10,field="yes")

# presence-only layer without stripes
silava_BezStripam1=ifel(organika_silavaLV==1&SilavasTelpa_10==1,1,NA)
silava_BezStripam=mask(silava_BezStripam1,template10)
plot(silava_BezStripam)
writeRaster(silava_BezStripam,
            "./RasterGrids_10m/2024/SoilTXT_OrganicSilava.tif",
            overwrite=TRUE)

```

#### 4.7.6 Organic soils: LU

The distribution of organic soils in farmlands was modelled by the University of Latvia project “Improvement of sustainable soil resource management in agriculture: E2SOILAGRI”.

From all the results we used layer YN\_prognozes\_smooth.tif stored at Geodata/2024/Soils/OrganicSoils\_LU/.

Preprocessing consisted of projecting the layer to match the 10 m template. Both presences and absences were saved for further processing.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Template ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")

# Organic Soils LU ----

kudra_norvegi=rast("./Geodata/2024/Soils/OrganicSoils_LU/YN_prognozes_smooth.tif")
kudra_norvLV=project(kudra_norvegi,template10)
plot(kudra_norvLV)

writeRaster(kudra_norvLV,
            "./RasterGrids_10m/2024/SoilTXT_OrganicLU.tif",
            overwrite=TRUE)

```

## 4.8 Dynamic World data

Dynamic World (DW) is a relatively new Earth observation system product that classifies land cover and land use (LULC) into nine categories (0=water, 1=trees, 2=grass, 3=flooded\_vegetation, 4=crops, 5=shrub\_and\_scrub, 6=built, 7=bare, 8=snow\_and\_ice), for each ESA Copernicus Sentinel-2 image with identified cloudiness  $\leq 35$ , allowing for filtering and various aggregations (Brown et al., 2022).

DW input information - raster layer for each season in each year - was prepared on the Google Earth Engine (GEE) platform (Gorelick et al., 2017) using a [replication script](#). To use this script, you need a [GEE account and project](#) and sufficient space on Google Drive. When executing the command lines, a download will be

offered for a file covering the time period from the value in row 7 to the value in row 8 (the file name should be specified in row 32, its description in row 33 and the directory on Google Drive in row 31, or all of this can be specified by confirming the save). This script is not optimized for preparing all seasonal periods for all years, so in order to reproduce or expand this study, it is necessary to change it manually.

Downloaded files are to be stored at Geodata/2024/DynamicWorld/RAW/.

During download, it can be seen that each layer covering all of Latvia is divided into several sheets. This is because, in order to ensure a true zero class (class “water” rather than background), the layers are encoded as Float rather than integers. All of these tiles need to be downloaded, and the following R command lines combine them, ensuring that the coordinate system and pixels correspond to the reference raster.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# 10 m template ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")

# DW export no GEE ----
faili=data.frame(faili=list.files("./Geodata/2024/DynamicWorld/RAW/"))
faili$celi_sakums=paste0("./Geodata/2024/DynamicWorld/RAW/",faili$faili)

# prepping ----
faili=faili %>%
  separate(faili,into=c("DW","gads","periods","parejais"),sep="_",remove = FALSE) %>%
  mutate(unikalais=paste0(DW,"_",gads,"_",periods),
        mosaic_name=paste0(unikalais,".tif"),
        mosaic_cels=paste0("./Geodata/2024/DynamicWorld/",mosaic_name))

# every layer consists of two tiles
unikalie=levels(factor(faili$unikalais))
min(table(faili$unikalais))
max(table(faili$unikalais))

# job
for(i in seq_along(unikalie)){
  unikalais=faili %>% filter(unikalais==unikalie[i])
  beigu_cels=unique(unikalais$mosaic_cels)

  print(i)

  viens=rast(unikalais$celi_sakums[1])
  divi=rast(unikalais$celi_sakums[2])

  viens2=project(viens,template10)
  divi2=project(divi,template10)

  mozaika=mosaic(viens2,divi2,fun="first")
  maskets=mask(mozaika,template10,
               filename=beigu_cels,
               overwrite=TRUE)

  print(beigu_cels)
}
```

## 4.9 The Global Forest Watch

The Global Forest Watch (GFW) is a widely known product that describes tree canopy cover in 2000, its annual growth from 2001 to 2012, and its annual loss from 2001 to the current version, which is updated annually (Hansen et al., 2013). The data is available both on the [project website](#) and on [GEE](#), where it was developed. This project used v1.12, in which the last year of tree loss dating was 2024, preparing it for

download on the GEE platform with this [replication script](#). To use this script, you need a [GEE account and project](#) and sufficient space on Google Drive. When executing the command lines, you will be offered to download the file, which you need to save to Google Drive.

After executing the command lines and preparing the results in Google Drive, four files become available for download. The location to download them is Geodata/2024/Trees/GFW/Raw/. After download, these files need to be projected to match the reference raster.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}

# 10 m rastra template ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")

# TreeCoverLoss ----
treecoverloss=rast("./Geodata/2024/Trees/GFW/Raw/TreeCoverLoss_v1_12.tif")

tcl=ifel(treecoverloss<1,NA,treecoverloss)

tcl2=terra::project(tcl,paraugs)
tcl3=mask(tcl2,paraugs,filename="./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif",]
  ↵  overwrite=TRUE)
```

## 4.10 Palsar

The Palsar Forests resource is based on PALSAR-2 synthetic aperture radar (SAR) reflectance classification of forest and non-forest land with a pixel resolution of 25 m. Forests are classified as areas of at least 0.5 ha covered with trees, where tree cover (at least 5 m high) is at least 10% (Shimada et al., 2013). The data is available at [GEE](#). This project used a 4-class version (1=Dense Forest, 2=Non-dense Forest, 3=Non-Forest, 4=Water), in which the last tree cover dating year was 2020, prepared for download on the GEE platform with this [replication script](#). To use this script, you need a [GEE account and project](#) and sufficient space on Google Drive. When executing the command lines, you will be offered to download the file, which you need to save to Google Drive.

After executing the command lines and preparing the results in Google Drive, four files become available for download. The location to download them is Geodata/2024/Trees/Palsar/Raw/. After download, these files need to be projected to match the reference raster and merged. In this resource, trees are coded into two groups: 1=Dense Forest and 2=Non-dense Forest, which need to be merged and the rest converted to missing values (see code below).

Although this resource reflects conditions in 2020 rather than 2024, we used it because [The Global Forest Watch data](#) provides reliable data on canopy loss, but the appearance of canopy cover is not so rapid that there would be significant changes over a four-year period.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}

# 10 m rastra template ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")

# PALSAR Forests ----

fnf1=rast("./Geodata/2024/Trees/Palsar/Raw/ForestNonForest-0000023296-0000023296.tif")
fnf2=rast("./Geodata/2024/Trees/Palsar/Raw/ForestNonForest-0000023296-0000000000.tif")
fnf3=rast("./Geodata/2024/Trees/Palsar/Raw/ForestNonForest-0000000000-0000023296.tif")
fnf4=rast("./Geodata/2024/Trees/Palsar/Raw/ForestNonForest-0000000000-0000000000.tif")

fnf1p=terra::project(fnf1,template10)
fnf2p=terra::project(fnf2,template10)
fnf3p=terra::project(fnf3,template10)
```

```

fnf4p=terra::project(fnf4,template10)

fnfA=terra::merge(fnf1p,fnf2p)
fnfB=terra::merge(fnfA,fnf3p)
fnfC=terra::merge(fnfB,fnf4p)
plot(fnfC)

fnf_X=ifel(fnfC<=2&fnfC>=1,1,NA)
plot(fnf_X)

fnf_XX=mask(fnf_X,template10,
            filename=".~/Geodata/2024/Trees/Palsar/Palsar_Forests.tif",
            overwrite=TRUE)

```

## 4.11 CHELSA v2.1

Climatologies at high resolution for the Earth's land surface areas (CHELSA) is a 30 arc second global down-scaled climate data set (Karger et al., 2017). The temperature algorithm is based on statistical downscaling of atmospheric temperatures. The precipitation algorithm incorporates orographic predictors including wind fields, valley exposition, and boundary layer height, with a subsequent bias correction. CHELSA climatological data has a similar accuracy as other products for temperature, but its predictions of precipitation patterns are better (Karger et al., 2017). Data (1980-2010 baseline) are freely available for download from the [homepage](#) forwarding to download server, with download links for selected products available. There is also technical specification available. In this project we used version 2.1.

The download links we used together with the renaming scheme are [included](#) with this document. The following command lines perform download, crop to the extent of Latvia (using 1 km vector grid) and save the files for further processing described with other EGVs.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(curl)) {install.packages("curl"); require(curl)}

# templates ----
# 1km grid
tikls1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
telpai=tikls1km %>%
  mutate(yes=1) %>%
  summarise(yes=max(yes)) %>%
  st_buffer(.,dist=10000)

# download and crop ----

links_names=read_csv("./Geodata/2024/CHELSA/CHELSAdownload_rename.csv")
links_names=links_names %>%
  filter(todownload==1)

for(i in seq_along(links_names$localname)){
  print(i)
  sakums=Sys.time()
  links=links_names$weblocation[i]
  saving1="./Geodata/2024/CHELSA/draza.tif"
  saving2=paste0("./Geodata/2024/CHELSA/",links_names$localname[i])

  curl_download(url=links,destfile = saving1,quiet = FALSE)
  fails=rast(saving1)
  telpa=st_transform(telpai,crs=st_crs(fails))
  nogriezts=crop(fails,telpa,
                 filename=saving2,

```

```

        overwrite=TRUE)
unlink(saving1)
beigas=Sys.time()
ilgums=beigas-sakums
print(ilgums)
}

```

## 4.12 HydroClim data

HydroClim is a near-global freshwater-specific environmental variable dataset, created for biodiversity analysis at 1 km resolution (Domisch et al., 2015). Dataset contains many different variables along the HydroSHEDS river network (Lehner et al., 2008), including upstream climate recalculated from worldclim (Hijmans et al., 2005). We downloaded (to Geodata/2024/HydroClim/) averaged upstream climate from [Zenodo repository](#) (available also from [Dryad](#)) and cropped to the extent of Latvia and renamed files for further processing with the code below. Renaming scheme is [published with document](#).

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
tikls1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")

# reading HydroClim ----
videjie=terra::rast("./Geodata/2024/HydroClim/hydroclim_average+sum.nc")

# reading dictionary ----
slanu_nosaukumi=read_csv("./Geodata/2024/HydroClim/HydroClim_renaming.csv")

# cropping ---
tikls1km_reproj=st_transform(tikls1km,crs=st_crs(videjie))
telpai=tikls1km %>%
  mutate(yes=1) %>%
  summarise(yes=max(yes)) %>%
  st_buffer(.,dist=10000) %>%
  st_transform(.,crs=st_crs(videjie))
videjie=terra::crop(videjie,telpai)

# layer names ---
names(videjie)=slanu_nosaukumi$local_name

# saving files ---
for(i in seq_along(slanu_nosaukumi$local_name)){
  nosaukumam=slanu_nosaukumi$local_name[i]
  writeRaster(videjie[[i]],
              paste0("./Geodata/2024/HydroClim/",nosaukumam),
              overwrite=TRUE)
}

```

The raster dataset contains values only where large enough rivers are detected in HydroSHEDS. However, for species distribution modelling in this project we need continuously covered raster surfaces. For necessary geoprocessing to create such surfaces, we downloaded also HydroBASINS (Lehner and Grill, 2013) [dataset](#) to Geodata/2024/HydroClim/. These procedures were EGV-specific and are described with other [EGVs](#).

## 4.13 Sentinel-2 indices

The European Space Agency (ESA) Copernicus program's Sentinel-2 mission is a constellation of two (three since 09/05/2024) identical satellites orbiting in the same orbit. The first satellite, Sentinel-2A, entered its orbit and underwent calibration tests on 2015-06-23, the second (Sentinel-2B) on 2017-03-07, with the first images available earlier. Each satellite captures high-resolution images (from 10 m (at the equator) pixel resolution) in 13 spectral channels with a return time of up to 5 days (more frequently closer to the poles) ([https://www.esa.int/Applications/Observing\\_the\\_Earth/Copernicus/Sentinel-2](https://www.esa.int/Applications/Observing_the_Earth/Copernicus/Sentinel-2)). The data from this mission is freely available, including on the Google Earth Engine platform (Gorelick et al., 2017) for various large-scale pre-processing and analysis. We used the harmonized Level-2A ([https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS\\_S2\\_SR\\_HARMONIZED#description](https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_S2_SR_HARMONIZED#description)) product, applying a cloud mask that includes not only cloud filtering but also shadow filtering. For each filtered image (cloud-free, April-October, 2020-2024), we computed the normalized difference vegetation index (NDVI), the normalized difference moisture index (NDMI), and the normalized difference water index (NDWI) as well as various metrics. A [replication script](#) can be used to prepare the data. To use this script, you need a [GEE account and project](#) and sufficient space on Google Drive. When executing the command lines, the following files will be offered for download:

- NDVI\_median-ST-[runtag, 20250820 by default] - NDVI short-term median (2020-2024) of annual medians (April to October)
- NDVI\_p25-ST-[runtag, 20250820 by default] - NDVI short-term median (2020-2024) of annual 25th percentiles (April to October)
- NDVI\_p75-ST-[runtag, 20250820 by default] - NDVI short-term median (2020-2024) of annual 75th percentiles (April to October)
- NDVI\_iqr-ST-[runtag, 20250820 by default] - NDVI short-term median (2020-2024) of inter-quartile ranges (April to October)
- NDVI\_median-LY-[runtag, 20250820 by default] - NDVI last-years (2024) median (April to October)
- NDMI\_median-ST-[runtag, 20250820 by default] - NDMI short-term median (2020-2024) of annual medians (April to October)
- NDMI\_p25-ST-[runtag, 20250820 by default] - NDMI short-term median (2020-2024) of annual 25th percentiles (April to October)
- NDMI\_p75-ST-[runtag, 20250820 by default] - NDMI short-term median (2020-2024) of annual 75th percentiles (April to October)
- NDMI\_iqr-ST-[runtag, 20250820 by default] - NDMI short-term median (2020-2024) of inter-quartile ranges (April to October)
- NDMI\_median-LY-[runtag, 20250820 by default] - NDMI last-years (2024) median (April to October)
- NDWI\_median-ST-[runtag, 20250820 by default] - NDWI short-term median (2020-2024) of annual medians (April to October)
- NDWI\_p25-ST-[runtag, 20250820 by default] - NDWI short-term median (2020-2024) of annual 25th percentiles (April to October)
- NDWI\_p75-ST-[runtag, 20250820 by default] - NDWI short-term median (2020-2024) of annual 75th percentiles (April to October)
- NDWI\_iqr-ST-[runtag, 20250820 by default] - NDWI short-term median (2020-2024) of inter-quartile ranges (April to October)
- NDWI\_median-LY-[runtag, 20250820 by default] - NDWI last-years (2024) median (April to October)

After executing the command lines and preparing the results in Google Drive, it can be seen that each layer covering all of Latvia is divided into several tiles. This is because the layers are encoded as Float and exceed 4 GB in size before GeoTIFF compression. All of these files need to be downloaded and located at Geodata/2024/S2indices/Raw. The following R commands combine them, ensuring the coordinate systems and its naming, and pixels matching to the reference raster, while renaming files to E0\_[index]-[term: ST or LY][statistic].

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# 10 m raster template ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")

# Fails as exported from GEE ----
faili=data.frame(fails=list.files("./Geodata/2024/S2indices/RAW/",pattern = ".tif"))
faili$celi_sakums=paste0("./Geodata/2024/S2indices/RAW/",faili$fails)

# file names ----
faili=faili %>%
  separate(fails,into=c("nosaukums","vidus","beigas"),sep="-",remove = FALSE) %>%
  mutate(mosaic_name=paste0("E0_",nosaukums,"_",beigas,tolower(vidus),".tif"),
        mosaic_cels=paste0("./Geodata/2024/S2indices/Mosaics/",mosaic_name))

unikalie=levels(factor(faili$mosaic_name))
min(table(faili$mosaic_name))
max(table(faili$mosaic_name))

# preparation of mosaics ----
for(i in seq_along(unikalie)){
  sakums=Sys.time()
  unikalais=faili %>% filter(mosaic_name==unikalie[i])
  beigu_cels=unique(unikalais$mosaic_cels)

  print(i)

  # there are exactly 2 tiles per file
  viens=rast(unikalais$celi_sakums[1])
  divi=rast(unikalais$celi_sakums[2])

  viens2=terra::project(viens,template10)
  divi2=terra::project(divi,template10)

  mozaika=terra::merge(viens2,divi2)
  maskets=mask(mozaika,template10,
                filename=beigu_cels,overwrite=TRUE,
                gdal=c("COMPRESS=LZW","TILED=YES","BIGTIFF=IF_SAFER"),
                datatype="FLT4S",
                NAflag=NA)

  plot(maskets,main=unikalie[i])
  print(beigu_cels)
  beigas=Sys.time()
  ilgums=beigas-sakums
  print(ilgums)
}

}

```

## 4.14 Waste and garbage disposal sites, landfills

Information on landfills has been compiled from [The Ministry of Smart Administration and Regional Development](#) and Latvian Environment, Geology and Meteorology Centre's report, [“Report on landfills in Latvia in 2023”](#) listed landfills and their addresses. The coordinates required for the preparation of EGVs were obtained by combining the resources <https://www.google.com/maps> and <https://balticmaps.eu/>. In addition to the resources mentioned above, an object was added at the address “Dardedzes C, Mārupes pag., Latvia, LV-2166”.

In addition, information from the [State Environmental Service on separate waste and deposit packaging collection points](#) was used, exporting it to an Excel file.

Both data sets were combined into a single file and [added](#) to this material.

## 4.15 Digital elevation/terrain models

With the publication of continuous aerial laser scanning data for the territory of Latvia (<https://www.lgia.gov.lv/lv/digitalie-augstuma-modeli-0>), various high-resolution (1 m and higher) digital surface models (DSM) and digital elevation models (DEM) have been developed. Since the input data was the same in all cases, the values of these (corresponding) models were identical across almost the entire territory of the country. However, airborne laser scanning data (1) was not available for the entire territory of the country, and (2) there were differences between the models in terms of filling (availability of values) outside inland waters and (3) filling of water bodies themselves. However, for areas covered by data on land, the values were almost identical. Pearson's correlation coefficients between the DEMs developed by LU GZZF, LVMI Silava, and LGIA were greater than 0.999999.

The two DEMs (LU GZZF and LVMI Silava) were combined (arithmetic mean) within the University of Latvia project “Improvement of sustainable soil resource management in agriculture: E2SOILAGRI”, was used as the working DEM. The resolution of this DEM is 1 m, which is too detailed for species distribution modeling input data, therefore the layer was designed to correspond to the reference 10 m raster.

When investigating the combined DEM, there were clearly visible areas with no data. This has been solved by using the DEM with a resolution of 10 m developed by Māris Nartiš (LU GZZF) in 2018, which covers the entire territory of Latvia without gaps. To avoid sharp edges and ensure smooth transitions, we created an arithmetic mean layer covering all of Latvia and aligned to the reference raster.

A slope layer has also been created from this raster, which is designed in accordance with the reference. The slope is expressed in degrees and calculated using the 8-neighbor approach. The same applies to the aspect or slope direction.

```
# libs
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}

# reference
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")

# LiDAR DEM 1 m to 10 m

lapas_1m=data.frame(faili=list.files("./Geodata/2024/DEM/meanDEM_1mOLD/",pattern="*.tif$"))
lapas_1m$numurs=substr(lapas_1m$faili,10,13)
lapas_1m$cels1=paste0("./Geodata/2024/DEM/meanDEM_1mOLD/",lapas_1m$faili)
lapas_1m$cels2=paste0("./Geodata/2024/DEM/meanDEM_10mOLD/",lapas_1m$faili)

kvadrati=st_read(dsn="GIS_Latvija10.2.gdb",layer="tks93_50000")
kvadrati$name=as.character(kvadrati$num50tk)

moz2=rast("./Geodata/2024/DEM/Nartiss_visa_Latvija/dem10_20_kopa.tif")

for(i in 1:length(kvadrati$name)){
  kvadrats=kvadrati[i,]
  nosaukums=kvadrats$name
  telpa=terra::ext(kvadrats)

  paraugs=crop(template10,telpa)
  nart=crop(moz2,telpa)
  nart2=project(nart,paraugs,mask=TRUE)

  dem1m=lapas_1m[lapas_1m$numurs==kvadrats$name,]
  if(nrow(dem1m)>0){
    sakumcels=dem1m$cels1
    dem=rast(sakumcels)
```

```

reproj=project(dem,paraugs,mask=TRUE,method="bilinear",use_gdal=TRUE)
videjais <- ifel(is.na(nart2),nart2,ifel(is.na(reproj),nart2,
                                         app(c(nart2,reproj), mean)))
writeRaster(videjais,overwrite=TRUE,
            filename=paste0("./Geodata/2024/DEM/meanDEM_10m/","vidDEM_",
                           nosaukums,".tif"))
}

else{
  writeRaster(nart2,overwrite=TRUE,
              filename=paste0("./Geodata/2024/DEM/meanDEM_10m/","vidDEM_",
                             nosaukums,".tif"))
}

# vrt un mosaic
lapas_10=data.frame(faili=list.files("./Geodata/2024/DEM/meanDEM_10m/",pattern="*.tif$"))
lapas_10$celi1=paste0("./Geodata/2024/DEM/meanDEM_10m/",lapas_10$faili)
mozaikai=vrt(lapas_10$celi1,overwrite=TRUE,
              filename="./Geodata/2024/DEM/vrtDEM_10m.tif")
mozaika=rast("./Geodata/2024/DEM/vrtDEM_10m.tif")
writeRaster(mozaika,"./Geodata/2024/DEM/mozDEM_10m.tif")

## slope
reljefs=rast("./Geodata/2024/DEM/mozDEM_10m.tif")
slipumi=terrain(reljefs, v="slope", neighbors=8, unit="degrees",
                 filename="./Geodata/2024/DEM/Terrain_Slope_10m.tif", overwrite=TRUE)

## aspect
reljefs=rast("./Geodata/2024/DEM/mozDEM_10m.tif")
virzieni=terrain(reljefs, v="aspect", neighbors=8, unit="degrees",
                  filename="./Geodata/2024/DEM/Terrain_Aspect_10m.tif", overwrite=TRUE)

```

## 4.16 Latvian Exclusive Economic Zone polygon

The waters of Latvia's Exclusive Economic Zone were obtained from the [HELCOM map and data service](#). After downloading, this line file was analogically connected to the coastline file obtained from the same resource.

## 4.17 Bogs and Mires: EDI

Data (training and classification) used in project “Remote Sensing and Machine Learning for Peatland Habitat Monitoring (PurvEO)” by the Institute of electronics and computer science (EDI) were stored at Geodata/2024/Bogs\_EDI.

Preprocessing was carried out to create two layers:

- EDI\_BogsYN.tif: training and classification results on open raised bogs (EU protected habitat codes 7110 and 7120) and locations where one of those overlapped with transitional mires (EU protected habitat code 7140);
- EDI\_TransitionalMiresYN.tif: training and classification results on transitional mires (EU protected habitat code 7140) with no overlap with open rised bogs.

```

# libs
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(terra)) {install.packages("terra"); require(terra)}
```

```

# Templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

nulles10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")

# Bogs ----
neatklata71107120=rast(paste0("./Geodata/2024/Bogs_EDI/purvi_EDI_projekts/",
                                   "purvi/!LV_kopa_apv1020_30_05_2022/",
                                   "!LV_kopa_apv1020_30_05_2022/",
                                   "Neatklata_purviem_raksturiga_zemsedze_7110_7120.tif"))
neatklata71107120=ifel(neatklata71107120>0,1,NA)
plot(neatklata71107120)
neatklata7140=rast(paste0("./Geodata/2024/Bogs_EDI/purvi_EDI_projekts/",
                           "purvi/!LV_kopa_apv1020_30_05_2022/",
                           "!LV_kopa_apv1020_30_05_2022/",
                           "Neatklata_purviem_raksturiga_zemsedze_7140.tif"))
neatklata7140=ifel(neatklata7140>0,1,NA)

raskturiaga71107120=rast(paste0("./Geodata/2024/Bogs_EDI/purvi_EDI_projekts/",
                                   "purvi/!LV_kopa_apv1020_30_05_2022/",
                                   "!LV_kopa_apv1020_30_05_2022/",
                                   "Purviem_neraksturiga_zemsedze_7110_7120.tif"))
raskturiaga71107120=ifel(raskturiaga71107120>0,1,NA)
raksturiaga7140=rast(paste0("./Geodata/2024/Bogs_EDI/purvi_EDI_projekts/",
                               "purvi/!LV_kopa_apv1020_30_05_2022/",
                               "!LV_kopa_apv1020_30_05_2022/",
                               "Purviem_neraksturiga_zemsedze_7140.tif"))
raksturiaga7140=ifel(raksturiaga7140>0,1,NA)

labels71107120=rast(paste0("./Geodata/2024/Bogs_EDI/purvi_EDI_projekts/",
                            "purvi/!LV_kopa_apv1020_30_05_2022/",
                            "!LV_kopa_apv1020_30_05_2022/",
                            "latvija_Labels_B7110_7120.tif"))
labels71107120=ifel(labels71107120>0,1,NA)
labels7140=rast(paste0("./Geodata/2024/Bogs_EDI/purvi_EDI_projekts/",
                        "purvi/!LV_kopa_apv1020_30_05_2022/",
                        "!LV_kopa_apv1020_30_05_2022/",
                        "latvija_Labels_B7140.tif"))
labels7140=ifel(labels7140>0,1,NA)

augstie=cover(cover(neatklata71107120,raskturiaga71107120),labels71107120)
parejas=cover(cover(neatklata7140,raksturiaga7140),labels7140)
tikai_parejas=ifel(parejas==1&augstie==1,NA,parejas)
sunainie=ifel(parejas==1&augstie==1,parejas,NA)

sunu_purvi=cover(augstie,sunainie)

sunu_proj=project(sunu_purvi,template10)
sunuYN=cover(sunu_proj,nulles10)
plot(sunuYN)
writeRaster(sunuYN,
            overwrite=TRUE,
            filename="./RasterGrids_10m/2024/EDI_BogsYN.tif")

# Transitional mires ----
parejas_proj=project(tikai_parejas,template10)
parejasYN=cover(parejas_proj,nulles10)
plot(parejasYN)
writeRaster(parejasYN,
            overwrite=TRUE,
            filename="./RasterGrids_10m/2024/EDI_TransitionalMiresYN.tif")

```

# Chapter 5

## Geodata products

Some raw data need extensive processing prior to EGVs creation. Often, EGVs relay on transforming raw geodata into intermediate products; in other cases, an EGV itself could be created from raw geodata, but it has to be spatially restricted to certain locations. This chapter describes these geodata products and the procedures involved in creating them.

### 5.1 Terrain products

In order to develop the topographic wetness index (TWI) and non-drainage depressions, it was necessary to address water flow in the environment. This is a multi-step procedure that is logical and reliable in mountainous areas and in environments with little hydrological impact. However, in the context of Latvia, this was challenging. These challenges can be addressed in various ways. For example, if reliable (accurate) information on the exact locations of rivers and ditches were available, it could be incorporated into the terrain. Unfortunately, there is no sufficiently accurate information available. Therefore, information about network structures from the [Melioration Cadastre Information System database](#) buffered by 10 m, bridges from the [topographic map](#) and transport structures and bridges from [LVM Open Data](#) were used to address the challenges (both buffered by 10 m). Information about the minimum height above sea level was incorporated into the DEM to be used in further processing.

```
# libs
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(arrow)) {install.packages("arrow"); require(arrow)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)){install.packages("exactextractr");require(exactextractr)}

# reference
template=rast("./Templates/TemplateRasters/LV10m_10km.tif")

# part one ----

# dem raster
reliefs=rast("./Geodata/2024/DEM/mozDEM_10m.tif")

# drainage network structures
st_layers("./Geodata/2024/MKIS/MKIS_2025.gpkg")

dtb=st_read("./Geodata/2024/MKIS/MKIS_2025.gpkg", layer="DrenazasTiklaBuves")
dtb_buffer=st_buffer(dtb,dist=10)

# bridges
tiltL=sfarrow::st_read_parquet("./Geodata/2024/TopographicMap/BridgeL_COMB.parquet")
tiltL_buffer=st_buffer(tiltL,dist=30)
tiltP=sfarrow::st_read_parquet("./Geodata/2024/TopographicMap/BridgeL_COMB.parquet")
```

```

tiltiP_buffer=st_buffer(tiltiP,dist=30)

# LVM
lvm_caurtekas=st_read("./Geodata/2024/LVM_OpenData/LVM_CAURTEKAS/LVM_CAURTEKAS_Shape.shp")
lvm_buffer=st_buffer(lvm_caurtekas,dist=30)

# buffers
st_geometry(dtb_buffer)="geometry"
st_geometry(tiltiL_buffer)="geometry"
st_geometry(tiltiP_buffer)="geometry"
st_geometry(lvm_buffer)="geometry"
visi_buferi=bind_rows(dtb_buffer,tiltiL_buffer,tiltiP_buffer,lvm_buffer)

# incorporation in DEM
visi_buferi$vertiba=exactextractr::exact_extract(reljefts,visi_buferi,"min")

caurumi=fasterize::fasterize(visi_buferi,templis,field="vertiba")
caurumi2=rast(caurumi)
caurumains=app(c(reljefts,caurumi2),fun="min",na.rm=TRUE,
               overwrite=TRUE,
               filename="./Geodata/2024/DEM/caurtDEM_10m.tif")

# cleaning
rm(caurumi)
rm(caurumi2)
rm(dtb)
rm(dtb_buffer)
rm(lvm_buffer)
rm(lvm_caurtekas)
rm(reljefts)
rm(tiltiL)
rm(tiltiL_buffer)
rm(tiltiP)
rm(tiltiP_buffer)
rm(visi_buferi)
rm(caurumains)

```

This DEM was then used for geoprocessing to identify terrain depressions and determine the topographic wetness index (TWI):

1. drainage depressions and their depth layers were prepared after incorporating flow breaks;
2. to calculate the topographic wetness index, terrain depressions without runoff were reviewed, allowing up to ten cell breaks in areas of lower resistance; the rest were filled in;
3. for additional security, the procedure of the second step was repeated to search for and fill in terrain depressions (Wang and Liu, 2006);
4. the result of the third step was used to determine the specific catchment area using D-infinity flow direction;
5. by combining the specific catchment area layer with the slope layer, the topographic wetness index was calculated. A graphical evaluation revealed individual extreme values, which were limited to **20**.

```

# libs
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(whitebox)){install.packages("whitebox");require(whitebox)}

# reference
template=rast("./Templates/TemplateRasters/LV10m_10km.tif")

# part two ----

# DEM

```

```

caurumainis=rast("./Geodata/2024/DEM/caurtDEM_10m.tif")

# Sinks
## breached sinks and depth in sinks
wbt_breach_depressions_least_cost(
  dem = "./Geodata/2024/DEM/caurtDEM_10m.tif",
  output = "./Geodata/2024/DEM/caurtDEM_breachedNF.tif",
  dist = 10,
  fill = FALSE)
wbt_depth_in_sink(dem="./Geodata/2024/DEM/caurtDEM_breachedNF.tif",
  output=("./Geodata/2024/DEM/Terrain_DiS_breached_10m.tif",
  zero_background = TRUE)
wbt_sink(input = "./Geodata/2024/DEM/caurtDEM_breachedNF.tif",
  output = "./Geodata/2024/DEM/Terrain_Sink_breached_10m.tif",
  verbose_mode = FALSE,zero_background = TRUE)
sinks=rast("./Geodata/2024/DEM/Terrain_Sink_breached_10m.tif")

sinks2 <- ifel(sinks >= 1, 1, sinks,
  filename="./Geodata/2024/DEM/Terrain_SinkYN_breached_10m.tif")
plot(sinks2)
unlink("./Geodata/2024/DEM/Terrain_Sink_breached_10m.tif")

# TWI
## breaching
wbt_breach_depressions_least_cost(
  dem = "./Geodata/2024/DEM/caurtDEM_10m.tif",
  output = "./Geodata/2024/DEM/caurtDEM_breachedF.tif",
  dist = 10,
  fill = TRUE)

### filling
wbt_fill_depressions_wang_and_liu(
  dem = "./Geodata/2024/DEM/caurtDEM_breachedF.tif",
  output = "./Geodata/2024/DEM/caurtDEM_BreachFill.tif"
)

### (d inf) flow direction
wbt_d_inf_flow_accumulation(input = "./Geodata/2024/DEM/caurtDEM_BreachFill.tif",
  output = "./Geodata/2024/DEM/caurtDEM_DInfAccu_SCA.tif",
  out_type = "Specific Contributing Area")

### twi
wbt_wetness_index(sca = "./Geodata/2024/DEM/caurtDEM_DInfAccu_SCA.tif",
  slope = "./Geodata/2024/DEM/Terrain_Slope_10m.tif",
  output = "./Geodata/2024/DEM/TWI_caurtDEM.tif")
twi=rast("./Geodata/2024/DEM/TWI_caurtDEM.tif")
hist(twi) # excessively large values
plot(twi)
twi2=ifel(twi>20,20,twi)
plot(twi2)
twi2x=ifel(is.na(twi2)&!is.na(template),20,twi2) # Lake Burtnieks

writeRaster(twi2x,filename="./Geodata/2024/DEM/Terrain_TWI_lim20_caurtDEM.tif")

# cleaning
rm(sinks)
rm(sinks2)
rm(caurumainis)
rm(twi)
rm(twi2)

```

Since the initial DEM input was created by filling in water bodies using interpolation methods, the water bodies show a pronounced terrain, which had to be removed. This was done by overlaying arithmetic mean values of these polygons.

```

# libs
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(arrow)) {install.packages("arrow"); require(arrow)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)){install.packages("exactextractr");require(exactextractr)}

# reference
template=rast("./Templates/TemplateRasters/LV10m_10km.tif")
# third part ----

# dealing with waterbodies
udeni=sfarrow::st_read_parquet("./Geodata/2024/TopographicMap/HidroA_COMB.parquet")

slope=rast("./Geodata/2024/DEM/Terrain_Slope_10m.tif")
aspect=rast("./Geodata/2024/DEM/Terrain_Aspect_10m.tif")
twi=rast("./Geodata/2024/DEM/Terrain_TWI_lim20_cauDEM.tif")
dis=rast("./Geodata/2024/DEM/Terrain_Dis_breached_10m.tif")

# average per waterbody
udeni$slopes=exactextractr::exact_extract(slope,udeni,"mean")
caurumi_slope=fasterize::fasterize(udeni,templis,field="slopes")
caurumi_slope2=rast(caurumi_slope)
caurumains_slope=app(c(caurumi_slope2,slope),fun="first",na.rm=TRUE,
                      overwrite=TRUE,
                      filename="./Geodata/2024/DEM/Terrain_Slope_udeni_10m.tif")
caurumains_slope=terra::rast("./Geodata/2024/DEM/Terrain_Slope_udeni_10m.tif")
caurumains_slope2=terra::mask(caurumains_slope,template,
                               overwrite=TRUE,
                               filename="./RasterGrids_10m/2024/Terrain_Slope_udeni2_10m.tif")
rm(slope)
rm(caurumi_slope)
rm(caurumi_slope2)
rm(caurumains_slope)
rm(caurumains_slope2)

udeni$aspect=exactextractr::exact_extract(aspect,udeni,"mean")
caurumi_aspect=fasterize::fasterize(udeni,templis,field="aspect")
caurumi_aspect2=rast(caurumi_aspect)
caurumi_aspect=app(c(caurumi_aspect2,aspect),fun="first",na.rm=TRUE,
                     overwrite=TRUE,
                     filename="./Geodata/2024/DEM/Terrain_Aspect_udeni_10m.tif")
caurumains_aspect=terra::rast("./Geodata/2024/DEM/Terrain_Aspect_udeni_10m.tif")
caurumains_aspect2=terra::mask(caurumains_aspect,template,
                               overwrite=TRUE,
                               filename="./RasterGrids_10m/2024/Terrain_Aspect_udeni2_10m.tif")
rm(aspect)
rm(caurumi_aspect)
rm(caurumi_aspect2)
rm(caurumains_aspect)
rm(caurumains_aspect2)

udeni$twis=exactextractr::exact_extract(twi,udeni,"mean")
caurumi_TWI=fasterize::fasterize(udeni,templis,field="twis")
caurumi_TWI2=rast(caurumi_TWI)
caurumains_TWI=app(c(caurumi_TWI2,twi),fun="first",na.rm=TRUE,
                     overwrite=TRUE,
                     filename="./Geodata/2024/DEM/Terrain_TWI_udeni_10m.tif")
caurumains_TWI=terra::rast("./Geodata/2024/DEM/Terrain_TWI_udeni_10m.tif")

```

```

caurumains_TWI2=terra::mask(caurumains_TWI,template,
                             overwrite=TRUE,
                             filename=".~/RasterGrids_10m/2024/Terrain_TWI_udeni2_10m.tif")
rm(twi)
rm(caurumi_TWI)
rm(caurumi_TWI2)
rm(caurumains_TWI)
rm(caurumains_TWI2)

udeni$disi=exactextractr::exact_extract(dis,udeni,"mean")
caurumi_Dis=fasterize::fasterize(udeni,templis,field="disi")
caurumi_Dis2=rast(caurumi_Dis)
caurumains_Dis=app(c(caurumi_Dis2,dis),fun="first",na.rm=TRUE,
                    overwrite=TRUE,
                    filename=".~/Geodata/2024/DEM/Terrain_Dis_udeni_10m.tif")
caurumains_Dis=terra::rast("./Geodata/2024/DEM/Terrain_Dis_udeni_10m.tif")
caurumains_Dis2=terra::mask(caurumains_Dis,template,
                            overwrite=TRUE,
                            filename=".~/RasterGrids_10m/2024/Terrain_Dis_udeni2_10m.tif")
rm(udeni)
rm(dis)
rm(caurumi_Dis)
rm(caurumi_Dis2)
rm(caurumains_Dis)
rm(caurumains_Dis2)

# cleaning
unlink("./Geodata/2024/DEM/caurtDEM_breachedF.tif")
unlink("./Geodata/2024/DEM/caurtDEM_breachedNF.tif")
unlink("./Geodata/2024/DEM/caurtDEM_BreachFill.tif")
unlink("./Geodata/2024/DEM/caurtDEM_DInfAccu_SCA.tif")

unlink("./Geodata/2024/DEM/Terrain_Slope_udeni_10m.tif")
unlink("./Geodata/2024/DEM/Terrain_Aspect_udeni_10m.tif")
unlink("./Geodata/2024/DEM/Terrain_Dis_udeni_10m.tif")
unlink("./Geodata/2024/DEM/Terrain_TWI_udeni_10m.tif")

```

## 5.2 Soil texture product

In this section, a unified layer describing categorised soil texture (sand = 1, silt = 2, clay = 3, organic = 4) was created from multiple preprocessed soil texture data sources. The creation of the soil texture product consisted of multiple overlay steps. These steps, along with the processed geodata used, are illustrated as follows:

1. the base soil texture source was [Soil texture layer from the European Soil Database](#). This layer had to be reclassified to match the other layers, as this was not performed during preprocessing;
2. the layer from the first step was overlaid with the [Latvian Quarternary geology data](#) coded as numeric starting with 1;
3. the layer from the second step was overlaid with the [20th century topsoil data in Latvian farmland](#) coded as numeric starting with 1;
4. the layer from [Organic soils as modelled by the LVMI Silava](#) (presence-only) was overlaid with the [Organic soils as modelled by the University of Latvia](#) (presence-absence). After the overlay, it was classified as presence-only;
5. the layer from the third step was the overlaid with the layer from the fourth step and saved for EGV creation.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}

# step 1
step1=rast("./RasterGrids_10m/2024/SoilTXT_ESDAC.tif")
step1x=ifel(step1==1,1,
            ifel(step1==2,2,
                ifel(step1==3,3,
                    ifel(step1==4,4,
                        ifel(step1==8,4,NA)))))
plot(step1x)
step1xy=as.numeric(step1x)
plot(step1xy)

# step 2
step2a=rast("./RasterGrids_10m/2024/SoilTXT_QuarternaryLV.tif")
step2a=as.numeric(step2a)+1
plot(step2a)

step2=cover(step2a,step1x)
plot(step2)

# step 3
step3a=rast("./RasterGrids_10m/2024/SoilTXT_topSoilLV.tif")
step3a=as.numeric(step3a)+1
plot(step3a)

step3=cover(step3a,step2)
plot(step3)

# step 4
step4a=rast("./RasterGrids_10m/2024/SoilTXT_OrganicLU.tif")
step4b=rast("./RasterGrids_10m/2024/SoilTXT_OrganicSilava.tif")

step4c=cover(step4a,step4b)

step4=ifel(step4c==1,4,NA)
plot(step4)

# step 5

step5=cover(step4,step3)
plot(step5)

writeRaster(step5,
            "./RasterGrids_10m/2024/SoilTXT_combined.tif",
            overwrite=TRUE)

```

## 5.3 Landscape classification

In this exercise, “landscape” refers to the representation of different types of land cover and land use classes. The order in which these classes are drawn is important because spatial data from different sources often have mismatching boundaries. This requires addressing both their overlap (1) and filling in gaps where no database information is available (2), as well as deciding how to emphasize objects through certain processing steps, such as buffering. Some elements that are important for characterizing the environment (especially edge effects) may be so small or poorly positioned that they disappear during the rasterisation process (3).

The general landscape layer also serves as a mask for the preparation of further environmental descriptions. This section describes the development of a general (simple) landscape and, in the following document, its

enrichment with more specific environmental ecogeographical variables. The general landscape is stored in the file `Ainava_vienk_mask.tif`. The classes in the order of overlay are as follow:

- Class **100** - Roads;
- (Subclass **720** - Reed, Sedge, Rush beds;)
- Class **200** - Waters;
- Class **300** - Farmlands;
- Class **400** - Allotment gardens, Orchards and Cottages;
- Class **500** - Built-up;
- Class **600** - Forests, Shrublands, Clearings;
- Class **700** - Wetlands;
- Class **800** - Bare Soil and Quarries.

The procedures for their creation are described below:

- Class **100** - **Roads**: roads from various sources. The following sources have been combined to create this class:
  - layers `RoadA_COMB` and `RoadL_COMB` (except the smallest size groups) from [topographic map](#), buffered by 10 m before rasterisation;
  - [LVM open data](#) layers `LVM_MEZA_AUTOCELI`, `LVM_ATTISTAMIE_AUTOCELI`, `LVM_APGRIESANAS_LAUKUMI`, `LVM_IZMAINISANAS_VIETAS`, and `LVM_NOBRAUKTUVES` buffered by 10 m;
  - information from the State Forest Register on unpaved forest tracks has not been used, as these roads do not usually form a continuous break in the canopy. Information on roads from this register is also available in other resources and has not been duplicated.

The command lines below create a layer with landscape class **100**, which is saved in the file `SimpleLandscape_class100_celi.tif` for further processing.

```
# Libs ----
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(arrow)) {install.packages("arrow"); require(arrow)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(gdalUtilities)){install.packages("gdalUtilities");require(gdalUtilities)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}

# templates ----
template_t=rast("./Templates/TemplateRasters/LV10m_10km.tif")
template_r=raster(template_t)

# class 100 ----

#poly
celi_topo=st_read_parquet("./Geodata/2024/TopographicMap/RoadA_COMB.parquet")
celi_topo=celi_topo %>%
  mutate(yes=100) %>%
  dplyr::select(yes)
ctb=st_buffer(celi_topo,dist=10)
r_celi_topo=fasterize(ctb,template_r,field="yes")

# pts
```

```

nobrauktuvės=st_read(
  "./Geodata/2024/LVM_OpenData/LVM_NOBRAUKTUVESES/LVM_NOBRAUKTUVESES_Shape.shp")
nobrauktuvės=nobrauktuvės %>%
  mutate(yes=100) %>%
  dplyr::select(yes)
izmainisanas=st_read(
  "./Geodata/2024/LVM_OpenData/LVM_IZMAINISANAS_VIETAS/LVM_IZMAINISANAS_VIETAS_Shape.shp")
izmainisanas=izmainisanas %>%
  mutate(yes=100) %>%
  dplyr::select(yes)
apgriesanas=st_read(
  "./Geodata/2024/LVM_OpenData/LVM_APGRIESANAS_LAUKUMI/LVM_APGRIESANAS_LAUKUMI_Shape.shp")
apgriesanas=apgriesanas %>%
  mutate(yes=100) %>%
  dplyr::select(yes)
cp=rbind(nobrauktuvės,izmainisanas,apgriesanas)
cpb=st_buffer(cp,dist=10)
r_celi_pts=fasterize(cpb,template_r,field="yes")

# lines
meza_autoceli=st_read(
  "./Geodata/2024/LVM_OpenData/LVM_MEZA_AUTOCELI/LVM_MEZA_AUTOCELI_Shape.shp")
meza_autoceli=meza_autoceli %>%
  mutate(yes=100) %>%
  dplyr::select(yes)
attistamie=st_read(
  "./Geodata/2024/LVM_OpenData/LVM_ATTISTAMIE_AUTOCELI/LVM_ATTISTAMIE_AUTOCELI_Shape.shp")
attistamie=attistamie %>%
  mutate(yes=100) %>%
  dplyr::select(yes)
topo_lines=st_read_parquet("./Geodata/2024/TopographicMap/RoadL_COMB.parquet")
topo_lines=topo_lines %>%
  mutate(yes=100) %>%
  dplyr::select(yes)
cl=bind_rows(meza_autoceli,attistamie,topo_lines)
cl=cl %>%
  dplyr::select(yes)
clb=st_buffer(cl,dist=10)
r_celi_lines=fasterize(clb,template_r,field="yes")

# cleaning
rm(apgriesanas)
rm(attistamie)
rm(celi_topo)
rm(topo_lines)
rm(ctb)
rm(cl)
rm(clb)
rm(cp)
rm(cpb)
rm(izmainisanas)
rm(meza_autoceli)
rm(nobrauktuvės)

# to terra
t_celi_topo=rast(r_celi_topo)
t_celi_pts=rast(r_celi_pts)
t_celi_lines=rast(r_celi_lines)

# cleaning
rm(r_celi_lines)
rm(r_celi_pts)
rm(r_celi_topo)

# union

```

```

plot(t_celi_topo)

road_union1=cover(t_celi_topo,t_celi_pts)
road_union2=cover(road_union1,t_celi_lines,
                  filename=".~/RasterGrids_10m/2024/SimpleLandscape_class100_celi.tif",
                  overwrite=TRUE)

# cleaning
rm(t_celi_topo)
rm(t_celi_pts)
rm(t_celi_lines)
rm(road_union1)
rm(road_union2)

```

- Class 200 - **Waters**: water bodies from various sources. The following are combined to create this class:
  - topographic map layers HidroA\_COMB and HidroL\_COMB (buffered by 5 m);
  - MKIS layer Gravji, buffered by 3 m;
  - LVM open data layers LVM\_GRAVJI, buffered by 5 m.
  - information about ditches from the State Forest Register was not used, as it is either already available in other resources or consists of structures so small that they do not cause a continuous break in the tree canopy.

The command lines below create a layer with landscape class 200, which is saved in the file SimpleLandscape\_class200\_udens\_premask.tif for further processing.

```

# Libs ----
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(arrow)) {install.packages("arrow"); require(arrow)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(gdalUtilities)){install.packages("gdalUtilities");require(gdalUtilities)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}

# templates ----
template_t=rast("./Templates/TemplateRasters/LV10m_10km.tif")
template_r=raster(template_t)

# class 200 ----

# topo
topo_udens_poly=st_read_parquet("./Geodata/2024/TopographicMap/HidroA_COMB.parquet")
topo_udens_poly=topo_udens_poly %>%
  mutate(yes=200) %>%
  dplyr::select(yes) %>%
  st_transform(crs=3059)
topo_udens_lines=st_read_parquet("./Geodata/2024/TopographicMap/HidroL_COMB.parquet")
topo_udens_lines=topo_udens_lines %>%
  mutate(yes=200) %>%
  st_buffer(dist=5) %>%
  dplyr::select(yes) %>%
  st_transform(crs=3059)
topo_udens=rbind(topo_udens_poly,topo_udens_lines)
r_topo_udens=fasterize(topo_udens,template_r,field="yes")
raster::writeRaster(r_topo_udens,
                    "./RasterGrids_10m/2024/SimpleLandscape_class200_topo.tif",
                    progress="text")

```

```

# cleaning
rm(topo_udens_lines)
rm(topo_udens_poly)
rm(topo_udens)
rm(r_topo_udens)

# mkis
st_layers("./Geodata/2024/MKIS/MKIS_2025.gpkg")
mkis_gravji=st_read("./Geodata/2024/MKIS/MKIS_2025.gpkg",layer="Gravji")

mkis_gravji=mkis_gravji %>%
  mutate(yes=200) %>%
  st_buffer(dist=3) %>%
  dplyr::select(yes)
r_mkis_udens=fasterize(mkis_gravji,template_r,field="yes")
raster::writeRaster(r_mkis_udens,
                     "./RasterGrids_10m/2024/SimpleLandscape_class200_mkis.tif",
                     progress="text")

# cleaning
rm(mkis_gravji)
rm(mkis_gravji2)
rm(mkis_gravji3)
rm(r_mkis_udens)

# lvm
lvm_gravji=st_read("./Geodata/2024/LVM_OpenData/LVM_GRAVJI/LVM_GRAVJI_Shape.shp")
lvm_gravji=lvm_gravji %>%
  mutate(yes=200) %>%
  st_buffer(dist=5) %>%
  dplyr::select(yes)
r_lvm_gravji=fasterize(lvm_gravji,template_r,field="yes")
raster::writeRaster(r_lvm_gravji,
                     "./RasterGrids_10m/2024/SimpleLandscape_class200_lvm.tif",
                     progress="text",
                     overwrite=TRUE)

# cleaning
rm(lvm_gravji)
rm(r_lvm_gravji)

# merging
a200=rast("./RasterGrids_10m/2024/SimpleLandscape_class200_topo.tif")
b200=rast("./RasterGrids_10m/2024/SimpleLandscape_class200_mkis.tif")
c200=rast("./RasterGrids_10m/2024/SimpleLandscape_class200_lvm.tif")

udens_cover1=cover(a200,b200)
udens_cover2=cover(udens_cover1,c200,
                  filename="./RasterGrids_10m/2024/SimpleLandscape_class200_udens_premask.tif",
                  overwrite=TRUE)

# cleaning
rm(a200)
rm(b200)
rm(c200)
rm(udens_cover1)
rm(udens_cover2)
unlink("./RasterGrids_10m/2024/SimpleLandscape_class200_topo.tif")
unlink("./RasterGrids_10m/2024/SimpleLandscape_class200_mkis.tif")
unlink("./RasterGrids_10m/2024/SimpleLandscape_class200_lvm.tif")

```

- Class 300 - **Farmland**: agricultural land from the LAD database. The following sources are combined to create this class:
  - [LAD database](#), which, following the decision on grouping (classes are available [here](#)), is divided into three broad groups (in the order of overlap with lower number dominating):

- arable land with class code `310`;
- fallow land with class code `320`;
- grassland with class code `330`;
- orchards and perennial shrub plantations in the general landscape are part of other landscape classes.

The command lines below create a layer with landscape class 300 and its subclasses, which are saved in the file SimpleLandscape\_class300\_lauki\_premask.tif for further processing.

```

# Libs ----
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(arrow)) {install.packages("arrow"); require(arrow)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(gdalUtilities)){install.packages("gdalUtilities");require(gdalUtilities)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}

# templates ----
template_t=rast("./Templates/TemplateRasters/LV10m_10km.tif")
template_r=raster(template_t)

# class 300 ----

# lad
lad_klasem=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
lad=st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")

## arable
amazemem=lad_klasem %>%
  filter(str_detect(SDM_grupa_sakums,"aramz"))
aramzemes=lad %>%
  filter(PRODUCT_CODE %in% amazemem$kods) %>%
  mutate(yes=310) %>%
  dplyr::select(yes)
r_aramzemes_lad=fasterize(aramzemes,template_r,field="yes")
raster::writeRaster(r_aramzemes_lad,
                     "./RasterGrids_10m/2024/SimpleLandscape_class310_aramzemes_lad.tif",
                     progress="text",
                     overwrite=TRUE)

# cleaning
rm(amazemem)
rm(aramzemes)
rm(r_aramzemes_lad)

## fallow
papuvem=lad_klasem %>%
  filter(str_detect(SDM_grupa_sakums,"papuv"))
papuves=lad %>%
  filter(PRODUCT_CODE %in% papuvem$kods) %>%
  mutate(yes=320) %>%
  dplyr::select(yes)
r_papuves_lad=fasterize(papuves,template_r,field="yes")
raster::writeRaster(r_papuves_lad,
                     "./RasterGrids_10m/2024/SimpleLandscape_class320_papuves_lad.tif",
                     progress="text",

```

```

        overwrite=TRUE)

# cleaning
rm(papuvem)
rm(papuves)
rm(r_papuves_lad)

## grassland
zalajiem=lad_klasem %>%
  filter(str_detect(SDM_grupa_sakums,"zālā"))
zalaji=lad %>%
  filter(PRODUCT_CODE %in% zalajiem$kods) %>%
  mutate(yes=330) %>%
  dplyr::select(yes)
r_zalaji_lad=fasterize(zalaji,template_r,field="yes")
raster::writeRaster(r_zalaji_lad,
                     "./RasterGrids_10m/2024/SimpleLandscape_class330_zalaji_lad.tif",
                     progress="text",
                     overwrite=TRUE)

# cleaning
rm(zalajiem)
rm(zalaji)
rm(r_zalaji_lad)

# merging
a300=rast("./RasterGrids_10m/2024/SimpleLandscape_class310_aramzemes_lad.tif")
b300=rast("./RasterGrids_10m/2024/SimpleLandscape_class320_papuves_lad.tif")
c300=rast("./RasterGrids_10m/2024/SimpleLandscape_class330_zalaji_lad.tif")

farmland_cover1=cover(a300,b300)
farmland_cover2=cover(farmland_cover1,
                      c300,
                      filename=paste0("./RasterGrids_10m/2024/",
                                      "SimpleLandscape_class300_lauki_premask.tif"),
                      overwrite=TRUE)

# cleaning
rm(lad)
rm(lad_klasem)
rm(a300)
rm(b300)
rm(c300)
rm(farmland_cover1)
rm(farmland_cover2)
unlink("./RasterGrids_10m/2024/SimpleLandscape_class310_aramzemes_lad.tif")
unlink("./RasterGrids_10m/2024/SimpleLandscape_class320_papuves_lad.tif")
unlink("./RasterGrids_10m/2024/SimpleLandscape_class330_zalaji_lad.tif")

```

- Class 400 - **Allotment Gardens, Orchards and Cottages**. To create this class, the following are combined (in order of overlap):
  - **topographic map** layer BuildA\_v3 values: “poligons\_Vasarnīcu\_apbūve”, “poligons\_Viensētu\_apbūve”, coded as 410;
  - **topographic map** layer LandusA\_COMB values: “poligons\_Augļudārzs”, “poligons\_Augļudārzs”, “poligons\_Saknudārzs”, “poligons\_Ogulājs”, “poligons\_Ogulajs”, “poligons\_Saknudarzs”, coded as 420;
  - **LAD database** rural information layer group (classes are available [here](#)) “augļudārzi”, the result of which is coded as 420.

The command lines below create a layer with landscape class 400, which is saved in the file SimpleLandscape\_class400\_vasarnicas\_premask.tif for further processing.

```

# Libs ----
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}

```

```

if(!require(arrow)) {install.packages("arrow"); require(arrow)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}

# templates ----
template_t=rast("./Templates/TemplateRasters/LV10m_10km.tif")
template_r=raster(template_t)

# class 400 ----

# topo built-up
viensvasar=st_read_parquet("./Geodata/2024/TopographicMap/BuildA_v3.parquet")
table(viensvasar$FNAME,useNA="always")
viensvasar=viensvasar %>%
  filter(FNAME %in% c("poligons_Vasarnīcu_apbūve","poligons_Viensētu_apbūve")) %>%
  mutate(yes=410) %>%
  dplyr::select(yes)
r_vienetasvasarnicas=fasterize(viensvasar,template_r,field="yes")
raster::writeRaster(r_vienetasvasarnicas,
  "./RasterGrids_10m/2024/SimpleLandscape_class410_vasarnicasvienetas_topo.tif",
  progress="text",
  overwrite=TRUE)

# cleaning
rm(viensvasar)
rm(r_darzini_topo)

# topo
darzini_topo=st_read_parquet("./Geodata/2024/TopographicMap/LandusA_COMB.parquet")
table(darzini_topo$FNAME,useNA="always")
darzini_topo=darzini_topo %>%
  filter(FNAME %in% c("poligons_Augludārzs","poligons_Augludārzs","poligons_Saknudārzs",
  "poligons_Ogulājs","poligons_Ogulajs","poligons_Saknudarzs")) %>%
  mutate(yes=410) %>%
  dplyr::select(yes)
r_darzini_topo=fasterize(darzini_topo,template_r,field="yes")
raster::writeRaster(r_darzini_topo,
  "./RasterGrids_10m/2024/SimpleLandscape_class410_darzini_topo.tif",
  progress="text",
  overwrite=TRUE)

# cleaning
rm(darzini_topo)
rm(r_darzini_topo)

# lad
lad_klasem=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
table(lad_klasem$SDM_grupa_sakums,useNA="always")
augludarziem=lad_klasem %>%
  filter(SDM_grupa_sakums=="augludārzi")
lad=st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad=lad %>%
  filter(PRODUCT_CODE %in% augludarziem$kods) %>%
  mutate(yes=420) %>%
  dplyr::select(yes)
r_darzini_lad=fasterize(lad,template_r,field="yes")
raster::writeRaster(r_darzini_lad,
  "./RasterGrids_10m/2024/SimpleLandscape_class420_darzini_lad.tif",
  progress="text",
  overwrite=TRUE)

# cleaning
rm(lad_klasem)
rm(augludarziem)

```

```

rm(lad)
rm(r_darzini_lad)

# merging
a400=rast("./RasterGrids_10m/2024/SimpleLandscape_class410_vasarnicasvienetas_topo.tif")
b400=rast("./RasterGrids_10m/2024/SimpleLandscape_class420_darzini_topo.tif")
c400=rast("./RasterGrids_10m/2024/SimpleLandscape_class420_darzini_lad.tif")

allotment_cover=cover(a400,
                      b400,
                      filename=paste0("./RasterGrids_10m/2024/",
                                      "SimpleLandscape_class400_varnacas_premask.tif"),
                      overwrite=TRUE)

# cleaning
rm(a400)
rm(b400)
rm(allotment_cover)
unlink("./RasterGrids_10m/2024/SimpleLandscape_class410_vasarnicasvienetas_topo.tif")
unlink("./RasterGrids_10m/2024/SimpleLandscape_class420_darzini_topo.tif")
unlink("./RasterGrids_10m/2024/SimpleLandscape_class420_darzini_lad.tif")

```

- Class 500 - **Built-up**: built-up areas, no particular layer or data source used. Filled in at the end (see section “merging and filling” of this chapter) using information from the Dynamic World for places not covered by other classes.
- Class 600 - **Forests, Shrublands, Clearings**: areas covered with trees and shrubs, clearings, and dead forest stands. The following sources have been combined to create this class (in order of overlap):
  - The Global Forest Watch layer records of tree canopy cover loss since 2020, coded as 610;
  - Forest State Register clearings and dead forest stands, the result of which is coded as 610;
  - Forest State Register marked forest stands that are lower than 5 m and seed production plantations, the result of which is coded as 620;
  - topographic map layer FloraL\_COMB classes related to shrubs, buffered by 10 m, coded as 620;
  - topographic map layers LandusA\_COMB classes: “poligons\_Krūmājs”, “poligons\_Krumajs”, “poligons\_Krūmaugu\_plant”, “poligons\_Plantacija\_krum”, coded as 620;
  - LAD database group (classes are available [here](#)) “krūmveida ilggadīgie stādījumi”, the result of which is coded with 620;
  - Forest State Register forest stands with a height of at least 5 m, coded as 630;
  - topographic map layer LandusA\_COMB classes: “poligons\_Parks”, “poligons\_Meza\_kapi”, “poligons\_Kapi”, “poligons\_Kapi\_meza”, the result of which is coded as 640;
  - topographic map layer FloraL\_COMB with tree-related classes, buffered by 10 m, coded as 640;
  - PALSAR Forests layer, coded as 630.

The command lines below create a layer with landscape class 600, which is saved in the file SimpleLandscape\_class600\_meziem\_premask.tif for further processing.

```

# Libs ----
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(arrow)) {install.packages("arrow"); require(arrow)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}

# templates ----
template_t=rast("./Templates/TemplateRasters/LV10m_10km.tif")
template_r=raster(template_t)

```

```

# class 600 ----

# mvr
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")

# clearcuts
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  mutate(yes=610) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,template_r,field="yes")
raster::writeRaster(r_izcirtumi_mvr,
                     "./RasterGrids_10m/2024/SimpleLandscape_class610_izcirtumi_mvr.tif",
                     progress="text",
                     overwrite=TRUE)

# cleaning
rm(izcirtumi)
rm(r_izcirtumi_mvr)

# low stands
# also zkat 16
zemas_audzes=mvr %>%
  filter((zkat == "10" & h10<5) | zkat=="16") %>%
  mutate(yes=620) %>%
  dplyr::select(yes)
r_zemas_mvr=fasterize(zemas_audzes,template_r,field="yes")
raster::writeRaster(r_zemas_mvr,
                     "./RasterGrids_10m/2024/SimpleLandscape_class620_zemas_mvr.tif",
                     progress="text",
                     overwrite=TRUE)

# cleaning
rm(zemas_audzes)
rm(r_zemas_mvr)

# high stands
augstas_audzes=mvr %>%
  filter(zkat == "10" & h10>=5) %>%
  mutate(yes=630) %>%
  dplyr::select(yes)
r_augstas_mvr=fasterize(augstas_audzes,template_r,field="yes")
raster::writeRaster(r_augstas_mvr,
                     "./RasterGrids_10m/2024/SimpleLandscape_class630_augstas_mvr.tif",
                     progress="text",
                     overwrite=TRUE)

# cleaning
rm(augstas_audzes)
rm(r_augstas_mvr)
rm(mvr)

# tcl - since 2020
tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,NA,610,
          filename="./RasterGrids_10m/2024/SimpleLandscape_class610_TCL.tif",
          overwrite=TRUE)

# cleaning
rm(tcl)
rm(tcl2)

# palsar
palsar=rast("./Geodata/2024/Trees/Palsar/Palsar_Forests.tif")
palsar2=ifel(palsar==1,630,NA,
             filename="./RasterGrids_10m/2024/SimpleLandscape_class630_Palsar.tif",
             overwrite=TRUE)

```

```

# cleaning
rm(palsar)
rm(palsar2)

# lad
lad_klasem=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
table(lad_klasem$SDM_grupa_sakums,useNA="always")
lad=st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
krumiem=lad_klasem %>%
  filter(str_detect(SDM_grupa_sakums,"krūmv"))
krumi=lad %>%
  filter(PRODUCT_CODE %in% krumiem$kods) %>%
  mutate(yes=620) %>%
  dplyr::select(yes)
r_krumi_lad=fasterize(krumi,template_r,field="yes")
raster::writeRaster(r_krumi_lad,
                    "./RasterGrids_10m/2024/SimpleLandscape_class620_krumi_lad.tif",
                    progress="text",
                    overwrite=TRUE)

# cleaning
rm(lad_klasem)
rm(lad)
rm(krumiem)
rm(krumi)
rm(r_krumi_lad)

# topo - pkk
pkk_topo=st_read_parquet("./Geodata/2024/TopographicMap/LandusA_COMB.parquet")
table(pkk_topo$FNAME,useNA="always")
pkk_topo=pkk_topo %>%
  filter(FNAME %in% c("poligons_Parks","poligons_Meza_Kapi","poligons_Kapi",
                      "poligons_Kapi_meza")) %>%
  mutate(yes=640) %>%
  dplyr::select(yes)
r_pkk_topo=fasterize(pkk_topo,template_r,field="yes")
raster::writeRaster(r_pkk_topo,
                    "./RasterGrids_10m/2024/SimpleLandscape_class640_pkk_topo.tif",
                    progress="text",
                    overwrite=TRUE)

# cleaning
rm(pkk_topo)
rm(r_pkk_topo)

# topo - shrubs
krumi_topo=st_read_parquet("./Geodata/2024/TopographicMap/LandusA_COMB.parquet")
table(krumi_topo$FNAME,useNA="always")
krumi_topo=krumi_topo %>%
  filter(FNAME %in% c("poligons_Krūmājs","poligons_Krumajs",
                      "poligons_Krūmaugu_plant","poligons_Plantacija_krum")) %>%
  mutate(yes=620) %>%
  dplyr::select(yes)
r_krumi_topo=fasterize(krumi_topo,template_r,field="yes")
raster::writeRaster(r_krumi_topo,
                    "./RasterGrids_10m/2024/SimpleLandscape_class620_krumi_topo.tif",
                    progress="text",
                    overwrite=TRUE)

# cleaning
rm(krumi_topo)
rm(r_krumi_topo)

# topo - linear vegetation
linijas_topo=st_read_parquet("./Geodata/2024/TopographicMap/Floral_COMB.parquet")
table(linijas_topo$FNAME,useNA="always")

# linear shrubs

```

```

krumu_linijsas_topo=linijas_topo %>%
  filter(FNAME=="Krūmu rinda dzīvzogs" | FNAME=="Krūmu rinda gar ceļiem upēm" |
         FNAME=="Krumu_rinda_dzivzogs" | FNAME=="Krumu_rinda_gar_celeiem_upem") %>%
  mutate(yes=620) %>%
  st_buffer(dist=10) %>%
  dplyr::select(yes)
r_krumu_linijsas_topo=fasterize(krumu_linijsas_topo,template_r,field="yes")
raster::writeRaster(r_krumu_linijsas_topo,
                    "./RasterGrids_10m/2024/SimpleLandscape_class620_KrumuLinijas_topo.tif",
                    progress="text",
                    overwrite=TRUE)

# cleaning
rm(krumu_linijsas_topo)
rm(r_krumu_linijsas_topo)

# linear trees
koku_linijsas_topo=linijas_topo %>%
  filter(str_detect(FNAME,"Koku")) %>%
  mutate(yes=640) %>%
  st_buffer(dist=10) %>%
  dplyr::select(yes)
r_koku_linijsas_topo=fasterize(koku_linijsas_topo,template_r,field="yes")
raster::writeRaster(r_koku_linijsas_topo,
                    "./RasterGrids_10m/2024/SimpleLandscape_class640_KokuLinijas_topo.tif",
                    progress="text",
                    overwrite=TRUE)

# cleaning
rm(koku_linijsas_topo)
rm(r_koku_linijsas_topo)
rm(linijas_topo)

# merging
r_krumi_lad=rast("./RasterGrids_10m/2024/SimpleLandscape_class620_krumi_lad.tif")
r_pkki_topo=rast("./RasterGrids_10m/2024/SimpleLandscape_class640_pkki_topo.tif")
r_krumi_topo=rast("./RasterGrids_10m/2024/SimpleLandscape_class620_krumi_topo.tif")
r_krumu_linijsas_topo=rast(
  "./RasterGrids_10m/2024/SimpleLandscape_class620_KrumuLinijas_topo.tif")
r_koku_linijsas_topo=rast(
  "./RasterGrids_10m/2024/SimpleLandscape_class640_KokuLinijas_topo.tif")
r_palsar=rast("./RasterGrids_10m/2024/SimpleLandscape_class630_palsar.tif")
r_tcl=rast("./RasterGrids_10m/2024/SimpleLandscape_class610_TCL.tif")
r_augstas_mvr=rast("./RasterGrids_10m/2024/SimpleLandscape_class630_augstas_mvr.tif")
r_zemas_mvr=rast("./RasterGrids_10m/2024/SimpleLandscape_class620_zemas_mvr.tif")
r_izcirtumi_mvr=rast("./RasterGrids_10m/2024/SimpleLandscape_class610_izcirtumi_mvr.tif")

mezu_cover=cover(r_tcl,r_izcirtumi_mvr)
mezu_cover=cover(mezu_cover,r_zemas_mvr)
mezu_cover=cover(mezu_cover,r_krumu_linijsas_topo)
mezu_cover=cover(mezu_cover,r_krumi_topo)
mezu_cover=cover(mezu_cover,r_krumi_lad)
mezu_cover=cover(mezu_cover,r_augstas_mvr)
mezu_cover=cover(mezu_cover,r_pkki_topo)
mezu_cover=cover(mezu_cover,r_koku_linijsas_topo)
mezu_cover=cover(mezu_cover,r_palsar,
                 filename="./RasterGrids_10m/2024/SimpleLandscape_class600_meziem_premask.tif",
                 overwrite=TRUE)

# cleaning
rm(r_krumi_lad)
rm(r_pkki_topo)
rm(r_krumi_topo)
rm(r_krumu_linijsas_topo)
rm(r_koku_linijsas_topo)
rm(r_palsar)

```

```

rm(r_tcl)
rm(r_augstas_mvr)
rm(r_zemas_mvr)
rm(r_izcirtumi_mvr)
rm(mezu_cover)

unlink("./RasterGrids_10m/2024/SimpleLandscape_class620_krumi_lad.tif")
unlink("./RasterGrids_10m/2024/SimpleLandscape_class640_pkk_topo.tif")
unlink("./RasterGrids_10m/2024/SimpleLandscape_class620_krumi_topo.tif")
unlink("./RasterGrids_10m/2024/SimpleLandscape_class620_KrumuLinijas_topo.tif")
unlink("./RasterGrids_10m/2024/SimpleLandscape_class640_KokuLinijas_topo.tif")
unlink("./RasterGrids_10m/2024/SimpleLandscape_class630_palsar.tif")
unlink("./RasterGrids_10m/2024/SimpleLandscape_class610_TCL.tif")
unlink("./RasterGrids_10m/2024/SimpleLandscape_class630_augstas_mvr.tif")
unlink("./RasterGrids_10m/2024/SimpleLandscape_class620_zemas_mvr.tif")
unlink("./RasterGrids_10m/2024/SimpleLandscape_class610_izcirtumi_mvr.tif")

```

- Class **700 - Wetlands**: combining geospatial data related to reed, sedge and rush beds, marshes, mires, and bogs, **filled in order except class 720 that dominates over waters**. To create this class, the following sources are combined (in order of overlap):
  - **topographic map** layer `LandusA_COMB` classes: “Meldrājs\_ūdenī\_poligons”, “poligons\_Grislajs”, “poligons\_Grīslājs”, “poligons\_Meldrajs”, “poligons\_Meldrājs”, “poligons\_Meldrajs\_udenī”, “poligons\_Nec\_purvs\_grīslājs”, “poligons\_Nec\_purvs\_meldrājs”, “Sēklis\_poligons”, the result of which is coded with 720;
  - **topographic map** layer `LandusA_COMB` classes: “poligons\_Nec\_purvs\_sūnājs”, “poligons\_Sunajs”, “poligons\_Sūnājs”, the result of which is coded with 710;
  - **topographic map** layer `SwampA_COMB`, the result of which is coded as 710;
  - land categories “21”, “22”, and “23” marked in the [State Forest Register](#), the result of which is coded as 710;
  - land categories “41” and “42” marked in the [State Forest Register](#), the result of which is coded as 730;
    - bogs from [Bogs and Mires: EDI](#);
    - transitional mires from [Bogs and Mires: EDI](#);

The command lines below create a layer with landscape class 700, which is saved in the file `SimpleLandscape_class700_mitraj_i_premask.tif` for further processing.

```

# Libs ----
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(arrow)) {install.packages("arrow"); require(arrow)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}

# templates ----
template_t=rast("./Templates/TemplateRasters/LV10m_10km.tif")
template_r=raster(template_t)

# class 700 ----

# topo
topo=st_read_parquet("./Geodata/2024/TopographicMap/LandusA_COMB.parquet")
table(topo$FNAME,useNA="always")

## ReedSedgeRush

```

```

niedraji_topo=topo %>%
  filter(FNAME %in% c("Meldrājs_ūdenī_polygons","poligons_Grislajs","poligons_Grislājs",
                      "poligons_Meldrajs","poligons_Meldrājs","poligons_Meldrajs_udenī",
                      "poligons_Nec_purvs_grislājs",
                      "poligons_Nec_purvs_meldrājs",
                      "Sēklis_polygons")) %>%
  mutate(yes=720) %>%
  dplyr::select(yes)
r_niedraji_topo=fasterize(niedraji_topo,template_r,field="yes")
raster::writeRaster(r_niedraji_topo,
                    "./RasterGrids_10m/2024/SimpleLandscape_class720_niedraji_topo.tif",
                    progress="text",
                    overwrite=TRUE)

# cleaning
rm(niedraji_topo)
rm(r_niedraji_topo)

## bogs
purvi_topo=topo %>%
  filter(FNAME %in% c("poligons_Nec_purvs_sūnājs",
                      "poligons_Sunajs","poligons_Sūnājs")) %>%
  mutate(yes=710) %>%
  dplyr::select(yes)
topo_purvi=st_read_parquet("./Geodata/2024/TopographicMap/SwampA_COMB.parquet")
topo_purvi=topo_purvi %>%
  mutate(yes=710) %>%
  dplyr::select(yes)
purvi=rbind(purvi_topo,topo_purvi)
r_purvi_topo=fasterize(purvi,template_r,field="yes")
raster::writeRaster(r_purvi_topo,
                    "./RasterGrids_10m/2024/SimpleLandscape_class710_purvi_topo.tif",
                    progress="text",
                    overwrite=TRUE)

# cleaning
rm(purvi_topo)
rm(topo_purvi)
rm(purvi)
rm(r_purvi_topo)

# mvr
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")

# bogs and mires
mvr_purvi=mvr %>%
  filter(zkat %in% c("21","22","23")) %>%
  mutate(yes=710) %>%
  dplyr::select(yes)
r_purvi_mvr=fasterize(mvr_purvi,template_r,field="yes")
raster::writeRaster(r_purvi_mvr,
                    "./RasterGrids_10m/2024/SimpleLandscape_class710_purvi_mvr.tif",
                    progress="text",
                    overwrite=TRUE)

# cleaning
rm(mvr_purvi)
rm(r_purvi_mvr)

# beavers
mvr_bebri=mvr %>%
  filter(zkat %in% c("41","42")) %>%
  mutate(yes=730) %>%
  dplyr::select(yes)
r_bebri_mvr=fasterize(mvr_bebri,template_r,field="yes")
raster::writeRaster(r_bebri_mvr,
                    "./RasterGrids_10m/2024/SimpleLandscape_class730_bebri_mvr.tif",

```

```

        progress="text",
        overwrite=TRUE)

# cleaning
rm(mvr_bebri)
rm(r_bebri_mvr)
rm(mvr)

# merging
r_niedraji_topo=rast("./RasterGrids_10m/2024/SimpleLandscape_class720_niedraji_topo.tif")
r_purvi_topo=rast("./RasterGrids_10m/2024/SimpleLandscape_class710_purvi_topo.tif")
r_purvi_mvr=rast("./RasterGrids_10m/2024/SimpleLandscape_class710_purvi_mvr.tif")
r_bebri_mvr=rast("./RasterGrids_10m/2024/SimpleLandscape_class730_bebri_mvr.tif")
mires=rast("./RasterGrids_10m/2024/EDI_TransitionalMiresYN.tif")
miresY=ifel(mires==1,NA)
bogs=rast("./RasterGrids_10m/2024/EDI_BogsYN.tif")
bogsY=ifel(bogs==1,NA)

wetlands_cover=cover(r_niedraji_topo,r_purvi_topo)
wetlands_cover=cover(wetlands_cover,r_purvi_mvr)
wetlands_cover=cover(wetlands_cover,r_bebri_mvr)
wetlands_cover=cover(wetlands_cover,miresY)
wetlands_cover=cover(wetlands_cover,
                     bogsY,
                     filename=paste0("./RasterGrids_10m/2024/",
                                     "SimpleLandscape_class700_mitraj_i_premask.tif"),
                     overwrite=TRUE)

# cleaning
rm(r_niedraji_topo)
rm(r_purvi_topo)
rm(r_purvi_mvr)
rm(r_bebri_mvr)
rm(bogs)
rm(bogsY)
rm(mires)
rm(miresY)
rm(topo)
rm(wetlands_cover)

unlink("./RasterGrids_10m/2024/SimpleLandscape_class710_purvi_topo.tif")
unlink("./RasterGrids_10m/2024/SimpleLandscape_class710_purvi_mvr.tif")
unlink("./RasterGrids_10m/2024/SimpleLandscape_class730_bebri_mvr.tif")

```

- Class 800 - **Bare Soil and Quarries**: combining layers related to bare soil, heaths, and quarries. The following have been combined to create this class (in order of overlap):
  - **topographic map** layer `LandusA_COMB` classes: “poligons\_Smiltajs”, “poligons\_Smiltajs”, “poligons\_Grants”, “poligons\_Kūdra”, “poligons\_Virsajs”, the result of which is coded with 800;
  - land categories “33” and “34” marked in the [State Forest Register](#), the result of which is coded as 800.

The command lines below create a layer with landscape class 800, which is saved in the file `SimpleLandscape_class800_smiltaji_premask.tif` for further processing.

```

# Libs ----
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(arrow)) {install.packages("arrow"); require(arrow)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

```

```

if(!require(readxl)) {install.packages("readxl"); require(readxl)}

# templates ----
template_t=rast("./Templates/TemplateRasters/LV10m_10km.tif")
template_r=raster(template_t)

# class 800 ----

smiltaji_topo=st_read_parquet("./Geodata/2024/TopographicMap/LandusA_COMB.parquet")


```

## Merging and filling

The command lines below combine the previously created layers with the landscape classes in the correct order and ensure that gaps are filled with the appropriately classified Dynamic World composite for April-August 2024. After masking to match the analysis space, the layer is saved in the file `Ainava_vienk_mask.tif` for further processing.

```

# Libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}

# templates ----
template_t=rast("./Templates/TemplateRasters/LV10m_10km.tif")
template_r=raster(template_t)

# final merging and covering ----

# DW
dynworld=rast("Geodata/2024/DynamicWorld/DW_2024_apraug.tif")
klases=matrix(c(0,200,
               1,620,
               2,330,
               3,720,
               4,310,
               5,710,
               6,500,
               7,800,
               8,500),ncol=2,byrow=TRUE)
dw2=terra::classify(dynworld,klases)
writeRaster(dw2,
            "./RasterGrids_10m/2024/DW_reclass.tif",
            overwrite=TRUE)

# other layers
celi=rast("./RasterGrids_10m/2024/SimpleLandscape_class100_celi.tif")
plot(celi)

niedraji=rast("RasterGrids_10m/2024/SimpleLandscape_class720_niedraji_topo.tif")
plot(niedraji)

udeni=rast("./RasterGrids_10m/2024/SimpleLandscape_class200_udens_premask.tif")
plot(udeni)

lauki=rast("./RasterGrids_10m/2024/SimpleLandscape_class300_lauki_premask.tif")
plot(lauki)

vasarnicas=rast("./RasterGrids_10m/2024/SimpleLandscape_class400_varnicas_premask.tif")
plot(vasarnicas)

mezi=rast("./RasterGrids_10m/2024/SimpleLandscape_class600_meziem_premask.tif")
plot(mezi)

mitraji=rast("./RasterGrids_10m/2024/SimpleLandscape_class700_mitraji_premask.tif")
plot(mitraji)

smiltaji=rast("./RasterGrids_10m/2024/SimpleLandscape_class800_smiltaji_premask.tif")
plot(smiltaji)

dw2=rast("./RasterGrids_10m/2024/DW_reclass.tif")
plot(dw2)

# covering in correct order
rastri_ainavai=cover(celi,niedraji)
rastri_ainavai=cover(rastri_ainavai,udeni)
rastri_ainavai=cover(rastri_ainavai,lauki)
rastri_ainavai=cover(rastri_ainavai,vasarnicas)
rastri_ainavai=cover(rastri_ainavai,mezi)
rastri_ainavai=cover(rastri_ainavai,mitraji)
rastri_ainavai=cover(rastri_ainavai,smiltaji)
rastri_ainavai=cover(rastri_ainavai,dw2,
                     filename="./RasterGrids_10m/2024/Ainava_vienkarsa.tif",
                     overwrite=TRUE)
plot(rastri_ainavai)

```

```

# cleaning
rm(celi)
rm(niedraji)
rm(udeni)
rm(lauki)
rm(vasarnicas)
rm(mezi)
rm(mitraji)
rm(smiltaji)
rm(klases)
rm(dynworld)
rm(dw2)
rm(rastri_ainava)

# masking
rastrs_ainava=rast("./RasterGrids_10m/2024/Ainava_vienkarsa.tif")
plot(rastrs_ainava)
freq(rastrs_ainava)

masketa_ainava=terra::mask(rastrs_ainava,
                           template_t,
                           filename="./RasterGrids_10m/2024/Ainava_vienk_mask.tif",
                           overwrite=TRUE)
plot(masketa_ainava)

# cleaning
rm(rastrs_ainava)
rm(masketa_ainava)

```

## 5.4 Landscape diversity

This subsection summarizes the input products related to the landscape described in the previous section – raster layers prepared at a 10 m resolution, which characterize the classes found in the landscape (environment), as well as the subsequent preprocessing for the preparation of the EGVs.

The calculations of the Shannon diversity index are so computationally intensive that it is not rationally possible to perform them at every landscape scale around each analysis cell (EGV-cell). Furthermore, they cannot be directly aggregated to speed up the calculation. Therefore, a decision has been made on the raster cell size, which:

- is formed as a multiplication of the EGV-cell by an integer;
- is large enough to account for environmental variability. Therefore, the EGV-cell itself (or multiplication by 1) is not suitable - there is very little variability in land cover and land use within an area of 1 ha. Consequently, the raster cell size for calculation of Shannon index should be as large as possible without becoming so large that it artificially inflates spatial autocorrelation and loose spatial relevance;
- allows to build every landscape scale from several diversity-index–level cells.

Since we use spatially weighted zonal statistics in the preparation of EGVs, and the smallest landscape scale is  $r = 500$  m around the centre of the EGV-cell, it has been decided to calculate the landscape diversity index for individual cells with a side length of 500 m (i.e., 25 ha landscapes). This means that the smallest number of units used for the development of the EGVs is nine (for a landscape scale of  $r = 500$  m around the centre of the EGV-cell).

Three principal environments are described using diversity indices: overall landscape, farmland, and forests. To make them easier to reproduce and locate, each is described in a separate section below.

### 5.4.1 Overall landscape

Combination of three layers is involved to describe overall landscape diversity:

- as the lowest in hierarchy is `Ainava_vienk_mask.tif`, prepared in section [Landscape classification](#);
- farmland diversity as the top layer in the hierarchy. Prepared based on relatively broad [agricultural codes \(field - SDM\\_grupa\\_sakums\)](#) from [Rural Support Service's information on declared fields](#). Only cells corresponding to declared fields contain values; others are empty will inherit values from other layers during overlay. Codes used range from 351 to 362;
- forest diversity is the second layer in hierarchy. This layer describes dominant tree species groups in each stand with stand, derived from stand-level inventory data combined with age group as used in forestry practice. Values used in this classification are available from [database description](#).
  - tree species groups:
    - \* coniferous species codes: “1”, “3”, “13”, “14”, “15”, “22”, “23”, “28”;
    - \* boreal deciduous species codes: “4”, “6”, “8”, “9”, “19”, “20”, “21”, “32”, “35”, “68”;
    - \* temperate deciduous species codes: “10”, “11”, “12”, “16”, “17”, “18”, “24”, “25”, “26”, “27”, “28”, “29”, “50”, “61”, “62”, “63”, “64”, “65”, “66”, “67”, “69”;
    - \* classification: a forest is considered coniferous if timber volume of coniferous species in the top tree layer constitutes at least 75% of the total timer volume. Otherwise, it can be considered boreal deciduous if the respective proportion is at least 75%, or temperate deciduous if the respective proportion is at least 50%; else it is considered mixed.
  - tree age groups:
    - \* forests are considered young if they are registered with age groups “1”, “2” or “3”;
    - \* forests are considered old if they are registered with age groups “4”, or “5”;
  - created codes are formatted as factors and then again as scalars, with 660 added.

Once the landscape classification is done, diversity index is calculated for 25 ha landscapes using the function `egvtools::landscape_function`. To guard value coverage, inverse distance weighted (power = 2) gap filling is incorporated; however, there were no gaps to fill.

```
# Libs ----
if(!require(egvtools)) {install.packages("egvtools"); require(egvtools)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(arrow)) {install.packages("arrow"); require(arrow)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}

# templates ----
template_t=rast("./Templates/TemplateRasters/LV10m_10km.tif")
template_r=raster(template_t)

# overall diversity ----

## Farmland broad ----

# classification
culturecodes=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
culturecodes$kods=as.character(culturecodes$kods)
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad2=lad %>%
  left_join(culturecodes, by=c("PRODUCT_CODE"="kods")) %>%
  mutate(numeric_code=as.numeric(as.factor(SDM_grupa_sakums))+350) %>%
  filter(!is.na(numeric_code))
table(lad2$numeric_code,useNA = "always")
```

```

# input layer
polygon2input(vector_data = lad2,
              template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
              out_path = "./RasterGrids_10m/2024/",
              file_name = "Diversity_FarmlandBroad_only.tif",
              value_field = "numeric_code",
              fun="first",
              prepare=FALSE,
              project_mode = "auto")

# cleaning
rm(culturecodes)
rm(lad)
rm(lad2)

## Forests broad ----

# data
mvr=sfarrow::st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")

# species groups
skujkoki=c("1","3","13","14","15","22","23","28") # 8
saurlapji=c("4","6","8","9","19","20","21","32","35","68") # 10
platlapji=c("10","11","12","16","17","18","24","25","26","27","28","29","50",
           "61","62","63","64","65","66","67","69") # 21

# classification
mvr2=mvr %>%
  mutate(vol_coniferous=ifelse(s10 %in% coniferous,v10,0) +
        ifelse(s11 %in% coniferous,v11,0)+ifelse(s12 %in% coniferous,v12,0) +
        ifelse(s13 %in% coniferous,v13,0)+ifelse(s14 %in% coniferous,v14,0),
        vol_boreal=ifelse(s10 %in% boreal_deciduous,v10,0) +
        ifelse(s11 %in% boreal_deciduous,v11,0)+ifelse(s12 %in% boreal_deciduous,v12,0) +
        ifelse(s13 %in% boreal_deciduous,v13,0)+ifelse(s14 %in% boreal_deciduous,v14,0),
        vol_temperate=ifelse(s10 %in% temperate_deciduous,v10,0) +
        ifelse(s11 %in% temperate_deciduous,v11,0)+ifelse(s12 %in%
          temperate_deciduous,v12,0) +
        ifelse(s13 %in% temperate_deciduous,v13,0)+ifelse(s14 %in%
          temperate_deciduous,v14,0)) %>%
  mutate(vol_total=vol_coniferous+vol_boreal+vol_temperate) %>%
  mutate(forest_type=ifelse(vol_coniferous/vol_total>=0.75,"coniferous",
                            ifelse(vol_boreal/vol_total>=0.75,"boreal",
                                   ifelse(vol_temperate/vol_total>0.5,"temperate",
                                         "mixed")))) %>%
  mutate(forest_age=ifelse(vgr=="1"|vgr=="2"|vgr=="3","young",
                           ifelse(vgr=="4"|vgr=="5","old",NA))) %>%
  filter(!is.na(forest_type)) %>%
  filter(!is.na(forest_age)) %>%
  mutate(divbroad_class=paste0(forest_type,"_",forest_age)) %>%
  mutate(divbroad_numeric=as.numeric(as.factor(divbroad_class))+660) %>%
  filter(!is.na(divbroad_numeric))

# input layer
polygon2input(vector_data = mvr2,
              template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
              out_path = "./RasterGrids_10m/2024/",
              file_name = "Diversity_ForestBroad_only.tif",
              value_field = "divbroad_numeric",
              fun="first",
              prepare=FALSE,
              project_mode = "auto",
              overwrite = TRUE)

# cleaning
rm(mvr)
rm(mvr2)

```

```

## overall classification ----

simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")

## Covered classes for general diversity ----

farmland_broad=rast("./RasterGrids_10m/2024/Diversity_FarmlandBroad_only.tif")
forests_broad=rast("./RasterGrids_10m/2024/Diversity_ForestBroad_only.tif")

diversity_classes=cover(farmland_broad,forests_broad)
diversity_classes2=cover(diversity_classes,simple_landscape,
                        filename="./RasterGrids_10m/2024/Diversity_GeneralLandscapeBroad.tif",
                        overwrite=TRUE)

rm(simple_landscape)
rm(farmland_broad)
rm(forests_broad)
rm(diversity_classes)
rm(diversity_classes2)

## Diversity index at 25ha ----

res_tbl <- landscape_function(
  landscape      = "./RasterGrids_10m/2024/Diversity_GeneralLandscapeBroad.tif",
  zones          = "./Templates/TemplateGrids/tikls500_sauzeme.parquet",
  id_field       = "rinda500",
  tile_field     = "tks50km",
  template       = "./Templates/TemplateRasters/LV500m_10km.tif",
  out_dir        = "./RasterGrids_500m/2024/",
  out_filename   = "Diversity_GeneralLandscape_500x.tif",
  out_layername  = "Diversity_GeneralLandscape_500x",
  what           = "lsm_l_shdi",
  rasterize_engine = "fasterize",
  n_workers      = 8,
  future_max_size = 3 * 1024^3,
  fill_gaps      = TRUE,
  plot_gaps      = TRUE,
  plot_result    = TRUE
)
print(res_tbl)
plot(rast("./RasterGrids_500m/2024/Diversity_GeneralLandscape_500x.tif"))
rm(res_tbl)

```

## 5.4.2 Forest diversity

An input grid with a cell size of 10 m covers the entire territory of Latvia. It contains the following values, in order of hierarchy:

- State Forest Service's Forest State Register code, in which the code of the dominant tree species is multiplied by 1000 and the age group code is added. However, before rasterisation, geometries in which no code has been assigned or one of the code components is 0, are excluded;
- forest diversity class values prepared in [Overall landscape diversity](#);
- forest classes from [Landscape classification](#);
- value 1 for all other cells located in the territory of Latvia.

Once the landscape classification is done, the Shannon's diversity index is calculated for 25 ha landscapes using the function `egvtools::landscape_function`. To ensure value coverage, inverse distance weighted (power = 2) gap filling is incorporated; however, there were no gaps to fill.

```

# Libs ----
if(!require(egvtools)) {install.packages("egvtools"); require(egvtools)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(arrow)) {install.packages("arrow"); require(arrow)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}

# templates ----
template_t=rast("./Templates/TemplateRasters/LV10m_10km.tif")
template_r=raster(template_t)

# forest diversity ----

## forest broad ----
forest_broad=rast("./RasterGrids_10m/2024/Diversity_ForestBroad_only.tif")

## forest codes ----
# mezi
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")

mvr=mvr %>%
  mutate(kods1=as.numeric(s10)*1000,
        kods2=as.numeric(vgr),
        kods=kods1+kods2) %>%
  filter(!is.na(kods)) %>%
  filter(kods1>0) %>%
  filter(kods2>0)

# input layer
polygon2input(vector_data = mvr,
              template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
              out_path = "./RasterGrids_10m/2024/",
              file_name = "Diversity_ForestCodes_only.tif",
              value_field = "kods",
              fun="first",
              prepare=FALSE,
              project_mode = "auto",
              overwrite = TRUE)

# cleaning
rm(mvr)

# simple forests
simple_forests=rast("./RasterGrids_10m/2024/SimpleLandscape_class600_meziem_premask.tif")

## Covered classes for forest diversity ----

forest_codes=rast("./RasterGrids_10m/2024/Diversity_ForestCodes_only.tif")
plot(forest_codes)
forest_covered=cover(forest_codes,forest_broad)
forest_covered=cover(forest_covered,simple_forests)
plot(forest_covered)

forest_covered2=cover(forest_covered,template_t,
                      filename="./RasterGrids_10m/2024/Diversity_ForestsDetailed.tif",
                      overwrite=TRUE)
plot(forest_covered2)

# cleaning
rm(forest_codes)
rm(forest_covered)
rm(forest_covered2)

```

```

rm(forest_broad)
rm(simple_forests)

## Diversity index at 25ha ----

res_tbl <- landscape_function(
  landscape      = "./RasterGrids_10m/2024/Diversity_ForestsDetailed.tif",
  zones          = "./Templates/TemplateGrids/tikls500_sauzeme.parquet",
  id_field       = "rinda500",
  tile_field     = "tks50km",
  template        = "./Templates/TemplateRasters/LV500m_10km.tif",
  out_dir         = "./RasterGrids_500m/2024/",
  out_filename   = "Diversity_Forests_500x.tif",
  out_layername  = "Diversity_Forests_500x",
  what           = "lsm_l_shdi",
  rasterize_engine = "fasterize",
  n_workers       = 8,
  future_max_size = 3 * 1024^3,
  fill_gaps       = TRUE,
  plot_gaps       = TRUE,
  plot_result     = TRUE
)
print(res_tbl)

plot(rast("./RasterGrids_500m/2024/Diversity_Forests_500x.tif"))
rm(res_tbl)

```

### 5.4.3 Farmland diversity

A grid with a cell size of 10 m covers the entire territory of Latvia. It contains the following values, listed in order of hierarchy:

- Rural Support Service crop codes with 1000 added;
- farmland diversity class values prepared in [Overall landscape diversity](#);
- farmland classes from [Landscape classification](#);
- value 1 for all other cells located within the territory of Latvia.

Once the landscape classification is done, diversity index is calculated for 25 ha landscapes with function `egvtools::landscape_function`. To guard value coverage, inverse distance weighted (power = 2) gap filling is incorporated; however, there were no gaps to fill.

```

# Libs ----
if(!require(egvtools)) {install.packages("egvtools"); require(egvtools)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(arrow)) {install.packages("arrow"); require(arrow)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}

# templates ----
template_t=rast("./Templates/TemplateRasters/LV10m_10km.tif")
template_r=raster(template_t)

# farmland diversity ----

```

```

## Farmland broad ----

farmland_broad=rast("./RasterGrids_10m/2024/Diversity_FarmlandBroad_only.tif")

## Farmland codes ----

lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$product_code=as.numeric(lad$PRODUCT_CODE)+1000

# input layer
polygon2input(vector_data = lad,
               template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
               out_path = "./RasterGrids_10m/2024/",
               file_name = "Diversity_FarmlandCodes_only.tif",
               value_field = "product_code",
               fun="first",
               prepare=FALSE,
               project_mode = "auto",
               overwrite = TRUE)

# cleaning
rm(lad)

# simple landscapes input
simple_farmland=rast("./RasterGrids_10m/2024/SimpleLandscape_class300_lauki_premask.tif")

## Covered classes for farmland diversity ----

farmland_codes=rast("./RasterGrids_10m/2024/Diversity_FarmlandCodes_only.tif")
farmland_covered=cover(farmland_codes,farmland_broad)
farmland_covered=cover(farmland_covered,simple_farmland)
farmland_covered2=cover(farmland_covered,template_t,
                        filename="./RasterGrids_10m/2024/Diversity_FarmlandDetailed.tif",
                        overwrite=TRUE)
plot(farmland_covered2)

# cleaning
rm(farmland_codes)
rm(farmland_covered)
rm(farmland_covered2)
rm(simple_farmland)
rm(farmland_broad)

## Diversity index at 25ha ----

res_tbl <- landscape_function(
  landscape      = "./RasterGrids_10m/2024/Diversity_FarmlandDetailed.tif",
  zones          = "./Templates/TemplateGrids/tikls500_sauzeme.parquet",
  id_field       = "rinda500",
  tile_field     = "tks50km",
  template       = "./Templates/TemplateRasters/LV500m_10km.tif",
  out_dir        = "./RasterGrids_500m/2024/",
  out_filename   = "Diversity_Farmland_500x.tif",
  out_layername  = "Diversity_Farmland_500x",
  what           = "lsm_l_shdi",
  rasterize_engine = "fasterize",
  n_workers      = 8,
  future_max_size = 3 * 1024^3,
  fill_gaps      = TRUE,
  plot_gaps      = TRUE,
  plot_result    = TRUE
)

```

```
)  
print(res_tbl)  
  
plot(rast("./RasterGrids_500m/2024/Diversity_Farmland_500x.tif"))  
rm(res_tbl)
```

# Chapter 6

## Ecogeographical variables

This section names and provides description (R code with its explanation in procedure) of each of the 538 EGVs created.

Refer to the flowchart below (Fig. 6.1) for a better understanding of how these variable relate. The names used in the figure correspond to EGV layer names and follow naming convention: [group] \_ [specific name] \_ [scale], where:

- group is a broader collection of EGVs describing the same phenomena or ecosystem, derived from the same source, etc.;
- specific name briefly describes the landscape class and/or metrics used in the creation of the layer;
- scale is one of: cell, 500, 1250, 3000, 10000 m around the centre of the EGV-cell. The resolution of each EGV is 1 ha; larger scales are summarised to this resolution.

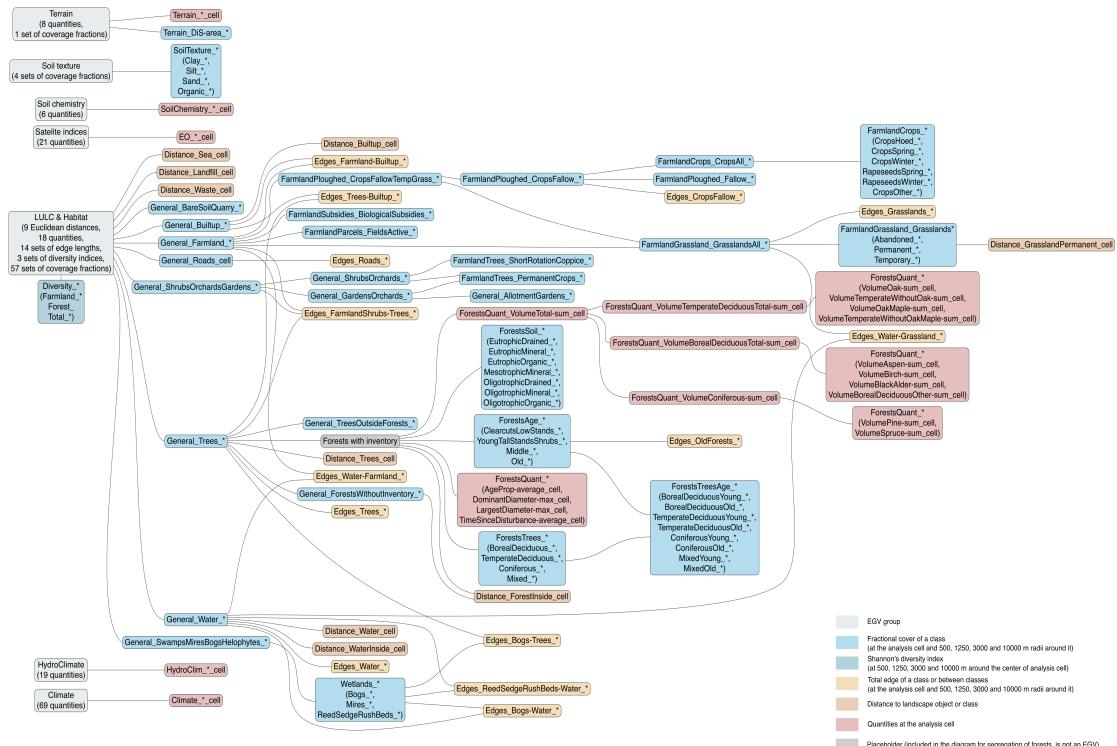


Figure 6.1: Relationships of the created ecogeographical variables.

For cover fraction and edge variables, we first calculated values at the EGV-cell resolution and then used `{exactextract}` to summarise values from larger scales. This package uses pixel area weights to calculate weighted summary statistics, making the aggregation error negligible, particularly at larger scales,

but reduces computation time thousands up to even hundreds of thousands times compared to input resolution (10 m). To further speed up the procedures, we used “sparse” mode in the workflow `egvtools::radius_function()`, thus summarising zonal statistics every 300 m for 3000 m radius buffers and every 1000 m for 10000 m buffers, obtaining near linear reduction in time relative to the number of zones (ninefold and 100 fold further computation time reduction), while loosing less than 0.001 % of variability overall.

We used a slightly different approach with diversity metrics. First, we calculated Shanon’s diversity index at 25 ha raster grid cells, as there is nearly no variability of landscape classes at 1 ha grid cells. Next, we calculated arithmetic mean as zonal staticics value (using the “sparse” mode with the workflow `egvtools::radius_function()`), but we did not create this EGV at the analysis cells scale.

## 6.1 Climate\_CHELSAv2.1-bio1\_cell

**filename:** `Climate_CHELSAv2.1-bio1_cell.tif`

**layername:** `egv_001`

**English name:** Mean annual daily mean air temperature (°C) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Gada vidējā ik dienas vidējā gaisa temperatūra (°C) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows **CHELSA v2.1**. EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----
localname="Climate_CHELSAv2.1-bio1_cell.tif"
layername="egv_001"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio1_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/Raw/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
nosaukums="Climate_CHELSAv2.1-bio1_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.2 Climate\_CHELSAv2.1-bio10\_cell

**filename:** Climate\_CHELSAv2.1-bio10\_cell.tif

**layername:** egv\_002

**English name:** Mean daily mean air temperatures (°C) of the warmest quarter (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Gada siltākā ceturķēna vidējā ik dienas vidējā gaisa temperatūra (°C) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-bio10_cell.tif"
layername="egv_002"
reading "./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio10_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio10_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.3 Climate\_CHELSAv2.1-bio11\_cell

**filename:** Climate\_CHELSAv2.1-bio11\_cell.tif

**layername:** egv\_003

**English name:** Mean daily mean air temperatures (°C) of the coldest quarter (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Gada aukstākā ceturķņa vidējā ik dienas vidējā gaisa temperatūra (°C) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-bio11_cell.tif"
layername="egv_003"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio11_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/RAW/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio11_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.4 Climate\_CHELSAv2.1-bio12\_cell

**filename:** Climate\_CHELSAv2.1-bio12\_cell.tif

**layername:** egv\_004

**English name:** Annual precipitation amount ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Nokrišņu daudzums ( $\text{kg m}^{-2}$  gadā) gadā (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
```

```

# job ----

localname="Climate_CHELSAv2.1-bio12_cell.tif"
layername="egv_004"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio12_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/Raw/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio12_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.5 Climate\_CHELSAv2.1-bio13\_cell

**filename:** Climate\_CHELSAv2.1-bio13\_cell.tif

**layername:** egv\_005

**English name:** Precipitation amount ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) of the wettest month (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Slapjākā mēneša nokrišņu daudzums ( $\text{kg m}^{-2} \text{ mēnesī}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-bio13_cell.tif"
layername="egv_005"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio13_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/Raw/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio13_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

```

grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
rawfile_path = reading,
out_path     = "./RasterGrids_100m/2024/RAW/",
file_name    = localname,
layer_name   = layername,
fill_gaps    = TRUE,
smooth       = TRUE,
smooth_radius_km = 5,
plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio13_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.6 Climate\_CHELSAv2.1-bio14\_cell

**filename:** Climate\_CHELSAv2.1-bio14\_cell.tif

**layername:** egv\_006

**English name:** Precipitation amount ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) of the driest month (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Sausākā mēneša nokrišņu daudzums ( $\text{kg m}^{-2} \text{ mēnesī}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egv-tools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-bio14_cell.tif"
layername="egv_006"
reading=".~/Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio14_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)

```

```

print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio14_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.7 Climate\_CHELSAv2.1-bio15\_cell

**filename:** Climate\_CHELSAv2.1-bio15\_cell.tif

**layername:** evg\_007

**English name:** Precipitation seasonality ( $\text{kg m}^{-2}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Nokrišņu sezonālitāte ( $\text{kg m}^{-2}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egv-tools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-bio15_cell.tif"
layername="evg_007"
reading "./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio15_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio15_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)

```

```

centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.8 Climate\_CHELSAv2.1-bio16\_cell

**filename:** Climate\_CHELSAv2.1-bio16\_cell.tif

**layername:** evg\_008

**English name:** Mean monthly precipitation amount ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) of the wettest quarter (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Slapjākā ceturkšņa vidējais nokrišņu daudzums mēnesī ( $\text{kg m}^{-2} \text{ mēnesī}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egv-tools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-bio16_cell.tif"
layername="evg_008"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio16_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/RAW/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio16_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.9 Climate\_CHELSAv2.1-bio17\_cell

**filename:** Climate\_CHELSAv2.1-bio17\_cell.tif

**layername:** evg\_009

**English name:** Mean monthly precipitation amount ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) of the driest quarter (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Sausākā ceturķņa vidējais nokrišņu daudzums mēnesī ( $\text{kg m}^{-2} \text{ mēnesī}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-bio17_cell.tif"
layername="evg_009"
reading "./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio17_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio17_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.10 Climate\_CHELSAv2.1-bio18\_cell

**filename:** Climate\_CHELSAv2.1-bio18\_cell.tif

**layername:** evg\_010

**English name:** Mean monthly precipitation amount ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) of the warmest quarter (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Siltākā ceturķšņa vidējais nokrišņu daudzums mēnesī ( $\text{kg m}^{-2}$  mēnesī) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-bio18_cell.tif"
layername="egv_010"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio18_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/RAW/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio18_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.11 Climate\_CHELSAv2.1-bio19\_cell

**filename:** Climate\_CHELSAv2.1-bio19\_cell.tif

**layername:** egv\_011

**English name:** Mean monthly precipitation amount ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) of the coldest quarter (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Aukstākā ceturķšņa vidējais nokrišņu daudzums mēnesī ( $\text{kg m}^{-2}$  mēnesī) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-bio19_cell.tif"
layername="egv_011"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio19_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/RAW/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio19_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.12 Climate\_CHELSAv2.1-bio2\_cell

**filename:** Climate\_CHELSAv2.1-bio2\_cell.tif

**layername:** egv\_012

**English name:** Mean diurnal air temperature range (°C) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējā diennakts gaisa temperatūru amplitūda (°C) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows **CHELSA v2.1**. EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-bio2_cell.tif"
layername="egv_012"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio2_cell.tif"

```

```

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path = "./RasterGrids_100m/2024/RAW/",
  file_name = localname,
  layer_name = layername,
  fill_gaps = TRUE,
  smooth = TRUE,
  smooth_radius_km = 5,
  plot_result = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio2_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.13 Climate\_CHELSAv2.1-bio3\_cell

**filename:** Climate\_CHELSAv2.1-bio3\_cell.tif

**layername:** egv\_013

**English name:** Isothermality (ratio of diurnal variation to annual variation in air temperatures) (°C) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Izotermalitāte (attiecība starp diennakts un gada gaisa temperatūras svārstībām) (°C) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-bio3_cell.tif"
layername="egv_013"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio3_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path = "./RasterGrids_100m/2024/RAW/",
  file_name = localname,
  layer_name = layername,
  fill_gaps = TRUE,

```

```

smooth      = TRUE,
smooth_radius_km = 5,
plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio3_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.14 Climate\_CHELSAv2.1-bio4\_cell

**filename:** Climate\_CHELSAv2.1-bio4\_cell.tif

**layername:** evg\_014

**English name:** Temperature seasonality (standard deviation of the monthly mean air temperatures) ( $^{\circ}\text{C}/100$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Temperatūru sezonalitāte (mēneša vidējo gaisa temperatūru standartnovirze) ( $^{\circ}\text{C}/100$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows **CHELSA v2.1**. EGV is prepared using the workflow `egv-tools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-bio4_cell.tif"
layername="evg_014"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio4_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

nosaukums="Climate_CHELSAv2.1-bio4_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.15 Climate\_CHELSAv2.1-bio5\_cell

**filename:** Climate\_CHELSAv2.1-bio5\_cell.tif

**layername:** egv\_015

**English name:** Mean daily maximum air temperature (°C) of the warmest month (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Siltākā mēneša vidējā ik dienas augstākā gaisa temperatūra (°C) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-bio5_cell.tif"
layername="egv_015"
reading "./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio5_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/RAW/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio5_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)

```

```

merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.16 Climate\_CHELSAv2.1-bio6\_cell

**filename:** Climate\_CHELSAv2.1-bio6\_cell.tif

**layername:** evg\_016

**English name:** Mean daily minimum air temperature (°C) of the coldest month (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Aukstākā mēneša vidējā ik dienas zemākā gaisa temperatūra (°C) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-bio6_cell.tif"
layername="evg_016"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio6_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/RAW/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio6_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.17 Climate\_CHELSAv2.1-bio7\_cell

**filename:** Climate\_CHELSAv2.1-bio7\_cell.tif

**layername:** evg\_017

**English name:** Annual range of air temperature (°C) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Gada gaisa temperatūru amplitūda (°C) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows **CHELSA v2.1**. EGV is prepared using the workflow `egv-tools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-bio7_cell.tif"
layername="evg_017"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio7_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio7_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.18 Climate\_CHELSAv2.1-bio8\_cell

**filename:** Climate\_CHELSAv2.1-bio8\_cell.tif

**layername:** evg\_018

**English name:** Mean daily mean air temperatures (°C) of the wettest quarter (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Slapjākā ceturkšņa vidējā ik dienas vidējā gaisa temperatūra (°C) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-bio8_cell.tif"
layername="egv_018"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio8_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio8_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.19 Climate\_CHELSAv2.1-bio9\_cell

**filename:** Climate\_CHELSAv2.1-bio9\_cell.tif

**layername:** egv\_019

**English name:** Mean daily mean air temperatures (°C) of the driest quarter (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Sausākā ceturķšņa vidējā ik dienas vidējā gaisa temperatūra (°C) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
```

```

# job ----

localname="Climate_CHELSAv2.1-bio9_cell.tif"
layername="egv_019"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-bio9_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/Raw/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-bio9_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.20 Climate\_CHELSAv2.1-clt-max\_cell

**filename:** Climate\_CHELSAv2.1-clt-max\_cell.tif

**layername:** egv\_020

**English name:** Mean of monthly maximum cloud area fraction (%) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Mēneša maksimumu vidējais mākoņu segums (%) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-clt-max_cell.tif"
layername="egv_020"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-clt-max_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/Raw/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-clt-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

```

grid_path = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
rawfile_path = reading,
out_path = "./RasterGrids_100m/2024/RAW/",
file_name = localname,
layer_name = layername,
fill_gaps = TRUE,
smooth = TRUE,
smooth_radius_km = 5,
plot_result = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-clt-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.21 Climate\_CHELSAv2.1-clt-mean\_cell

**filename:** Climate\_CHELSAv2.1-clt-mean\_cell.tif

**layername:** evg\_021

**English name:** Mean monthly mean cloud area fraction (%) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējais ik mēneša vidējais mākoņu segums (%) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows CHELSA v2.1. EGV is prepared using the workflow egvtools::downscale2egv() with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-clt-mean_cell.tif"
layername="evg_021"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-clt-mean_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path = "./RasterGrids_100m/2024/RAW/",
  file_name = localname,
  layer_name = layername,
  fill_gaps = TRUE,
  smooth = TRUE,
  smooth_radius_km = 5,
  plot_result = TRUE)
print(df)

```

```

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-clt-mean_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.22 Climate\_CHELSAv2.1-clt-min\_cell

**filename:** Climate\_CHELSAv2.1-clt-min\_cell.tif

**layername:** evg\_022

**English name:** Mean of monthly minimum cloud area fraction (%) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Mēneša minimumu vidējais mākoņu segums (%) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egv-tools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-clt-min_cell.tif"
layername="evg_022"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-clt-min_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/RAW/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-clt-min_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)

```

```

centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.23 Climate\_CHELSAv2.1-clt-range\_cell

**filename:** Climate\_CHELSAv2.1-clt-range\_cell.tif

**layername:** evg\_023

**English name:** Annual range of monthly cloud area fraction (%) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Gada mākoņu seguma amplitūda (%) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egv-tools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-clt-range_cell.tif"
layername="evg_023"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-clt-range_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-clt-range_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.24 Climate\_CHELSAv2.1-cmi-max\_cell

**filename:** Climate\_CHELSAv2.1-cmi-max\_cell.tif

**layername:** evg\_024

**English name:** Mean of monthly maximum climate moisture index ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējais ik mēneša maksimālais klimata mitruma indekss ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-cmi-max_cell.tif"
layername="evg_024"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-cmi-max_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-cmi-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.25 Climate\_CHELSAv2.1-cmi-mean\_cell

**filename:** Climate\_CHELSAv2.1-cmi-mean\_cell.tif

**layername:** evg\_025

**English name:** Mean of monthly mean climate moisture index ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējais ik mēneša vidējais klimata mitruma indekss ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-cmi-mean_cell.tif"
layername="egv_025"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-cmi-mean_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/RAW/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-cmi-mean_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.26 Climate\_CHELSAv2.1-cmi-min\_cell

**filename:** Climate\_CHELSAv2.1-cmi-min\_cell.tif

**layername:** egv\_026

**English name:** Mean of monthly minimum climate moisture index ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējais ik mēneša minimālais klimata mitruma indekss ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-cmi-min_cell.tif"
layername="egv_026"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-cmi-min_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-cmi-min_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.27 Climate\_CHELSAv2.1-cmi-range\_cell

**filename:** Climate\_CHELSAv2.1-cmi-range\_cell.tif

**layername:** egv\_027

**English name:** Annual range of monthly climate moisture index ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Gada klimata mitruma indeksa amplitūda ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-cmi-range_cell.tif"
layername="egv_027"

```

```

reading= "./Geodata/2024/CHELSA/Climate_CHELSAv2.1-cmi-range_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-cmi-range_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.28 Climate\_CHELSAv2.1-fcf\_cell

**filename:** Climate\_CHELSAv2.1-fcf\_cell.tif

**layername:** evg\_028

**English name:** Frost change frequency (number of events in which tmin or tmax go above or below 0°C) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Sasalšanas gadījumu biežums (zemākā vai augstākā temperatūra šķērso 0°C) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egv-tools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-fcf_cell.tif"
layername="evg_028"
reading= "./Geodata/2024/CHELSA/Climate_CHELSAv2.1-fcf_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,

```

```

layer_name = layername,
fill_gaps = TRUE,
smooth = TRUE,
smooth_radius_km = 5,
plot_result = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-fcf_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.29 Climate\_CHELSAv2.1-fgd\_cell

**filename:** Climate\_CHELSAv2.1-fgd\_cell.tif

**layername:** egv\_029

**English name:** First day of the growing season (TREELIM) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Veģetācijas sezonas pirmā diena (TREELIM) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-fgd_cell.tif"
layername="egv_029"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-fgd_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path = "./RasterGrids_100m/2024/RAW/",
  file_name = localname,
  layer_name = layername,
  fill_gaps = TRUE,
  smooth = TRUE,
  smooth_radius_km = 5,
  plot_result = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

nosaukums="Climate_CHELSAv2.1-fgd_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.30 Climate\_CHELSAv2.1-gdd0\_cell

**filename:** Climate\_CHELSAv2.1-gdd0\_cell.tif

**layername:** egv\_030

**English name:** Growing degree days temerature sum above 0°C (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Aktīvo temperatūru summa no 0°C (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-gdd0_cell.tif"
layername="egv_030"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-gdd0_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/Raw/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-gdd0_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

```

filename=saglabasanas_cels,
overwrite=TRUE

```

## 6.31 Climate\_CHELSAv2.1-gdd10\_cell

**filename:** Climate\_CHELSAv2.1-gdd10\_cell.tif

**layername:** evg\_031

**English name:** Growing degree days temerature sum above 10°C (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Aktīvo temperatūru summa no 10°C (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-gdd10_cell.tif"
layername="evg_031"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-gdd10_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-gdd10_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.32 Climate\_CHELSAv2.1-gdd5\_cell

**filename:** Climate\_CHELSAv2.1-gdd5\_cell.tif

**layername:** egv\_032

**English name:** Growing degree days temerature sum above 5°C (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Aktīvo temperatūru summa no 5°C (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-gdd5_cell.tif"
layername="egv_032"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-gdd5_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/Raw/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-gdd5_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.33 Climate\_CHELSAv2.1-gddlgd0\_cell

**filename:** Climate\_CHELSAv2.1-gddlgd0\_cell.tif

**layername:** egv\_033

**English name:** Last growing degree day above 0°C (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Veģetācijas sezonas no 0°C pēdējā diena (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-gddlgd0_cell.tif"
layername="egv_033"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-gddlgd0_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/RAW/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-gddlgd0_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.34 Climate\_CHELSAv2.1-gddlgd10\_cell

**filename:** Climate\_CHELSAv2.1-gddlgd10\_cell.tif

**layername:** egv\_034

**English name:** Last growing degree day above 10°C (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Veģētācijas sezonas no 10°C pēdējā diena (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows **CHELSA v2.1**. EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-gddlgd10_cell.tif"
layername="egv_034"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-gddlgd10_cell.tif"

```

```

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path = "./RasterGrids_100m/2024/RAW/",
  file_name = localname,
  layer_name = layername,
  fill_gaps = TRUE,
  smooth = TRUE,
  smooth_radius_km = 5,
  plot_result = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-gddlgd10_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.35 Climate\_CHELSAv2.1-gddlgd5\_cell

**filename:** Climate\_CHELSAv2.1-gddlgd5\_cell.tif

**layername:** egv\_035

**English name:** Last growing degree day above 5°C (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Veģetācijas sezonas no 5°C pēdējā diena (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egv-tools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-gddlgd5_cell.tif"
layername="egv_035"
reading=".~/Geodata/2024/CHELSA/Climate_CHELSAv2.1-gddlgd5_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path = "./RasterGrids_100m/2024/RAW/",
  file_name = localname,
  layer_name = layername,
  fill_gaps = TRUE,
  smooth = TRUE,
  smooth_radius_km = 5,
  plot_result = TRUE)

```

```

print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-gddlgd5_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.36 Climate\_CHELSAv2.1-gdgfd0\_cell

**filename:** Climate\_CHELSAv2.1-gdgfd0\_cell.tif

**layername:** evg\_036

**English name:** First growing degree day above 0°C (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Veģētācijas sezonas no 0°C pirmā diena (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egv-tools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-gdgfd0_cell.tif"
layername="evg_036"
reading "./Geodata/2024/CHELSA/Climate_CHELSAv2.1-gdgfd0_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-gdgfd0_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)

```

```

centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.37 Climate\_CHELSAv2.1-gdgd10\_cell

**filename:** Climate\_CHELSAv2.1-gdgd10\_cell.tif

**layername:** evg\_037

**English name:** First growing degree day above 10°C (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Veģetācijas sezonas no 10°C pirmā diena (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-gdgd10_cell.tif"
layername="evg_037"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-gdgd10_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-gdgd10_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.38 Climate\_CHELSAv2.1-gdgd5\_cell

**filename:** Climate\_CHELSAv2.1-gdgd5\_cell.tif

**layername:** evg\_038

**English name:** First growing degree day above 5°C (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Veģetācijas sezonas no 5°C pirmā diena (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egv-tools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-gdgd5_cell.tif"
layername="evg_038"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-gdgd5_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-gdgd5_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.39 Climate\_CHELSAv2.1-gsl\_cell

**filename:** Climate\_CHELSAv2.1-gsl\_cell.tif

**layername:** evg\_039

**English name:** Length of the growing season (TREELIM) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Veģetācijas sezonas garums (TREELIM) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egv-tools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power

= 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-gsl_cell.tif"
layername="egv_039"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-gsl_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/Raw/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-gsl_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.40 Climate\_CHELSAv2.1-gsp\_cell

**filename:** Climate\_CHELSAv2.1-gsp\_cell.tif

**layername:** egv\_040

**English name:** Accumulated precipitation amount ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) on growing season days (TREELIM) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Veģetācijas sezonā (TREELIM) uzkrātais nokrišņu daudzums ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----
```

```

localname="Climate_CHELSAv2.1-gsp_cell.tif"
layername="egv_040"
reading=".~/Geodata/2024/CHELSA/Climate_CHELSAv2.1-gsp_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/Raw/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-gsp_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.41 Climate\_CHELSAv2.1-gst\_cell

**filename:** Climate\_CHELSAv2.1-gst\_cell.tif

**layername:** egv\_041

**English name:** Mean daily mean air temperature of the growing season (TREELIM) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējā ik dienas vidējā gaisa temperatūra (°C) veģetācijas sezonā (TREELIM) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-gst_cell.tif"
layername="egv_041"
reading=".~/Geodata/2024/CHELSA/Climate_CHELSAv2.1-gst_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/Raw/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-gst_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

```

rawfile_path = reading,
out_path    = "./RasterGrids_100m/2024/RAW/",
file_name   = localname,
layer_name  = layername,
fill_gaps   = TRUE,
smooth      = TRUE,
smooth_radius_km = 5,
plot_result = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-gst_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.42 Climate\_CHELSAv2.1-hurs-max\_cell

**filename:** Climate\_CHELSAv2.1-hurs-max\_cell.tif

**layername:** egv\_042

**English name:** Mean of monthly maximum near-surface relative humidity (%) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējais ik mēneša maksimālais piezemes slāņa gaisa mitrums (%) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egv-tools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-hurs-max_cell.tif"
layername="egv_042"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-hurs-max_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)

```

```

print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-hurs-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.43 Climate\_CHELSAv2.1-hurs-mean\_cell

**filename:** Climate\_CHELSAv2.1-hurs-mean\_cell.tif

**layername:** egv\_043

**English name:** Mean of monthly mean near-surface relative humidity (%) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējais ik mēneša vidējais piezemes slāņa gaisa mitrums (%) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows **CHELSA v2.1**. EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-hurs-mean_cell.tif"
layername="egv_043"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-hurs-mean_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/Raw/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-hurs-mean_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)

```

```

saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.44 Climate\_CHELSAv2.1-hurs-min\_cell

**filename:** Climate\_CHELSAv2.1-hurs-min\_cell.tif

**layername:** evg\_044

**English name:** Mean of monthly minimum near-surface relative humidity (%) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējais ik mēnesā minimālais gaisa mitrums (%) (CHELSA v2.1) analīzēs šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-hurs-min_cell.tif"
layername="evg_044"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-hurs-min_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/Raw/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-hurs-min_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.45 Climate\_CHELSAv2.1-hurs-range\_cell

**filename:** Climate\_CHELSAv2.1-hurs-range\_cell.tif

**layername:** evg\_045

**English name:** Annual range of monthly near-surface relative humidity (%) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Gada gaisa mitruma piezemes slānī amplitūda (%) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-hurs-range_cell.tif"
layername="evg_045"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-hurs-range_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-hurs-range_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.46 Climate\_CHELSAv2.1-lgd\_cell

**filename:** Climate\_CHELSAv2.1-lgd\_cell.tif

**layername:** evg\_046

**English name:** Last day of the growing season (TREELIM) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Pēdējā veģetācijas sezonas diena (TREELIM) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-lgd_cell.tif"
layername="egv_046"
reading= "./Geodata/2024/CHELSA/Climate_CHELSAv2.1-lgd_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-lgd_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.47 Climate\_CHELSAv2.1-ngd0\_cell

**filename:** Climate\_CHELSAv2.1-ngd0\_cell.tif

**layername:** egv\_047

**English name:** Number of days in which air temperature at 2 m is  $> 0^{\circ}\text{C}$  (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Dienu skaits, kurā gaisa temperatūra 2 m augstumā pārsniedz  $0^{\circ}\text{C}$  (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
```

```

# job ----

localname="Climate_CHELSAv2.1-ngd0_cell.tif"
layername="egv_047"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-ngd0_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/Raw/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-ngd0_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.48 Climate\_CHELSAv2.1-ngd10\_cell

**filename:** Climate\_CHELSAv2.1-ngd10\_cell.tif

**layername:** egv\_048

**English name:** Number of days in which air temperature at 2 m is > 10°C (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Dienu skaits, kurā gaisa temperatūra 2 m augstumā pārsniedz 10°C (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-ngd10_cell.tif"
layername="egv_048"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-ngd10_cell.tif"

df <- downscale2egv(

```

```

template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
rawfile_path = reading,
out_path     = "./RasterGrids_100m/2024/RAW/",
file_name    = localname,
layer_name   = layername,
fill_gaps    = TRUE,
smooth       = TRUE,
smooth_radius_km = 5,
plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-ngd10_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.49 Climate\_CHELSAv2.1-ngd5\_cell

**filename:** Climate\_CHELSAv2.1-ngd5\_cell.tif

**layername:** egv\_049

**English name:** Number of days in which air temperature at 2 m is > 5°C (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Dienu skaits, kurā gaisa temperatūra 2 m augstumā pārsniedz 5°C (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egv-tools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-ngd5_cell.tif"
layername="egv_049"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-ngd5_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,

```

```

smooth_radius_km = 5,
plot_result = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-ngd5_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.50 Climate\_CHELSAv2.1-npp\_cell

**filename:** Climate\_CHELSAv2.1-npp\_cell.tif

**layername:** egv\_050

**English name:** Net primary productivity ( $\text{g C m}^{-2} \text{ year}^{-1}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Neto primārā produkcija ( $\text{g C m}^{-2} \text{ year}^{-1}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-npp_cell.tif"
layername="egv_050"
reading "./Geodata/2024/CHELSA/Climate_CHELSAv2.1-npp_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path = "./RasterGrids_100m/2024/RAW/",
  file_name = localname,
  layer_name = layername,
  fill_gaps = TRUE,
  smooth = TRUE,
  smooth_radius_km = 5,
  plot_result = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-npp_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)

```

```

slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.51 Climate\_CHELSAv2.1-pet-penman-max\_cell

**filename:** Climate\_CHELSAv2.1-pet-penman-max\_cell.tif

**layername:** evg\_051

**English name:** Mean of monthly maximum potential evapotranspiration ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējā ik mēneša maksimālā potenciālā evapotranspirācija ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-pet-penman-max_cell.tif"
layername="evg_051"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-pet-penman-max_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-pet-penman-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.52 Climate\_CHELSAv2.1-pet-penman-mean\_cell

**filename:** Climate\_CHELSAv2.1-pet-penman-mean\_cell.tif

**layername:** evg\_052

**English name:** Mean of monthly mean potential evapotranspiration ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējā ik mēneša vidējā potenciālā evapotranspirācija ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-pet-penman-mean_cell.tif"
layername="evg_052"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-pet-penman-mean_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/RAW/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-pet-penman-mean_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.53 Climate\_CHELSAv2.1-pet-penman-min\_cell

**filename:** Climate\_CHELSAv2.1-pet-penman-min\_cell.tif

**layername:** evg\_053

**English name:** Mean of monthly minimum potential evapotranspiration ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējā ik mēneša minimālā potenciālā evapotranspirācija ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-pet-penman-min_cell.tif"
layername="egv_053"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-pet-penman-min_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/RAW/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-pet-penman-min_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.54 Climate\_CHELSAv2.1-pet-penman-range\_cell

**filename:** Climate\_CHELSAv2.1-pet-penman-range\_cell.tif

**layername:** egv\_054

**English name:** Annual range of monthly potential evapotranspiration ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Gada potenciālās evapotranspirācijas amplitūda ( $\text{kg m}^{-2} \text{ month}^{-1}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-pet-penman-range_cell.tif"
layername="egv_054"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-pet-penman-range_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-pet-penman-range_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.55 Climate\_CHELSAv2.1-rsds-max\_cell

**filename:** Climate\_CHELSAv2.1-rsds-max\_cell.tif

**layername:** egv\_055

**English name:** Mean of monthly maximum surface downwelling shortwave flux in air (MJ m<sup>-2</sup> d<sup>-1</sup>) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējā ik mēneša maksimālā Zemes virsmu sasniedzošā saules radiācija (MJ m<sup>-2</sup> d<sup>-1</sup>) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows **CHELSA v2.1**. EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-rsds-max_cell.tif"
layername="egv_055"

```

```

reading= "./Geodata/2024/CHELSA/Climate_CHELSAv2.1-rsds-max_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-rsds-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.56 Climate\_CHELSAv2.1-rsds-mean\_cell

**filename:** Climate\_CHELSAv2.1-rsds-mean\_cell.tif

**layername:** evg\_056

**English name:** Mean of monthly mean surface downwelling shortwave flux in air ( $\text{MJ m}^{-2} \text{ d}^{-1}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējā ik mēneša vidējā Zemes virsmu sasniedzošā saules radiācija ( $\text{MJ m}^{-2} \text{ d}^{-1}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egv-tools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-rsds-mean_cell.tif"
layername="evg_056"
reading= "./Geodata/2024/CHELSA/Climate_CHELSAv2.1-rsds-mean_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,

```

```

layer_name = layername,
fill_gaps = TRUE,
smooth = TRUE,
smooth_radius_km = 5,
plot_result = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-rsds-mean_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.57 Climate\_CHELSAv2.1-rsds-min\_cell

**filename:** Climate\_CHELSAv2.1-rsds-min\_cell.tif

**layername:** evg\_057

**English name:** Mean of monthly minimum surface shortwave flux in air (MJ m<sup>-2</sup> d<sup>-1</sup>) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējā ik mēneša minimālā Zemes virsmu sasniedzošā saules radiācija (MJ m<sup>-2</sup> d<sup>-1</sup>) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-rsds-min_cell.tif"
layername="evg_057"
reading "./Geodata/2024/CHELSA/Climate_CHELSAv2.1-rsds-min_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path = "./RasterGrids_100m/2024/RAW/",
  file_name = localname,
  layer_name = layername,
  fill_gaps = TRUE,
  smooth = TRUE,
  smooth_radius_km = 5,
  plot_result = TRUE)
print(df)

# standardisation ----

```

```

if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-rsds-min_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.58 Climate\_CHELSAv2.1-rsds-range\_cell

**filename:** Climate\_CHELSAv2.1-rsds-range\_cell.tif

**layername:** egv\_058

**English name:** Annual range of daily mean surface downwelling shortwave flux in air (MJ m<sup>-2</sup> d<sup>-1</sup>) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Gada amplitūda ik dienas vidējai Zemes virsmu sasniedzošajai saules radiācijai (MJ m<sup>-2</sup> d<sup>-1</sup>) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-rsds-range_cell.tif"
layername="egv_058"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-rsds-range_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/RAW/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-rsds-range_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)

```

```

centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.59 Climate\_CHELSAv2.1-scd\_cell

**filename:** Climate\_CHELSAv2.1-scd\_cell.tif

**layername:** egv\_059

**English name:** Number of days with snow cover (TREELIM) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Dienu ar sniega segu skaits (TREELIM) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egv-tools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-scd_cell.tif"
layername="egv_059"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-scd_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-scd_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.60 Climate\_CHELSAv2.1-sfcWind-max\_cell

**filename:** Climate\_CHELSAv2.1-sfcWind-max\_cell.tif

**layername:** evg\_060

**English name:** Mean of monthly maximum near-surface wind speed ( $\text{m s}^{-1}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējais ik mēneša maksimālais piezemes slāņa vēja ātrums ( $\text{m s}^{-1}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-sfcWind-max_cell.tif"
layername="evg_060"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-sfcWind-max_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-sfcWind-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.61 Climate\_CHELSAv2.1-sfcWind-mean\_cell

**filename:** Climate\_CHELSAv2.1-sfcWind-mean\_cell.tif

**layername:** evg\_061

**English name:** Mean of monthly mean near-surface wind speed ( $\text{m s}^{-1}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējais ik mēneša vidējais piezemes slāņa vēja ātrums ( $\text{m s}^{-1}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-sfcWind-mean_cell.tif"
layername="egv_061"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-sfcWind-mean_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/RAW/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-sfcWind-mean_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.62 Climate\_CHELSAv2.1-sfcWind-min\_cell

**filename:** Climate\_CHELSAv2.1-sfcWind-min\_cell.tif

**layername:** egv\_062

**English name:** Mean of monthly minimum near-surface wind speed ( $\text{m s}^{-1}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējais ik mēneša minimālais piezemes slāņa vēja ātrums ( $\text{m s}^{-1}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-sfcWind-min_cell.tif"
layername="egv_062"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-sfcWind-min_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/RAW/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-sfcWind-min_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.63 Climate\_CHELSAv2.1-sfcWind-range\_cell

**filename:** Climate\_CHELSAv2.1-sfcWind-range\_cell.tif

**layername:** egv\_063

**English name:** Annual range of monthly mean near-surface wind speed ( $\text{m s}^{-1}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Gada amplitūda ik mēneša vidējam piezemes slāņa vēja ātrumam ( $\text{m s}^{-1}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-sfcWind-range_cell.tif"
layername="egv_063"

```

```

reading= "./Geodata/2024/CHELSA/Climate_CHELSAv2.1-sfcWind-range_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-sfcWind-range_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.64 Climate\_CHELSAv2.1-swb\_cell

**filename:** Climate\_CHELSAv2.1-swb\_cell.tif

**layername:** evg\_064

**English name:** Site water balance ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Ūdens bilance ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows **CHELSA v2.1**. EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-swb_cell.tif"
layername="evg_064"
reading= "./Geodata/2024/CHELSA/Climate_CHELSAv2.1-swb_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,

```

```

smooth_radius_km = 5,
plot_result = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-swb_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.65 Climate\_CHELSAv2.1-swe\_cell

**filename:** Climate\_CHELSAv2.1-swe\_cell.tif

**layername:** evg\_065

**English name:** Snow water equivalent ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Ūdens ekvivalenta sniegā ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-swe_cell.tif"
layername="evg_065"
reading "./Geodata/2024/CHELSA/Climate_CHELSAv2.1-swe_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path = "./RasterGrids_100m/2024/RAW/",
  file_name = localname,
  layer_name = layername,
  fill_gaps = TRUE,
  smooth = TRUE,
  smooth_radius_km = 5,
  plot_result = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-swe_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)

```

```

slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.66 Climate\_CHELSAv2.1-vpd-max\_cell

**filename:** Climate\_CHELSAv2.1-vpd-max\_cell.tif

**layername:** evg\_066

**English name:** Mean of monthly maximum vapor pressure deficit (Pa) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējais ik mēneša maksimālais iztvaikošanas spiediena deficīts (Pa) (CHELSA v2.1) analīzēs šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egv-tools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-vpd-max_cell.tif"
layername="evg_066"
reading="./Geodata/2024/CHELSA/Climate_CHELSAv2.1-vpd-max_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-vpd-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.67 Climate\_CHELSAv2.1-vpd-mean\_cell

**filename:** Climate\_CHELSAv2.1-vpd-mean\_cell.tif

**layername:** evg\_067

**English name:** Mean of monthly mean vapor pressure deficit (Pa) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējais ik mēneša vidējais iztvaikošanas spiediena defīcīts (Pa) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-vpd-mean_cell.tif"
layername="evg_067"
reading=".~/Geodata/2024/CHELSA/Climate_CHELSAv2.1-vpd-mean_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-vpd-mean_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.68 Climate\_CHELSAv2.1-vpd-min\_cell

**filename:** Climate\_CHELSAv2.1-vpd-min\_cell.tif

**layername:** evg\_068

**English name:** Mean of monthly minimum vapor pressure deficit (Pa) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Vidējais ik mēneša minimālais iztvaikošanas spiediena deficitis (Pa) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-vpd-min_cell.tif"
layername="egv_068"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-vpd-min_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path     = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path  = reading,
  out_path      = "./RasterGrids_100m/2024/RAW/",
  file_name     = localname,
  layer_name    = layername,
  fill_gaps     = TRUE,
  smooth        = TRUE,
  smooth_radius_km = 5,
  plot_result   = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-vpd-min_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.69 Climate\_CHELSAv2.1-vpd-range\_cell

**filename:** Climate\_CHELSAv2.1-vpd-range\_cell.tif

**layername:** egv\_069

**English name:** Annual range of monthly mean vapor pressure deficit (Pa) (CHELSA v2.1) within the analysis cell (1 ha)

**Latvian name:** Gada iztvaikošanas spiediena deficitā ik mēneša vidējo amplitūdu (Pa) (CHELSA v2.1) analīzes šūnā (1 ha)

**Procedure:** Directly follows [CHELSA v2.1](#). EGV is prepared using the workflow `egvtools::downscale2egv()` with inverse distance weighted (power = 2) gap filling and soft smoothing (power = 0.5) over 5 km radius around each cell. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# job ----

localname="Climate_CHELSAv2.1-vpd-range_cell.tif"
layername="egv_069"
reading=("./Geodata/2024/CHELSA/Climate_CHELSAv2.1-vpd-range_cell.tif"

df <- downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = reading,
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name    = localname,
  layer_name   = layername,
  fill_gaps    = TRUE,
  smooth       = TRUE,
  smooth_radius_km = 5,
  plot_result  = TRUE)
print(df)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Climate_CHELSAv2.1-vpd-range_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.70 HydroClim\_01-max\_cell

**filename:** HydroClim\_01-max\_cell.tif

**layername:** egv\_070

**English name:** Maximum per subcatchment upstream mean annual air temperature (°C) (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina maksimālā vidējā gaisa temperatūra augstecē (°C) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (max) are calculated (`exactextractr::exact_extract()`), and the results are rasterised with the workflow `egvtools::polygon2input()`. Once rasterised to input data, EGV is created using the workflow `egvtools::input2egv()`. To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
← 12_v1c/hybas_lake_eu_lev12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km,crs=3059)
level12=st_transform(level12,crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_01-max_cell.tif"
layername="egv_070"
summary_function="max"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = localname,
  value_field = "Hydro_values",
  fun="first",
  value_type = "continuous",
  prepare=FALSE,
  project_mode = "auto",
  check_na = FALSE,
  plot_result=FALSE,
  plot_gaps = FALSE,
  overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = localname,
  layername = layername,
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = FALSE)
egvrez

unlink(paste0("./RasterGrids_10m/2024/",localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_01-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,

```

```
overwrite=TRUE)
```

## 6.71 HydroClim\_02-max\_cell

**filename:** HydroClim\_02-max\_cell.tif

**layername:** evg\_071

**English name:** Maximum per subcatchment upstream mean diurnal air temperature range (°C) (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina maksimālā diennakts gaisa temperatūras amplitūda augštečē (°C) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (max) are calculated (`exactextractr::exact_extract()`), and the results are rasterised with the workflow `egvtools::polygon2input()`. Once rasterised to input data, EGV is created using the workflow `egvtools::input2egv()`. To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
  ↳ 12_v1c/hybas_lake_eu_lev12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km, crs=3059)
level12=st_transform(level12, crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_02-max_cell.tif"
layername="evg_071"
summary_function="max"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
              template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
              out_path = "./RasterGrids_10m/2024/",
              file_name = localname,
              value_field = "Hydro_values",
              fun="first",
              value_type = "continuous",
              prepare=FALSE,
              project_mode = "auto",
              check_na = FALSE,
              plot_result=FALSE,
              plot_gaps = FALSE,
              overwrite=TRUE)
```

```

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
  outlocation = "./RasterGrids_100m/2024/RAW/",
  outfilename = localname,
  layername = layername,
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = FALSE)
egvrez

unlink(paste0("./RasterGrids_10m/2024/",localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_02-max_cell.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.72 HydroClim\_03-max\_cell

**filename:** HydroClim\_03-max\_cell.tif

**layername:** egv\_072

**English name:** Maximum per subcatchment upstream isothermality (ratio of diurnal variation to annual variation in temperatures) (°C) (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina maksimālā izotermalitāte (diennakts un gada temperatūru variabilitātes attiecība) augstecē (°C) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (max) are calculated (`exactextractr::exact_extract()`), and the the results are rasterised with the workflow `egvtools::polygon2input()`. Once rasterised to input data, EGV is created using the workflow `egvtools::input2egv()`. To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
 ↴ 12_v1c/hybas_lake_eu_lev12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km,crs=3059)

```

```

level12=st_transform(level12,crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_03-max_cell.tif"
layername="egv_072"
summary_function="max"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
              template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
              out_path = "./RasterGrids_10m/2024/",
              file_name = localname,
              value_field = "Hydro_values",
              fun="first",
              value_type = "continuous",
              prepare=FALSE,
              project_mode = "auto",
              check_na = FALSE,
              plot_result=FALSE,
              plot_gaps = FALSE,
              overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "average",
                  missing_job = "FillOutput",
                  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
                  outlocation = "./RasterGrids_100m/2024/Raw/",
                  outfilename = localname,
                  layername = layername,
                  idw_weight = 2,
                  plot_gaps = FALSE,plot_final = FALSE)
egvrez

unlink(paste0("./RasterGrids_10m/2024/",localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_03-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.73 HydroClim\_04-max\_cell

**filename:** HydroClim\_04-max\_cell.tif

**layername:** egv\_073

**English name:** Maximum per subcatchment upstream temperature seasonality (standard deviation of the monthly mean temperatures) (°C/100) (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina maksimālā temperatūras sezonalitāte (mēneša vidējo temperatūru standartnovirze) augštecē (°C/100) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (max) are calculated (`exactextractr::exact_extract()`), and the results are rasterised with the workflow `egvtools::polygon2input()`. Once rasterised to input data, EGV is created using the workflow `egvtools::input2egv()`. To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
  ↳ 12_v1c/hybas_lake_eu_lev12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km,crs=3059)
level12=st_transform(level12,crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_04-max_cell.tif"
layername="egv_073"
summary_function="max"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = localname,
  value_field = "Hydro_values",
  fun="first",
  value_type = "continuous",
  prepare=FALSE,
  project_mode = "auto",
  check_na = FALSE,
  plot_result=FALSE,
  plot_gaps = FALSE,
  overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = localname,
  layername = layername,
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = FALSE)
egvrez
```

```

unlink(paste0("./RasterGrids_10m/2024/", localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_04-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/", nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/", nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis, fun="mean", na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets, fun="rms", na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.74 HydroClim\_05-max\_cell

**filename:** HydroClim\_05-max\_cell.tif

**layername:** egv\_074

**English name:** Maximum per subcatchment upstream mean daily maximum air temperature (°C) of the warmest month (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina maksimālā augsteces dienas augstākā gaisa temperatūra siltākajā mēnesī (°C) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (max) are calculated (`exactextractr::exact_extract()`), and the results are rasterised with the workflow `egvtools::polygon2input()`. Once rasterised to input data, EGV is created using the workflow `egvtools::input2egv()`. To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
← 12_v1c/hybas_lake_eu_lev12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km, crs=3059)
level12=st_transform(level12, crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_05-max_cell.tif"
layername="egv_074"
summary_function="max"

```

```

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
              template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
              out_path = "./RasterGrids_10m/2024/",
              file_name = localname,
              value_field = "Hydro_values",
              fun="first",
              value_type = "continuous",
              prepare=FALSE,
              project_mode = "auto",
              check_na = FALSE,
              plot_result=FALSE,
              plot_gaps = FALSE,
              overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "average",
                  missing_job = "FillOutput",
                  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
                  outlocation = "./RasterGrids_100m/2024/Raw/",
                  outfilename = localname,
                  layername = layername,
                  idw_weight = 2,
                  plot_gaps = FALSE,plot_final = FALSE)
egvrez

unlink(paste0("./RasterGrids_10m/2024/",localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_05-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.75 HydroClim\_06-min\_cell

**filename:** HydroClim\_06-min\_cell.tif

**layername:** egv\_075

**English name:** Minimum per subcatchment upstream mean daily minimum air temperature (°C) of the coldest month (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina minimālā augšteces dienas zemākā gaisa temperatūra vēsākajā mēnesī (°C) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (min) are calculated (`exactextractr::exact_extract()`), and the results are rasterised with the workflow `egvtools::polygon2input()`. Once rasterised to input data, EGV is created using the workflow `egvtools::input2egv()`. To prevent from gaps at the edges, inverse distance weighted (power = 2) gap

filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
 ↳ 12_v1c/hybas_lake_eu_lev12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km,crs=3059)
level12=st_transform(level12,crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_06-min_cell.tif"
layername="egv_075"
summary_function="min"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
              template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
              out_path = "./RasterGrids_10m/2024/",
              file_name = localname,
              value_field = "Hydro_values",
              fun="first",
              value_type = "continuous",
              prepare=FALSE,
              project_mode = "auto",
              check_na = FALSE,
              plot_result=FALSE,
              plot_gaps = FALSE,
              overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "average",
                  missing_job = "FillOutput",
                  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
                  outlocation = "./RasterGrids_100m/2024/Raw/",
                  outfilename = localname,
                  layername = layername,
                  idw_weight = 2,
                  plot_gaps = FALSE,plot_final = FALSE)
egvrez

unlink(paste0("./RasterGrids_10m/2024/",localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_06-min_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)

```

```

slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.76 HydroClim\_07-max\_cell

**filename:** HydroClim\_07-max\_cell.tif

**layername:** evg\_076

**English name:** Maximum per subcatchment upstream annual range of air temperature (°C) (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina maksimālā augšteces gada gaisa temperatūru amplitūda (°C) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (max) are calculated (`exactextractr::exact_extract()`), and the results are rasterised with the workflow `egvtools::polygon2input()`. Once rasterised to input data, EGV is created using the workflow `egvtools::input2egv()`. To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
 ↳ 12_v1c/hybas_lake_eu_lev12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km,crs=3059)
level12=st_transform(level12,crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_07-max_cell.tif"
layername="evg_076"
summary_function="max"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
    template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
    out_path = "./RasterGrids_10m/2024/",
    file_name = localname,
    value_field = "Hydro_values",
    fun="first",
    value_type = "continuous",
    
```

```

    prepare=FALSE,
    project_mode = "auto",
    check_na = FALSE,
    plot_result=FALSE,
    plot_gaps = FALSE,
    overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = localname,
  layername = layername,
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = FALSE)
egvrez

unlink(paste0("./RasterGrids_10m/2024/",localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_07-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.77 HydroClim\_08-max\_cell

**filename:** HydroClim\_08-max\_cell.tif

**layername:** egv\_077

**English name:** Maximum per subcatchment upstream mean daily mean air temperatures (°C) of the wettest quarter (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina maksimālā augšteces dienas vidējā gaisa temperatūra mitrākajā ceturksnī (°C) (HydroClim) analīzēs šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (max) are calculated (`exactextractr::exact_extract()`), and the results are rasterised with the workflow `egvtools::polygon2input()`. Once rasterised to input data, EGV is created using the workflow `egvtools::input2egv()`. To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}

```

```

if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
← 12_v1c/hybas_lake_eu_lev12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km,crs=3059)
level12=st_transform(level12,crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_08-max_cell.tif"
layername="egv_077"
summary_function="max"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
              template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
              out_path = "./RasterGrids_10m/2024/",
              file_name = localname,
              value_field = "Hydro_values",
              fun="first",
              value_type = "continuous",
              prepare=FALSE,
              project_mode = "auto",
              check_na = FALSE,
              plot_result=FALSE,
              plot_gaps = FALSE,
              overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "average",
                  missing_job = "FillOutput",
                  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
                  outlocation = "./RasterGrids_100m/2024/Raw/",
                  outfilename = localname,
                  layername = layername,
                  idw_weight = 2,
                  plot_gaps = FALSE,plot_final = FALSE)
egvrez

unlink(paste0("./RasterGrids_10m/2024/",localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_08-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.78 HydroClim\_09-min\_cell

**filename:** HydroClim\_09-min\_cell.tif

**layername:** evg\_078

**English name:** Minimum per subcatchment upstream mean daily mean air temperatures (°C) of the driest quarter (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina minimālā augšteces dienas vidējā gaisa temperatūra sausākajā ceturksnī (°C) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (min) are calculated (`exactextractr::exact_extract()`), and the results are rasterised with the workflow `egvtools::polygon2input()`. Once rasterised to input data, EGV is created using the workflow `egvtools::input2egv()`. To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
↳ 12_v1c/hybas_lake_eu_lev12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km,crs=3059)
level12=st_transform(level12,crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_09-min_cell.tif"
layername="evg_078"
summary_function="min"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
              template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
              out_path = "./RasterGrids_10m/2024/",
              file_name = localname,
              value_field = "Hydro_values",
              fun="first",
              value_type = "continuous",
              prepare=FALSE,
              project_mode = "auto",
              check_na = FALSE,
              plot_result=FALSE,
              plot_gaps = FALSE,
              overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "average",
                  missing_job = "FillOutput",
                  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
```

```

        outlocation = "./RasterGrids_100m/2024/RAW/",
        outfilename = localname,
        layername = layername,
        idw_weight = 2,
        plot_gaps = FALSE,plot_final = FALSE)
egvrez

unlink(paste0("./RasterGrids_10m/2024/",localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_09-min_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.79 HydroClim\_10-max\_cell

**filename:** HydroClim\_10-max\_cell.tif

**layername:** egv\_079

**English name:** Maximum per subcatchment upstream mean daily mean air temperatures (°C) of the warmest quarter (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina maksimālā augšteces dienas vidējā gaisa temperatūra siltākajā ceturksnī (°C) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (max) are calculated (`exactextractr::exact_extract()`), and the results are rasterised with the workflow `egvtools::polygon2input()`. Once rasterised to input data, EGV is created using the workflow `egvtools::input2egv()`. To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
← 12_v1c/hybas_lake_eu_lev12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km,crs=3059)
level12=st_transform(level12,crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

```

```

# job ----

localname="HydroClim_10-max_cell.tif"
layername="egv_079"
summary_function="max"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
              template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
              out_path = "./RasterGrids_10m/2024/",
              file_name = localname,
              value_field = "Hydro_values",
              fun="first",
              value_type = "continuous",
              prepare=FALSE,
              project_mode = "auto",
              check_na = FALSE,
              plot_result=FALSE,
              plot_gaps = FALSE,
              overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "average",
                  missing_job = "FillOutput",
                  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
                  outlocation = "./RasterGrids_100m/2024/Raw/",
                  outfilename = localname,
                  layername = layername,
                  idw_weight = 2,
                  plot_gaps = FALSE,plot_final = FALSE)
egvrez

unlink(paste0("./RasterGrids_10m/2024/",localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_10-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.80 HydroClim\_11-min\_cell

**filename:** HydroClim\_11-min\_cell.tif

**layername:** egv\_080

**English name:** Minimum per subcatchment upstream mean daily mean air temperatures (°C) of the coldest quarter (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina minimālā augsteces dienas vidējā gaisa temperatūra vēsākajā ceturksnī (°C) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (min) are calculated (`exactextractr::exact_extract()`), and the results are rasterised with the workflow `egvtools::polygon2input()`. Once rasterised to input data, EGV is created using the workflow `egvtools::input2egv()`. To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
← 12_v1c/hybas_lake_eu_lev12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km,crs=3059)
level12=st_transform(level12,crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_11-min_cell.tif"
layername="egv_080"
summary_function="min"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
              template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
              out_path = "./RasterGrids_10m/2024/",
              file_name = localname,
              value_field = "Hydro_values",
              fun="first",
              value_type = "continuous",
              prepare=FALSE,
              project_mode = "auto",
              check_na = FALSE,
              plot_result=FALSE,
              plot_gaps = FALSE,
              overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "average",
                  missing_job = "FillOutput",
                  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
                  outlocation = "./RasterGrids_100m/2024/Raw/",
                  outfilename = localname,
                  layername = layername,
                  idw_weight = 2,
                  plot_gaps = FALSE,plot_final = FALSE)
egvrez

unlink(paste0("./RasterGrids_10m/2024/",localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}

```

```

if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_11-min_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.81 HydroClim\_12-max\_cell

**filename:** HydroClim\_12-max\_cell.tif

**layername:** evg\_081

**English name:** Maximum per subcatchment upstream annual precipitation amount ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina maksimālais augšteces nokrišņu daudzums gadā ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (max) are calculated (`exactextractr::exact_extract()`), and the results are rasterised with the workflow `egvtools::polygon2input()`. Once rasterised to input data, EGV is created using the workflow `egvtools::input2egv()`. To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
 ↳ 12_v1c/hybas_lake_eu_lev12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km,crs=3059)
level12=st_transform(level12,crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_12-max_cell.tif"
layername="evg_081"
summary_function="max"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
              template_path = "./Templates/TemplateRasters/LV10m_10km.tif",

```

```

out_path = "./RasterGrids_10m/2024/",
file_name = localname,
value_field = "Hydro_values",
fun="first",
value_type = "continuous",
prepare=FALSE,
project_mode = "auto",
check_na = FALSE,
plot_result=FALSE,
plot_gaps = FALSE,
overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
summary_function = "average",
missing_job = "FillOutput",
input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
outlocation = "./RasterGrids_100m/2024/Raw/",
outfile = localname,
layername = layername,
idw_weight = 2,
plot_gaps = FALSE,plot_final = FALSE)
egvrez

unlink(paste0("./RasterGrids_10m/2024/",localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_12-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
filename=saglabasanas_cels,
overwrite=TRUE)

```

## 6.82 HydroClim\_13-max\_cell

**filename:** HydroClim\_13-max\_cell.tif

**layername:** egv\_082

**English name:** Maximum per subcatchment upstream precipitation amount ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) of the wettest month (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina maksimālais augšteces nokrišņu daudzums mitrākajā mēnesī ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (max) are calculated (`exactextractr::exact_extract()`), and the results are rasterised with the workflow `egvtools::polygon2input()`. Once rasterised to input data, EGV is created using the workflow `egvtools::input2egv()`. To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
```

```

if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
↳ 12_v1c/hybas_lake_eu_lev12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km,crs=3059)
level12=st_transform(level12,crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_13-max_cell.tif"
layername="egv_082"
summary_function="max"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = localname,
  value_field = "Hydro_values",
  fun="first",
  value_type = "continuous",
  prepare=FALSE,
  project_mode = "auto",
  check_na = FALSE,
  plot_result=FALSE,
  plot_gaps = FALSE,
  overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = localname,
  layername = layername,
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = FALSE)
egvrez

unlink(paste0("./RasterGrids_10m/2024/",localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_13-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]

```

```

writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.83 HydroClim\_14-max\_cell

**filename:** HydroClim\_14-max\_cell.tif

**layername:** egv\_083

**English name:** Maximum per subcatchment upstream precipitation amount ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) of the driest month (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina maksimālais augšteces nokrišņu daudzums sausākajā mēnesī ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (max) are calculated (`exactextractr::exact_extract()`), and the results are rasterised with the workflow `egvtools::polygon2input()`. Once rasterised to input data, EGV is created using the workflow `egvtools::input2egv()`. To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
↳ 12_v1c/hybas_lake_eu_lev12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km,crs=3059)
level12=st_transform(level12,crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_14-max_cell.tif"
layername="egv_083"
summary_function="max"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = localname,
  value_field = "Hydro_values",
  fun="first",
  value_type = "continuous",
  prepare=FALSE,
  project_mode = "auto",
  check_na = FALSE,
  plot_result=FALSE,
  plot_gaps = FALSE,

```

```

    overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = localname,
  layername = layername,
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = FALSE)
egvrez

unlink(paste0("./RasterGrids_10m/2024/",localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_14-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.84 HydroClim\_15-max\_cell

**filename:** HydroClim\_15-max\_cell.tif

**layername:** egv\_084

**English name:** Maximum per subcatchment upstream precipitation seasonality ( $\text{kg m}^{-2}$ ) (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina maksimālais augšteces nokrišņu daudzuma sezonalitāte ( $\text{kg m}^{-2}$ ) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (max) are calculated (`exactextractr::exact_extract()`), and the results are rasterised with the workflow `egvtools::polygon2input()`. Once rasterised to input data, EGV is created using the workflow `egvtools::input2egv()`. To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
 ↴ 12_v1c/hybas_lake_eu_lev12_v1c.shp")

```

```

grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km,crs=3059)
level12=st_transform(level12,crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_15-max_cell.tif"
layername="egv_084"
summary_function="max"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
              template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
              out_path = "./RasterGrids_10m/2024/",
              file_name = localname,
              value_field = "Hydro_values",
              fun="first",
              value_type = "continuous",
              prepare=FALSE,
              project_mode = "auto",
              check_na = FALSE,
              plot_result=FALSE,
              plot_gaps = FALSE,
              overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "average",
                  missing_job = "FillOutput",
                  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
                  outlocation = "./RasterGrids_100m/2024/Raw/",
                  outfilename = localname,
                  layername = layername,
                  idw_weight = 2,
                  plot_gaps = FALSE,plot_final = FALSE)

egvrez

unlink(paste0("./RasterGrids_10m/2024/",localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_15-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.85 HydroClim\_16-max\_cell

**filename:** HydroClim\_16-max\_cell.tif

**layername:** egv\_085

**English name:** Maximum per subcatchment upstream mean monthly precipitation amount ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) of the wettest quarter (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina maksimālais augšteces mēneša vidējais nokrišņu daudzums mitrākajā ceturksnī ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (max) are calculated (`exactextractr::exact_extract()`), and the results are rasterised with the workflow `egvtools::polygon2input()`. Once rasterised to input data, EGV is created using the workflow `egvtools::input2egv()`. To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
 ↳ 12_v1c/hybas_lake_eu_lev12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km,crs=3059)
level12=st_transform(level12,crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_16-max_cell.tif"
layername="egv_085"
summary_function="max"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
              template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
              out_path = "./RasterGrids_10m/2024/",
              file_name = localname,
              value_field = "Hydro_values",
              fun="first",
              value_type = "continuous",
              prepare=FALSE,
              project_mode = "auto",
              check_na = FALSE,
              plot_result=FALSE,
              plot_gaps = FALSE,
              overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "average",
                  missing_job = "FillOutput",
                  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
                  outlocation = "./RasterGrids_100m/2024/RAW/",
                  outfilename = localname,
                  layername = layername,
                  idw_weight = 2,
```

```

    plot_gaps = FALSE, plot_final = FALSE)
egvrez

unlink(paste0("./RasterGrids_10m/2024/", localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_16-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/", nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/", nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis, fun="mean", na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets, fun="rms", na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.86 HydroClim\_17-max\_cell

**filename:** HydroClim\_17-max\_cell.tif

**layername:** egv\_086

**English name:** Maximum per subcatchment upstream mean monthly precipitation amount ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) of the driest quarter (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina maksimālais augšteces mēneša vidējais nokrišņu daudzums sausākajā ceturksnī ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (max) are calculated (exactextractr::exact\_extract()), and the results are rasterised with the workflow egvtools::polygon2input(). Once rasterised to input data, EGV is created using the workflow egvtools::input2egv(). To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_leve01-
 ↳ 12_v1c/hybas_lake_eu_leve12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km, crs=3059)
level12=st_transform(level12, crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_17-max_cell.tif"
layername="egv_086"

```

```

summary_function="max"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
              template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
              out_path = "./RasterGrids_10m/2024/",
              file_name = localname,
              value_field = "Hydro_values",
              fun="first",
              value_type = "continuous",
              prepare=FALSE,
              project_mode = "auto",
              check_na = FALSE,
              plot_result=FALSE,
              plot_gaps = FALSE,
              overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "average",
                  missing_job = "FillOutput",
                  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
                  outlocation = "./RasterGrids_100m/2024/Raw/",
                  outfilename = localname,
                  layername = layername,
                  idw_weight = 2,
                  plot_gaps = FALSE,plot_final = FALSE)
egvrez

unlink(paste0("./RasterGrids_10m/2024/",localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_17-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.87 HydroClim\_18-max\_cell

**filename:** HydroClim\_18-max\_cell.tif

**layername:** egv\_087

**English name:** Maximum per subcatchment upstream mean monthly precipitation amount ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) of the warmest quarter (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina maksimālais augšteces mēneša vidējais nokrišņu daudzums siltākajā ceturksnī ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (max) are calculated (`exactextractr::exact_extract()`), and the results are rasterised with the

workflow `egvtools::polygon2input()`. Once rasterised to input data, EGV is created using the workflow `egvtools::input2egv()`. To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
← 12_v1c/hybas_lake_eu_lev12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km,crs=3059)
level12=st_transform(level12,crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_18-max_cell.tif"
layername="egv_087"
summary_function="max"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = localname,
  value_field = "Hydro_values",
  fun="first",
  value_type = "continuous",
  prepare=FALSE,
  project_mode = "auto",
  check_na = FALSE,
  plot_result=FALSE,
  plot_gaps = FALSE,
  overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = localname,
  layername = layername,
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = FALSE)
egvrez

unlink(paste0("./RasterGrids_10m/2024/",localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_18-max_cell.tif"
```

```

ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.88 HydroClim\_19-max\_cell

**filename:** HydroClim\_19-max\_cell.tif

**layername:** evg\_088

**English name:** Maximum per subcatchment upstream mean monthly precipitation amount ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) of the coldest quarter (HydroClim) within the analysis cell (1 ha)

**Latvian name:** Sateces apakšbaseina maksimālais augšteces mēneša vidējais nokrišņu daudzums vēsākajā ceturksnī ( $\text{kg m}^{-2} \text{ year}^{-1}$ ) (HydroClim) analīzes šūnā (1 ha)

**Procedure:** Information from the [HydroClim data](#) - including both basin and raster layers - is used. First, basin CRS is transformed to EPSG:3059. Then, zonal statistics (per basin) using a layer specific summary function (max) are calculated (exactextractr::exact\_extract()), and the results are rasterised with the workflow egvtools::polygon2input(). Once rasterised to input data, EGV is created using the workflow egvtools::input2egv(). To prevent from gaps at the edges, inverse distance weighted (power = 2) gap filling is implemented. To save disk space, the intermediate input layer is unlinked. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(exactextractr)) {install.packages("exactextractr"); require(exactextractr)}

# basins ----
level12=st_read("./Geodata/2024/HydroClim/hybas_lake_eu_lev01-
  ↳ 12_v1c/hybas_lake_eu_lev12_v1c.shp")
grid_1km=sfarrow::st_read_parquet("./Templates/TemplateGrids/tikls1km_sauzeme.parquet")
grid_1km=st_transform(grid_1km,crs=3059)
level12=st_transform(level12,crs=3059)
level12=level12[grid_1km,,]

level12=st_make_valid(level12)

# job ----

localname="HydroClim_19-max_cell.tif"
layername="evg_088"
summary_function="max"

slanis=rast(paste0("./Geodata/2024/HydroClim/",localname))
level12$Hydro_values=exact_extract(slanis,level12,fun=summary_function)

polygon2input(vector_data = level12,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = localname,
  value_field = "Hydro_values",

```

```

    fun="first",
    value_type = "continuous",
    prepare=FALSE,
    project_mode = "auto",
    check_na = FALSE,
    plot_result=FALSE,
    plot_gaps = FALSE,
    overwrite=TRUE)

egvrez=input2egv(input=paste0("./RasterGrids_10m/2024/",localname),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  input_template = "./Templates/TemplateRasters/LV10m_10km.tif",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = localname,
  layername = layername,
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = FALSE)
egvrez

unlink(paste0("./RasterGrids_10m/2024/",localname))

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="HydroClim_19-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.89 Distance\_Builtup\_cell

**filename:** Distance\_Builtup\_cell.tif

**layername:** egv\_089

**English name:** Distance to Built-Up features, average within the analysis cell (1 ha)

**Latvian name:** Attālums līdz apbūvei, vidējais analīzes šūnā (1 ha)

**Procedure:** Derived from the [Landscape classification](#), with class 500 reclassified as 1 and others as 0. Processed using the workflow `egvtools::distance2egv()`. To prevent potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Distance_Builtup_cell.tif egv_89 ----
simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")
builtup=ifel(simple_landscape==500,1,0)
plot(builtup)
distegv=distance2egv(input = builtup,
  template_egv = template100,

```

```

    values_as_one = 1,
    fill_gaps = TRUE, idw_weight = 2,
    outlocation = "RasterGrids_100m/2024/RAW/",
    outfilename = "Distance_Builtup_cell.tif",
    layername = "egv_089")
distegv
plot(rast("RasterGrids_100m/2024/RAW/Distance_Builtup_cell.tif"))
rm(builtup)
rm(distegv)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Distance_Builtup_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.90 Distance\_ForestInside\_cell

**filename:** Distance\_ForestInside\_cell.tif

**layername:** egv\_090

**English name:** Distance to Forest Edge Inside Forests, average within the analysis cell (1 ha)

**Latvian name:** Attālums līdz meža malai tā iekšienē, vidējais analīzes šūnā (1 ha)

**Procedure:** Derived from the [Landscape classification](#), with values in a range from 630 to 700 reclassified as 0 and others as 1. Processed using the workflow `egvtools::distance2egv()`. To prevent potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Distance_ForestInside_cell.tif    egv_90 ----
simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")
trees_inside=ifel(simple_landscape>=630&simple_landscape<700,0,1)
plot(trees_inside)
distegv=distance2egv(input = trees_inside,
                      template_egv = template100,
                      values_as_one = 1,
                      fill_gaps = TRUE, idw_weight = 2,
                      outlocation = "RasterGrids_100m/2024/RAW/",
                      outfilename = "Distance_ForestInside_cell.tif",
                      layername = "egv_090")
distegv
plot(rast("RasterGrids_100m/2024/RAW/Distance_ForestInside_cell.tif"))
rm(trees_inside)
rm(distegv)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

nosaukums="Distance_ForestInside_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.91 Distance\_GrasslandPermanent\_cell

**filename:** Distance\_GrasslandPermanent\_cell.tif

**layername:** egv\_091

**English name:** Distance to Permanent Grasslands, average within the analysis cell (1 ha)

**Latvian name:** Attālums līdz ilggadīgiem zālājiem, vidējais analīzes šūnā (1 ha)

**Procedure:** Derived from the [Rural Support Service's information on declared fields](#) where PRODUCT\_CODE=="710" classified as 1 and the rest of the country as 0. Processed using the workflow `egv-tools::distance2egv()`. To prevent potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")

rastra_pamatne=raster(template10)

# Distance_GrasslandPermanent_cell.tif egv_91 ----
kodes=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
permgrass=lad %>%
  filter(PRODUCT_CODE=="710") %>%
  mutate(yes=1)
permgrass_r=fasterize(permgrass,rastra_pamatne,field="yes",fun="first")
permgrass_t=rast(permgrass_r)
permgrass_t2=cover(permgrass_t,nulls10)
plot(permgrass_t2)
distegv=distance2egv(input = permgrass_t2,
                      template_egv = template10,
                      values_as_one = 1,
                      fill_gaps = TRUE, idw_weight = 2,
                      outlocation = "RasterGrids_100m/2024/RAW/",
                      outfilename = "Distance_GrasslandPermanent_cell.tif",
                      layername = "egv_091")
distegv

```

```

plot(rast("RasterGrids_100m/2024/RAW/Distance_GrasslandPermanent_cell.tif"))
rm(distegv)
rm(kodes)
rm(lad)
rm(permgrass)
rm(permgrass_r)
rm(permgrass_t)
rm(permgrass_t2)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Distance_GrasslandPermanent_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.92 Distance\_Landfill\_cell

**filename:** Distance\_Landfill\_cell.tif

**layername:** evg\_092

**English name:** Distance to Landfills, average within the analysis cell (1 ha)

**Latvian name:** Attālums līdz atkritumu poligoniem, vidējais analīzes šūnā (1 ha)

**Procedure:** Directly follows [Waste and garbage disposal sites, landfills](#).

1. From the [attached file](#), read sheet “Poligoni”;
2. Create an sf object (EPSG:3059);
3. Rasterize and cover so that cells of interest are 1 and others are 0;
4. Create an EGV using the workflow `egvtools::distance2egv()`. Expect warning regarding nothing to do with aggregation. This occurs because `egvtools::distance2egv()` already operate at EGV template not the input template resolution. To prevent potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# Distance_Landfill_cell.tif evg_92 ----

# reading coordinates
landfills=read_excel("./Geodata/2024/GarbageWasteLandfills/Atkritumi.xlsx",sheet="Poligoni")

```

```

#sf object
landfills_sf=st_as_sf(landfills,coords=c("X","Y"),crs=3059)
# rasterize
landfills_rast=rasterize(landfills_sf,template100)
# raster to 1=Cell of interest, 0=background
landfills_bg=cover(landfills_rast,nulls100)

# create an egv
distegv=distance2egv(input = landfills_bg,
    template_egv = template100,
    values_as_one = 1,
    fill_gaps = TRUE, idw_weight = 2,
    outlocation = "RasterGrids_100m/2024/Raw//",
    outfilename = "Distance_Landfill_cell.tif",
    layername = "egv_092")
distegv

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Distance_Landfill_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw//",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled//",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.93 Distance\_Sea\_cell

**filename:** Distance\_Sea\_cell.tif

**layername:** egv\_093

**English name:** Distance to Sea, average within the analysis cell (1 ha)

**Latvian name:** Attālums līdz jūrai, vidējais analīzes šūnā (1 ha)

**Procedure:** Directly follows Latvian Exclusive Economic Zone polygon.

1. Read layer as sf object (it already is EPSG:3059);
2. Rasterize and cover so that cells of interest are 1 and others are 0;
3. Create an egv with the workflow egvtools::distance2egv(). The {fasterize} package does not write CRS with WKT from the EPSG-string; therefore, it is better to use project\_to\_template\_input=TRUE and define input-template. However, the only difference is in how the CRS is stored, therefore this can be ignored - distance will be calculated on the input CRS and only resulting layer will be projected to match EGV template (faster due to 10x aggregation of resolution). To protect against possible data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

```

```

# templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
rastrs10=raster::raster(template10)

# Distance_Sea_cell.tif evg_93 ----

# sea layer, sf
sea=st_read("./Geodata/2024/LV_EEZ/LV_EEZ.shp")

# quick rasterisation
sea_r=fasterize(sea,rastrs10,field="LV_EEZ")
sea_rast=rast(sea_r)

# raster to 1=Cell of interest, 0=background
sea_bg=cover(sea_rast,nulls10)

# create an evg
distegv=distance2egv(input = sea_bg,
                      template_egv = "./Templates/TemplateRasters/LV100m_10km.tif",
                      values_as_one = 1,
                      project_to_template_input=TRUE, # fasterize stores CRS differently
                      template_input=template10,
                      fill_gaps = TRUE, idw_weight = 2,
                      outlocation = "RasterGrids_100m/2024/RAW/",
                      outfilename = "Distance_Sea_cell.tif",
                      layername = "evg_093")
distegv

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Distance_Sea_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.94 Distance\_Trees\_cell

**filename:** Distance\_Trees\_cell.tif

**layername:** evg\_094

**English name:** Distance to Trees, average within the analysis cell (1 ha)

**Latvian name:** Attālums līdz kokiem, vidējais analīzes šūnā (1 ha)

**Procedure:** Derived from the [Landscape classification](#), with values in a range from 630 to 700 reclassified as 1 and all others as 0. Processed using the workflow `egvtools::distance2egv()`. To prevent possible data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
```

```

if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Distance_Trees_cell.tif  egv_94 ----
simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")
trees=ifel(simple_landscape>=630&simple_landscape<700,1,0)
plot(trees)
distegv=distance2egv(input = trees,
                      template_egv = template100,
                      values_as_one = 1,
                      fill_gaps = TRUE, idw_weight = 2,
                      outlocation = "RasterGrids_100m/2024/Raw/",
                      outfilename = "Distance_Trees_cell.tif",
                      layername = "egv_094")
distegv
plot(rast("RasterGrids_100m/2024/Raw/Distance_Trees_cell.tif"))
rm(trees)
rm(distegv)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Distance_Trees_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.95 Distance\_Waste\_cell

**filename:** Distance\_Waste\_cell.tif

**layername:** egv\_095

**English name:** Distance to Waste disposal sites, average within the analysis cell (1 ha)

**Latvian name:** Attālums līdz atkritumu šķirošanas un uzglabāšanas vietām, vidējais analīzes šūnā (1 ha)

**Procedure:** Directly follows [Waste and garbage disposal sites, landfills](#).

1. From the [attached file](#), read sheet “AtkritumuVietas” and clean names;
2. Create an sf object (EPSG:3059);
3. Filter to non-deposit collection locations;
4. Rasterize and cover so that cells of interest are 1 and others are 0;
5. Create an EGV using the workflow `egvtools::distance2egv()`. Expect warning regarding nothing to do with aggregation. That is because `egvtools::distance2egv()` already operate at EGV template not the input template resolution. To prevent possible data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented.
6. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# Distance_Waste_cell.tif egv_95 ----

# reading coordinates
waste=read_excel("./Geodata/2024/GarbageWasteLandfills/Atkritumi.xlsx",sheet=_
  ↪ "AtkritumuVietas")
# cleaning names
waste2=janitor::clean_names(waste)
#sf object
waste_sf=st_as_sf(waste2,coords=c("y_koordinata_lks92_tm","x_koordinata_lks92_tm"),crs=3059)
# filtering to non-deposit
table(waste_sf$pienemsanas_vietas_tips)
waste_sf2=waste_sf %>%
  filter(!str_detect(pienemsanas_vietas_tips,"Depozīta"))
# rasterize
waste_rast=rasterize(waste_sf2,template100)
# raster to 1=Cell of interest, 0=background
wastw_bg=cover(waste_rast,nulls100)

# create an egv
distegv=distance2egv(input = wastw_bg,
  template_egv = template100,
  values_as_one = 1,
  fill_gaps = TRUE, idw_weight = 2,
  outlocation = "RasterGrids_100m/2024/Raw/",
  outfilename = "Distance_Waste_cell.tif",
  layername = "egv_095")
distegv

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Distance_Waste_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.96 Distance\_Water\_cell

**filename:** Distance\_Water\_cell.tif

**layername:** egv\_096

**English name:** Distance to Waterbodies, average within the analysis cell (1 ha)

**Latvian name:** Attālums līdz ūdenstilpēm, vidējais analīzes šūnā (1 ha)

**Procedure:** Derived from the [Landscape classification](#), with class 200 reclassified as 1 and all others as 0. Processed using the workflow `egvtools::distance2egv()`. To prevent possible data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Distance_Water_cell.tif  egv_96 ----
simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")
water=ifel(simple_landscape==200,1,0)
plot(water)
distegv=distance2egv(input = water,
                      template_egv = template100,
                      values_as_one = 1,
                      fill_gaps = TRUE, idw_weight = 2,
                      outlocation = "RasterGrids_100m/2024/RAW/",
                      outfilename = "Distance_Water_cell.tif",
                      layername = "egv_096")

distegv
plot(rast("RasterGrids_100m/2024/RAW/Distance_Water_cell.tif"))
rm(water)
rm(distegv)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Distance_Water_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)
```

## 6.97 Distance\_WaterInside\_cell

**filename:** Distance\_WaterInside\_cell.tif

**layername:** egv\_097

**English name:** Distance to Waterbody Edge Inside Waterbody, average within the analysis cell (1 ha)

**Latvian name:** Attālums līdz ūdenstilpes malai tās iekšienē, vidējais analīzes šūnā (1 ha)

**Procedure:** Derived from the [Landscape classification](#), with class 200 reclassified as 0 and all others as 1. Processed using the workflow `egvtools::distance2egv()`. To prevent possible data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Distance_WaterInside_cell.tif  egv_97 ----
```

```

simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")
water_outside=ifel(simple_landscape==200,0,1)
plot(water_outside)
distegv=distance2egv(input = water_outside,
                      template_egv = template100,
                      values_as_one = 1,
                      fill_gaps = TRUE, idw_weight = 2,
                      outlocation = "RasterGrids_100m/2024/RAW/",
                      outfilename = "Distance_WaterInside_cell.tif",
                      layername = "egv_097")
distegv
plot(rast("RasterGrids_100m/2024/RAW/Distance_WaterInside_cell.tif"))
rm(water_outside)
rm(distegv)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Distance_WaterInside_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.98 Diversity\_Farmland\_r500

**filename:** Diversity\_Farmland\_r500.tif

**layername:** egv\_098

**English name:** Average farmland class  $\alpha$ -diversity of 500 m grid cells within the 0.5 km landscape

**Latvian name:** Vidējā lauku ainavas klašu 500 m šūnu  $\alpha$ -daudzveidība 0,5 km ainavā

**Procedure:** Derived from the [Landscape diversity](#), more precisely [Farmland diversity](#). The average value of 25 ha cells diversity index values is calculated using the workflow `egvtools::radius_function()`. To prevent possible data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. File is written twice, to ensure layername. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii
radius_function(
  kvadратi_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_500m/2024/Diversity_Farmland_500x.tif"),
  layer_prefixes = c("Diversity_Farmland"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",

```

```

n_workers    = 12,
radii        = c("r500"),
radius_mode  = "sparse",
extract_fun   = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 5 * 1024^3)

# Diversity_Farmland_r500.tif  egv_98
slanis=rast("./RasterGrids_100m/2024/RAW/Diversity_Farmland_r500.tif")
names(slanis)="egv_098"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Diversity_Farmland_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Diversity_Farmland_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.99 Diversity\_Farmland\_r1250

**filename:** Diversity\_Farmland\_r1250.tif

**layername:** egv\_099

**English name:** Average farmland class  $\alpha$ -diversity of 500 m grid cells within the 1.25 km landscape

**Latvian name:** Vidējā lauku ainavas klašu 500 m šūnu  $\alpha$ -daudzveidība 1,25 km ainavā

**Procedure:** Derived from the [Landscape diversity](#), more precisely [Farmland diversity](#). The average value of 25 ha cells diversity index values is calculated using the workflow `egvtools::radius_function()`. To prevent possible data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. File is written twice, to ensure layername. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii
radius_function(
  kvadратi_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_500m/2024/Diversity_Farmland_500x.tif"),
  layer_prefixes = c("Diversity_Farmland"))

```

```

output_dir  = "./RasterGrids_100m/2024/RAW/",
n_workers   = 12,
radii       = c("r1250"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight  = 2,
future_max_size = 5 * 1024^3)

# Diversity_Farmland_r1250.tif  egv_99
slanis=rast("./RasterGrids_100m/2024/RAW/Diversity_Farmland_r1250.tif")
names(slanis)="egv_099"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Diversity_Farmland_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Diversity_Farmland_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.100 Diversity\_Farmland\_r3000

**filename:** Diversity\_Farmland\_r3000.tif

**layername:** egv\_100

**English name:** Average farmland class  $\alpha$ -diversity of 500 m grid cells within the 3 km landscape

**Latvian name:** Vidējā lauku ainavas klašu 500 m šūnu  $\alpha$ -daudzveidība 3 km ainavā

**Procedure:** Derived from the [Landscape diversity](#), more precisely [Farmland diversity](#). The average value of 25 ha cells diversity index values is calculated using the workflow `egvtools::radius_function()`. To prevent possible data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. File is written twice, to ensure layername. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   = c("./RasterGrids_500m/2024/Diversity_Farmland_500x.tif"),

```

```

layer_prefixes = c("Diversity_Farmland"),
output_dir    = "./RasterGrids_100m/2024/RAW/",
n_workers     = 12,
radii         = c("r3000"),
radius_mode   = "sparse",
extract_fun   = "mean",
fill_missing   = TRUE,
IDW_weight    = 2,
future_max_size = 5 * 1024^3)

# Diversity_Farmland_r3000.tif  egv_100
slanis=rast("./RasterGrids_100m/2024/RAW/Diversity_Farmland_r3000.tif")
names(slanis)="egv_100"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Diversity_Farmland_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Diversity_Farmland_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.101 Diversity\_Farmland\_r10000

**filename:** Diversity\_Farmland\_r10000.tif

**layername:** egv\_101

**English name:** Average farmland class  $\alpha$ -diversity of 500 m grid cells within the 10 km landscape

**Latvian name:** Vidējā lauku ainavas klašu 500 m šūnu  $\alpha$ -daudzveidība 10 km ainavā

**Procedure:** Derived from the [Landscape diversity](#), more precisely [Farmland diversity](#). The average value of 25 ha cells diversity index values is calculated using the workflow `egvtools::radius_function()`. To prevent possible data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. File is written twice, to ensure layername. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",

```

```

input_layers = c("./RasterGrids_500m/2024/Diversity_Farmland_500x.tif"),
layer_prefixes = c("Diversity_Farmland"),
output_dir = "./RasterGrids_100m/2024/RAW/",
n_workers = 12,
radii = c("r10000"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 5 * 1024^3)

# Diversity_Farmland_r10000.tif egv_101
slanis=rast("./RasterGrids_100m/2024/RAW/Diversity_Farmland_r10000.tif")
names(slanis)="egv_101"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Diversity_Farmland_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Diversity_Farmland_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.102 Diversity\_Forest\_r500

**filename:** Diversity\_Forest\_r500.tif

**layername:** egv\_102

**English name:** Average forest class  $\alpha$ -diversity of 500 m grid cells within the 0.5 km landscape

**Latvian name:** Vidējā mežu ainavas klasu 500 m šūnu  $\alpha$ -daudzveidība 0,5 km ainavā

**Procedure:** Derived from the [Landscape diversity](#), more precisely [Forest diversity](#). The average value of 25 ha cells diversity index values is calculated using the workflow `egvtools::radius_function()`. To prevent possible data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. File is written twice, to ensure layername. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",

```

```

template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers  = c("./RasterGrids_500m/2024/Diversity_Forests_500x.tif"),
layer_prefixes = c("Diversity_Forest"),
output_dir    = "./RasterGrids_100m/2024/Raw//",
n_workers     = 12,
radii         = c("r500"),
radius_mode   = "sparse",
extract_fun   = "mean",
fill_missing   = TRUE,
IDW_weight   = 2,
future_max_size = 5 * 1024^3)

# Diversity_Forest_r500.tif egv_102
slanis=rast("./RasterGrids_100m/2024/Raw/Diversity_Forest_r500.tif")
names(slanis)="egv_102"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/Raw/Diversity_Forest_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Diversity_Forest_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.103 Diversity\_Forest\_r1250

**filename:** Diversity\_Forest\_r1250.tif

**layername:** egv\_103

**English name:** Average forest class  $\alpha$ -diversity of 500 m grid cells within the 1.25 km landscape

**Latvian name:** Vidējā mežu ainavas klašu 500 m šūnu  $\alpha$ -daudzveidība 1,25 km ainavā

**Procedure:** Derived from the [Landscape diversity](#), more precisely [Forest diversity](#). The average value of 25 ha cells diversity index values is calculated using the workflow `egvtools::radius_function()`. To prevent possible data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. File is written twice, to ensure layername. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  
```

```

tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers = c("./RasterGrids_500m/2024/Diversity_Forests_500x.tif"),
layer_prefixes = c("Diversity_Forest"),
output_dir = "./RasterGrids_100m/2024/Raw//",
n_workers = 12,
radii = c("r1250"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 5 * 1024^3)

# Diversity_Forest_r1250.tif    egv_103
slanis=rast("./RasterGrids_100m/2024/Raw/Diversity_Forest_r1250.tif")
names(slanis)="egv_103"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/Raw/Diversity_Forest_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Diversity_Forest_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.104 Diversity\_Forest\_r3000

**filename:** Diversity\_Forest\_r3000.tif

**layername:** egv\_104

**English name:** Average forest class  $\alpha$ -diversity of 500 m grid cells within the 3 km landscape

**Latvian name:** Vidējā mežu ainavas klašu 500 m šūnu  $\alpha$ -daudzveidība 3 km ainavā

**Procedure:** Derived from the [Landscape diversity](#), more precisely [Forest diversity](#). The average value of 25 ha cells diversity index values is calculated using the workflow `egvtools::radius_function()`. To prevent possible data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. File is written twice, to ensure layername. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",

```

```

radii_path = "./Templates/TemplateGridPoints/tiles/",
tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers = c("./RasterGrids_500m/2024/Diversity_Forests_500x.tif"),
layer_prefixes = c("Diversity_Forest"),
output_dir = "./RasterGrids_100m/2024/Raw//",
n_workers = 12,
radii = c("r3000"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 5 * 1024^3)

# Diversity_Forest_r3000.tif      egv_104
slanis=rast("./RasterGrids_100m/2024/Raw/Diversity_Forest_r3000.tif")
names(slanis)="egv_104"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/Raw/Diversity_Forest_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Diversity_Forest_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.105 Diversity\_Forest\_r10000

**filename:** Diversity\_Forest\_r10000.tif

**layername:** egv\_105

**English name:** Average forest class  $\alpha$ -diversity of 500 m grid cells within the 10 km landscape

**Latvian name:** Vidējā mežu ainavas klašu 500 m šūnu  $\alpha$ -daudzveidība 10 km ainavā

**Procedure:** Derived from the [Landscape diversity](#), more precisely [Forest diversity](#). The average value of 25 ha cells diversity index values is calculated using the workflow `egvtools::radius_function()`. To prevent possible data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. File is written twice, to ensure layername. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii
radius_function(

```

```

kvadrati_path = "./Templates/TemplateGrids/tiles/",
radii_path    = "./Templates/TemplateGridPoints/tiles/",
tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers  = c("./RasterGrids_500m/2024/Diversity_Forests_500x.tif"),
layer_prefixes = c("Diversity_Forest"),
output_dir    = "./RasterGrids_100m/2024/RAW/",
n_workers     = 12,
radii         = c("r10000"),
radius_mode   = "sparse",
extract_fun   = "mean",
fill_missing   = TRUE,
IDW_weight    = 2,
future_max_size = 5 * 1024^3)

# Diversity_Forest_r10000.tif  egv_105
slanis=rast("./RasterGrids_100m/2024/RAW/Diversity_Forest_r10000.tif")
names(slanis)="egv_105"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Diversity_Forest_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Diversity_Forest_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.106 Diversity\_Total\_r500

**filename:** Diversity\_Total\_r500.tif

**layername:** egv\_106

**English name:** Average combined landscape  $\alpha$ -diversity of 500 m grid cells within the 0.5 km landscape

**Latvian name:** Vidējā visu ainavas klašu 500 m šūnu  $\alpha$ -daudzveidība 0,5 km ainavā

**Procedure:** Derived from the [Landscape diversity](#), more precisely [Overall landscape diversity](#). The average value of 25 ha cells diversity index values is calculated using the workflow `egvtools::radius_function()`. To prevent possible data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. File is written twice, to ensure layername. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii

```

```

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_500m/2024/Diversity_GeneralLandscape_500x.tif"),
  layer_prefixes = c("Diversity_Total"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 5 * 1024^3)

# Diversity_Total_r500.tif  egv_106
slanis=rast("./RasterGrids_100m/2024/RAW/Diversity_Total_r500.tif")
names(slanis)="egv_106"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Diversity_Total_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Diversity_Total_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.107 Diversity\_Total\_r1250

**filename:** Diversity\_Total\_r1250.tif

**layername:** egv\_107

**English name:** Average combined landscape  $\alpha$ -diversity of 500 m grid cells within the 1.25 km landscape

**Latvian name:** Vidējā visu ainavas klašu 500 m šūnu  $\alpha$ -daudzveidība 1,25 km ainavā

**Procedure:** Derived from the [Landscape diversity](#), more precisely [Overall landscape diversity](#). The average value of 25 ha cells diversity index values is calculated using the workflow `egvtools::radius_function()`. To prevent possible data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. File is written twice, to ensure layername. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

```

```

# radii
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_500m/2024/Diversity_GeneralLandscape_500x.tif"),
  layer_prefixes = c("Diversity_Total"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii        = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 5 * 1024^3)

# Diversity_Total_r1250.tif egv_107
slanis=rast("./RasterGrids_100m/2024/RAW/Diversity_Total_r1250.tif")
names(slanis)="egv_107"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Diversity_Total_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Diversity_Total_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.108 Diversity\_Total\_r3000

**filename:** Diversity\_Total\_r3000.tif

**layername:** egv\_108

**English name:** Average combined landscape  $\alpha$ -diversity of 500 m grid cells within the 3 km landscape

**Latvian name:** Vidējā visu ainavas klašu 500 m šūnu  $\alpha$ -daudzveidība 3 km ainavā

**Procedure:** Derived from the [Landscape diversity](#), more precisely [Overall landscape diversity](#). The average value of 25 ha cells diversity index values is calculated using the workflow `egvtools::radius_function()`. To prevent possible data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. File is written twice, to ensure layername. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

```

```

# radii
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_500m/2024/Diversity_GeneralLandscape_500x.tif"),
  layer_prefixes = c("Diversity_Total"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 5 * 1024^3)

# Diversity_Total_r3000.tif evg_108
slanis=rast("./RasterGrids_100m/2024/RAW/Diversity_Total_r3000.tif")
names(slanis)="evg_108"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Diversity_Total_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Diversity_Total_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.109 Diversity\_Total\_r10000

**filename:** Diversity\_Total\_r10000.tif

**layername:** evg\_109

**English name:** Average combined landscape  $\alpha$ -diversity of 500 m grid cells within the 10 km landscape

**Latvian name:** Vidējā visu ainavas klašu 500 m šūnu  $\alpha$ -daudzveidība 10 km ainavā

**Procedure:** Derived from the [Landscape diversity](#), more precisely [Overall landscape diversity](#). The average value of 25 ha cells diversity index values is calculated using the workflow `egvtools::radius_function()`. To prevent possible data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. File is written twice, to ensure layername. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}

# templates ----

```

```

template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_500m/2024/Diversity_GeneralLandscape_500x.tif"),
  layer_prefixes = c("Diversity_Total"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 12,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 5 * 1024^3)

# Diversity_Total_r10000.tif      egv_109
slanis=rast("./RasterGrids_100m/2024/Raw/Diversity_Total_r10000.tif")
names(slanis)="egv_109"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/Diversity_Total_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Diversity_Total_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.110 Edges\_Bogs-Trees\_cell

**filename:** Edges\_Bogs-Trees\_cell.tif

**layername:** egv\_110

**English name:** Edge pixels of Bogs, Mires bordering with Trees within the analysis cell (1 ha)

**Latvian name:** Purvu malu ar kokiem pikseļu skaits analīzēs šūnā (1 ha)

**Procedure:** First, values from 620 to 700 from the [Landscape classification](#) are coded as 0, and all other values as NA. Then bog and transitional mire layers from the [EDI](#) are reclassified to presence-only (value 1) and combined. Then, bog-and-mire layer (1 = presence) is covered over tree layer (presence = 0) and written to file (matching the input). Then, with the workflow `egvtools::landscape_function()` total edge between the two classes is calculated. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}

```

```

if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")

# simple landscape ----
simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# Edges_Bogs-Trees_input.tif ----

trees_from620=ifel(simple_landscape>=620 & simple_landscape<700, 0, NA)
plot(trees_from620)

bogs=rast("./RasterGrids_10m/2024/EDI_BogsYN.tif")
bogs=subst(bogs, 0, NA)
plot(bogs)
mires=rast("./RasterGrids_10m/2024/EDI_TransitionalMiresYN.tif")
mires=subst(mires, 0, NA)
plot(mires)
bogs_mires=cover(bogs, mires)
plot(bogs_mires)

bm_trees=cover(bogs_mires, trees_from620)
plot(bm_trees)

edge_bm_trees=project(bm_trees, template10,
                      filename="./RasterGrids_10m/2024/Edges_Bogs-Trees_input.tif",
                      overwrite=TRUE)

rm(edge_bm_trees)
rm(bm_trees)

# Edges_Bogs-Trees_cell.tif egv_110

landscape_function(
  landscape = "./RasterGrids_10m/2024/Edges_Bogs-Trees_input.tif",
  zones     = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  id_field  = "id",
  tile_field = "tks50km",
  template   = "./Templates/TemplateRasters/LV100m_10km.tif",
  out_dir    = "./RasterGrids_100m/2024/RAW",
  out_filename = "Edges_Bogs-Trees_cell.tif",
  out_layername = "egv_110",
  what       = "lsm_l_te",
  lm_args    = list(count_boundary = FALSE),
  rasterize_engine = "fasterize",
  n_workers   = 12,
  future_max_size = 20 * 1024^3,
  fill_gaps   = TRUE,
  plot_gaps   = FALSE,
  plot_result = FALSE
)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Bogs-Trees_cell.tif"

```

```

ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.111 Edges\_Bogs-Trees\_r500

**filename:** Edges\_Bogs-Trees\_r500.tif

**layername:** evg\_111

**English name:** Edge pixels of Bogs, Mires bordering with Trees within the 0.5 km landscape

**Latvian name:** Purvu malu ar kokiem pikseļu skaits 0,5 km ainavā

**Procedure:** The total edge within a 500 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Bogs-Trees_cell.tif"),
  layer_prefixes = c("Edges_Bogs-Trees"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 4,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

# Edges_Bogs-Trees_r500.tif evg_111
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Bogs-Trees_r500.tif")
names(slanis)="evg_111"
slanis2=project(slanis,template100)
writeRaster(slanis2,
    "./RasterGrids_100m/2024/RAW/Edges_Bogs-Trees_r500.tif",
    overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

nosaukums="Edges_Bogs-Trees_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.112 Edges\_Bogs-Trees\_r1250

**filename:** Edges\_Bogs-Trees\_r1250.tif

**layername:** egv\_112

**English name:** Edge pixels of Bogs, Mires bordering with Trees within the 1.25 km landscape

**Latvian name:** Purvu malu ar kokiem pikselu skaits 1,25 km ainavā

**Procedure:** The total edge within a 1250 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Bogs-Trees_cell.tif"),
  layer_prefixes = c("Edges_Bogs-Trees"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 4,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Bogs-Trees_r1250.tif      egv_112
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Bogs-Trees_r1250.tif")
names(slanis)="egv_112"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Edges_Bogs-Trees_r1250.tif",
            overwrite=TRUE)

# standardisation -----

```

```

if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Bogs-Trees_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.113 Edges\_Bogs-Trees\_r3000

**filename:** Edges\_Bogs-Trees\_r3000.tif

**layername:** egv\_113

**English name:** Edge pixels of Bogs, Mires bordering with Trees within the 3 km landscape

**Latvian name:** Purvu malu ar kokiem pikseļu skaits 3 km ainavā

**Procedure:** Total edge within a 3000 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Bogs-Trees_cell.tif"),
  layer_prefixes = c("Edges_Bogs-Trees"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 4,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Bogs-Trees_r3000.tif      egv_113
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Bogs-Trees_r3000.tif")
names(slanis)="egv_113"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Bogs-Trees_r3000.tif",
  overwrite=TRUE)

```

```

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Bogs-Trees_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovidze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovidze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.114 Edges\_Bogs-Trees\_r10000

**filename:** Edges\_Bogs-Trees\_r10000.tif

**layername:** egv\_114

**English name:** Edge pixels of Bogs, Mires bordering with Trees within the 10 km landscape

**Latvian name:** Purvu malu ar kokiem pikselu skaits 10 km ainavā

**Procedure:** The total edge within a 10000 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Bogs-Trees_cell.tif"),
  layer_prefixes = c("Edges_Bogs-Trees"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 4,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

# Edges_Bogs-Trees_r10000.tif  egv_114
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Bogs-Trees_r10000.tif")
names(slanis)="egv_114"
slanis2=project(slanis,template100)
writeRaster(slanis2,

```

```

"./RasterGrids_100m/2024/Raw/Edges_Bogs-Trees_r10000.tif",
overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Bogs-Trees_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.115 Edges\_Bogs-Water\_cell

**filename:** Edges\_Bogs-Water\_cell.tif

**layername:** egv\_115

**English name:** Edge pixels of Bogs, Mires bordering with Water within the analysis cell (1 ha)

**Latvian name:** Purvu malu ar ūdeni pikseļu skaits analīzē šūnā (1 ha)

**Procedure:** First, values 200 from the [Landscape classification](#) are coded as 0, and all other values as NA. Then bog and transitional mire layers from [EDI](#) are reclassified to presence-only (value 1) and combined. Then, bog-and-mire layer (1 = presence) is covered over water layer (presence = 0) and written to file (matching the input). Then, using the workflow `egvtools::landscape_function()` total edge between the two classes is calculated. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")

# simple landscape ----
simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# Edges_Bogs-Water_input.tif ----
bogs=rast("./RasterGrids_10m/2024/EDI_BogsYN.tif")
bogs=subst(bogs,0,NA)
plot(bogs)
mires=rast("./RasterGrids_10m/2024/EDI_TransitionalMiresYN.tif")
mires=subst(mires,0,NA)
plot(mires)
bogs_mires=cover(bogs,mires)

```

```

plot(bogs_mires)

water=ifel(simple_landscape==200,0,NA)
plot(water)

bm_water=cover(bogs_mires,water)
plot(bm_water)

edge_bm_water=project(bm_water,template10,
                      filename=".~/RasterGrids_10m/2024/Edges_Bogs-Water_input.tif",
                      overwrite=TRUE)
rm(edge_bm_water)
rm(bm_water)

# Edges_Bogs-Water_cell.tif egv_115 ----
landscape_function(
  landscape   = ".~/RasterGrids_10m/2024/Edges_Bogs-Water_input.tif",
  zones       = ".~/Templates/TemplateGrids/tikls100_sauzeme.parquet",
  id_field    = "id",
  tile_field   = "tks50km",
  template    = ".~/Templates/TemplateRasters/LV100m_10km.tif",
  out_dir     = ".~/RasterGrids_100m/2024/RAW",
  out_filename = "Edges_Bogs-Water_cell.tif",
  out_layername = "egv_115",
  what        = "lsm_l_te",
  lm_args     = list(count_boundary = FALSE),
  rasterize_engine = "fasterize",
  n_workers   = 12,
  future_max_size = 20 * 1024^3,
  fill_gaps   = TRUE,
  plot_gaps   = FALSE,
  plot_result  = FALSE
)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Bogs-Water_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.116 Edges\_Bogs-Water\_r500

**filename:** Edges\_Bogs-Water\_r500.tif

**layername:** egv\_116

**English name:** Edge pixels of Bogs, Mires bordering with Water within the 0.5 km landscape

**Latvian name:** Purvu malu ar ūdeni pikseļu skaits 0,5 km ainavā

**Procedure:** The total edge within a 500 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the

output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Bogs-Water_cell.tif"),
  layer_prefixes = c("Edges_Bogs-Water"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Bogs-Water_r500.tif egv_116
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Bogs-Water_r500.tif")
names(slanis)="egv_116"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Bogs-Water_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Bogs-Water_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.117 Edges\_Bogs-Water\_r1250

**filename:** Edges\_Bogs-Water\_r1250.tif

**layername:** egv\_117

**English name:** Edge pixels of Bogs, Mires bordering with Water within the 1.25 km landscape

**Latvian name:** Purvu malu ar ūdeni pikseļu skaits 1,25 km ainavā

**Procedure:** The total edge within a 1250 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`.

During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")


# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Bogs-Water_cell.tif"),
  layer_prefixes = c("Edges_Bogs-Water"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

# Edges_Bogs-Water_r1250.tif    egv_117
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Bogs-Water_r1250.tif")
names(slanis)="egv_117"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Bogs-Water_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Bogs-Water_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.118 Edges\_Bogs-Water\_r3000

**filename:** Edges\_Bogs-Water\_r3000.tif

**layername:** egv\_118

**English name:** Edge pixels of Bogs, Mires bordering with Water within the 3 km landscape

**Latvian name:** Purvu malu ar ūdeni pikseļu skaits 3 km ainavā

**Procedure:** The total edge within a 3000 m radius around the analysis grid cell is calculated as the area-weighted sum of the `analysis cells` inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Bogs-Water_cell.tif"),
  layer_prefixes = c("Edges_Bogs-Water"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Bogs-Water_r3000.tif      egv_118
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Bogs-Water_r3000.tif")
names(slanis)="egv_118"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Bogs-Water_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Bogs-Water_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.119 Edges\_Bogs-Water\_r10000

**filename:** Edges\_Bogs-Water\_r10000.tif

**layername:** egv\_119

**English name:** Edge pixels of Bogs, Mires bordering with Water within the 10 km landscape

**Latvian name:** Purvu malu ar ūdeni pikselu skaits 10 km ainavā

**Procedure:** The total edge within a 10000 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Bogs-Water_cell.tif"),
  layer_prefixes = c("Edges_Bogs-Water"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

# Edges_Bogs-Water_r10000.tif  egv_119
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Bogs-Water_r10000.tif")
names(slanis)="egv_119"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Bogs-Water_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Bogs-Water_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.120 Edges\_Farmland-Builtup\_cell

**filename:** Edges\_Farmland-Builtup\_cell.tif

**layername:** egv\_120

**English name:** Edge pixels of Farmland bordering with Built-Up areas within the analysis cell (1 ha)

**Latvian name:** Lauksaimniecības zemju malu ar apbūvi pikselu skaits analīzes šūnā (1 ha)

**Procedure:** First, values larger than 300 and smaller than 400 from [Landscape classification](#) are coded as 1, and all other values as NA. Then values 500 from the [Landscape classification](#) are coded as 0, and all other values as NA. Then, the first layer (1 = presence) is covered over the second layer (presence = 0) and written to file (matching the input). Next, using the workflow `egvtools::landscape_function()` total edge between the two classes is calculated. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")

# simple landscape ----
simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# Edges_Farmland-Builtup_input.tif ----
farmland=ifel(simple_landscape>300 & simple_landscape<400, 1, NA)
plot(farmland)

builtup=ifel(simple_landscape==500, 0, NA)
plot(builtup)

farmland_builtup=cover(farmland,builtup)
plot(farmland_builtup)

edge_farmland_builtup=project(farmland_builtup,template10,
                               filename="./RasterGrids_10m/2024/Edges_Farmland-Builtup_input.tif",
                               overwrite=TRUE)
rm(edge_farmland_builtup)
rm(farmland_builtup)

# Edges_Farmland-Builtup_cell.tif  egv_120 ----
landscape_function(
  landscape    = "./RasterGrids_10m/2024/Edges_Farmland-Builtup_input.tif",
  zones        = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  id_field    = "id",
  tile_field   = "tks50km",
  template     = "./Templates/TemplateRasters/LV100m_10km.tif",
  out_dir      = "./RasterGrids_100m/2024/RAW",
  out_filename = "Edges_Farmland-Builtup_cell.tif",
  out_layername = "egv_120",
  what         = "lsm_l_te",
  lm_args      = list(count_boundary = FALSE),
  rasterize_engine = "fasterize",
  n_workers    = 12,
  future_max_size = 20 * 1024^3,
  fill_gaps    = TRUE,
  plot_gaps    = FALSE,
  plot_result  = FALSE
)
```

```

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Farmland-Builtup_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovidze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovidze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.121 Edges\_Farmland-Builtup\_r500

**filename:** Edges\_Farmland-Builtup\_r500.tif

**layername:** egv\_121

**English name:** Edge pixels of Farmland bordering with Built-Up areas within the 0.5 km landscape

**Latvian name:** Lauksaimniecības zemju malu ar apbūvi pikseļu skaits 0,5 km ainavā

**Procedure:** The total edge within a 500 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Farmland-Builtup_cell.tif"),
  layer_prefixes = c("Edges_Farmland-Builtup"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Farmland-Builtup_r500.tif  egv_121 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Farmland-Builtup_r500.tif")
names(slanis)="egv_121"
slanis2=project(slanis,template100)

```

```

writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/Edges_Farmland-Builtup_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Farmland-Builtup_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.122 Edges\_Farmland-Builtup\_r1250

**filename:** Edges\_Farmland-Builtup\_r1250.tif

**layername:** egv\_122

**English name:** Edge pixels of Farmland bordering with Built-Up areas within the 1.25 km landscape

**Latvian name:** Lauksaimniecības zemju malu ar apbūvi pikselu skaits 1,25 km ainavā

**Procedure:** The total edge within a 1250 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/Edges_Farmland-Builtup_cell.tif"),
  layer_prefixes = c("Edges_Farmland-Builtup"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 12,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Farmland-Builtup_r1250.tif egv_122 ----

```

```

slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Farmland-Builtup_r1250.tif")
names(slanis)="egv_122"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Edges_Farmland-Builtup_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Farmland-Builtup_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.123 Edges\_Farmland-Builtup\_r3000

**filename:** Edges\_Farmland-Builtup\_r3000.tif

**layername:** egv\_123

**English name:** Edge pixels of Farmland bordering with Built-Up areas within the 3 km landscape

**Latvian name:** Lauksaimniecības zemju malu ar apbūvi pikselu skaits 3 km ainavā

**Procedure:** The total edge within a 3000 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Farmland-Builtup_cell.tif"),
  layer_prefixes = c("Edges_Farmland-Builtup"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

```

```

# Edges_Farmland-Builtup_r3000.tif egv_123 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Farmland-Builtup_r3000.tif")
names(slanis)="egv_123"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Edges_Farmland-Builtup_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Farmland-Builtup_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.124 Edges\_Farmland-Builtup\_r10000

**filename:** Edges\_Farmland-Builtup\_r10000.tif

**layername:** egv\_124

**English name:** Edge pixels of Farmland bordering with Built-Up areas within the 10 km landscape

**Latvian name:** Lauksaimniecības zemju malu ar apbūvi pikselu skaits 10 km ainavā

**Procedure:** The total edge within a 10000 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path  = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   = c("./RasterGrids_100m/2024/RAW/Edges_Farmland-Builtup_cell.tif"),
  layer_prefixes = c("Edges_Farmland-Builtup"),
  output_dir     = "./RasterGrids_100m/2024/RAW/",
  n_workers      = 12,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",

```

```

fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 20 * 1024^3)

# Edges_Farmland-Builtup_r10000.tif egv_124 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Farmland-Builtup_r10000.tif")
names(slanis)="egv_124"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Farmland-Builtup_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Farmland-Builtup_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.125 Edges\_Trees-Builtup\_cell

**filename:** Edges\_Trees-Builtup\_cell.tif

**layername:** egv\_125

**English name:** Edge pixels of Trees bordering with Built-Up areas within the analysis cell (1 ha)

**Latvian name:** Koku malu ar apbūvi pikseļu skaits analīzēs šūnā (1 ha)

**Procedure:** First, values larger than 630 and smaller than 700 from [Landscape classification](#) are coded as 1, and other values as NA. Then values 500 from the [Landscape classification](#) are coded as 0, and other values as NA. Then, the first layer (1 = presence) is covered over the second layer (presence = 0) and written to file (matching the input). Next, using the workflow `egvtools::landscape_function()` total edge between the two classes is calculated. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")

# simple landscape ----

```

```

simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# Edges_Trees-Builtup_input.tif ----
trees_from630=ifel(simple_landscape>=630 & simple_landscape<700,1,NA)
plot(trees_from630)

builtup=ifel(simple_landscape==500,0,NA)
plot(builtup)

trees630_builtup=cover(trees_from630,builtup)
plot(trees630_builtup)

edge_trees630_builtup=project(trees630_builtup,template10,
                               filename="./RasterGrids_10m/2024/Edges_Trees-Builtup_input.tif",
                               overwrite=TRUE)
rm(edge_trees630_builtup)
rm(trees630_builtup)

# Edges_Trees-Builtup_cell.tif  egv_125 ----
landscape_function(
  landscape  = "./RasterGrids_10m/2024/Edges_Trees-Builtup_input.tif",
  zones      = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  id_field   = "id",
  tile_field = "tks50km",
  template   = "./Templates/TemplateRasters/LV100m_10km.tif",
  out_dir    = "./RasterGrids_100m/2024/RAW",
  out_filename = "Edges_Trees-Builtup_cell.tif",
  out_layername = "egv_125",
  what       = "lsm_l_te",
  lm_args    = list(count_boundary = FALSE),
  rasterize_engine = "fasterize",
  n_workers  = 12,
  future_max_size = 20 * 1024^3,
  fill_gaps   = TRUE,
  plot_gaps   = FALSE,
  plot_result = FALSE
)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Trees-Builtup_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.126 Edges\_Trees-Builtup\_r500

**filename:** Edges\_Trees-Builtup\_r500.tif

**layername:** egv\_126

**English name:** Edge pixels of Trees bordering with Built-Up areas within the 0.5 km landscape

**Latvian name:** Koku malu ar apbūvi pikseļu skaits 0,5 km ainavā

**Procedure:** The total edge within a 500 m radius around the analysis grid cell is calculated as the area-weighted sum of the `analysis cells` inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Trees-Builtup_cell.tif"),
  layer_prefixes = c("Edges_Trees-Builtup"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Trees-Builtup_r500.tif  egv_126 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Trees-Builtup_r500.tif")
names(slanis)="egv_126"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Trees-Builtup_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Trees-Builtup_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.127 Edges\_Trees-Builtup\_r1250

**filename:** Edges\_Trees-Builtup\_r1250.tif

**layername:** egv\_127

**English name:** Edge pixels of Trees bordering with Built-Up areas within the 1.25 km landscape

**Latvian name:** Koku malu ar apbūvi pikseļu skaits 1,25 km ainavā

**Procedure:** The total edge within a 1250 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes:::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/Edges_Trees-Builtup_cell.tif"),
  layer_prefixes = c("Edges_Trees-Builtup"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 12,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Trees-Builtup_r1250.tif egv_127 ----
slanis=rast("./RasterGrids_100m/2024/Raw/Edges_Trees-Builtup_r1250.tif")
names(slanis)="egv_127"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/Edges_Trees-Builtup_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Trees-Builtup_r1250.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.128 Edges\_Trees-Builtup\_r3000

**filename:** Edges\_Trees-Builtup\_r3000.tif

**layername:** evg\_128

**English name:** Edge pixels of Trees bordering with Built-Up areas within the 3 km landscape

**Latvian name:** Koku malu ar apbūvi pikseļu skaits 3 km ainavā

**Procedure:** The total edge within a 3000 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Trees-Builtup_cell.tif"),
  layer_prefixes = c("Edges_Trees-Builtup"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii        = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Trees-Builtup_r3000.tif evg_128 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Trees-Builtup_r3000.tif")
names(slanis)="evg_128"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Trees-Builtup_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Trees-Builtup_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.129 Edges\_Trees-Builtup\_r10000

**filename:** Edges\_Trees-Builtup\_r10000.tif

**layername:** egv\_129

**English name:** Edge pixels of Trees bordering with Built-Up areas within the 10 km landscape

**Latvian name:** Koku malu ar apbūvi pikseļu skaits 10 km ainavā

**Procedure:** The total edge within a 10000 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Trees-Builtup_cell.tif"),
  layer_prefixes = c("Edges_Trees-Builtup"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

# Edges_Trees-Builtup_r10000.tif    egv_129 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Trees-Builtup_r10000.tif")
names(slanis)="egv_129"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Trees-Builtup_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Trees-Builtup_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.130 Edges\_CropsFallow\_cell

**filename:** Edges\_CropsFallow\_cell.tif

**layername:** evg\_130

**English name:** Edge pixels of Cropland, Fallow land within the analysis cell (1 ha)

**Latvian name:** Aramzemju malu pikselu skaits analīzes šūnā (1 ha)

**Procedure:** First, values larger than or equal to 310 and smaller than 325 from the [Landscape classification](#) are coded as 1, and all other values as NA. Then, the layer (1 = presence) is covered over the nulls layer (presence = 0) and written to file (matching the input). Next, using the workflow `egv-tools::landscape_function()` total edge between the two classes is calculated. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}


# Templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")

# simple landscape ----
simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# Edges_CropsFallow_input.tif ----
cropsfallow=ifel(simple_landscape>=310 & simple_landscape<325, 1, NA)
plot(cropsfallow)
cropsfallow=cover(cropsfallow,nulls10)
plot(cropsfallow)

edge_cropsfallow=project(cropsfallow,template10,
                         filename="./RasterGrids_10m/2024/Edges_CropsFallow_input.tif",
                         overwrite=TRUE)
rm(edge_cropsfallow)


# Edges_CropsFallow_cell.tif      evg_130 ----
landscape_function(
  landscape   = "./RasterGrids_10m/2024/Edges_CropsFallow_input.tif",
  zones       = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  id_field    = "id",
  tile_field  = "tks50km",
  template    = "./Templates/TemplateRasters/LV100m_10km.tif",
  out_dir     = "./RasterGrids_100m/2024/RAW",
  out_filename = "Edges_CropsFallow_cell.tif",
  out_layername = "evg_130",
  what        = "lsm_l_te",
  lm_args     = list(count_boundary = FALSE),
  rasterize_engine = "fasterize",
  n_workers   = 12,
  future_max_size = 20 * 1024^3,
  fill_gaps   = TRUE,
  plot_gaps   = FALSE,
  plot_result = FALSE
```

```

)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_CropsFallow_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.131 Edges\_CropsFallow\_r500

**filename:** Edges\_CropsFallow\_r500.tif

**layername:** egv\_131

**English name:** Edge pixels of Cropland, Fallow land within the 0.5 km landscape

**Latvian name:** Aramzemju malu pikselu skaits 0,5 km ainavā

**Procedure:** The total edge within a 500 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/Edges_CropsFallow_cell.tif"),
  layer_prefixes = c("Edges_CropsFallow"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 12,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

# Edges_CropsFallow_r500.tif    egv_131 ----
slanis=rast("./RasterGrids_100m/2024/Raw/Edges_CropsFallow_r500.tif")
names(slanis)="egv_131"

```

```

slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Edges_CropsFallow_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_CropsFallow_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.132 Edges\_CropsFallow\_r1250

**filename:** Edges\_CropsFallow\_r1250.tif

**layername:** egv\_132

**English name:** Edge pixels of Cropland, Fallow land within the 1.25 km landscape

**Latvian name:** Aramzemju malu pikselu skaits 1,25 km ainavā

**Procedure:** The total edge within a 1250 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_CropsFallow_cell.tif"),
  layer_prefixes = c("Edges_CropsFallow"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

```

```

# Edges_CropsFallow_r1250.tif  evg_132 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_CropsFallow_r1250.tif")
names(slanis)="evg_132"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Edges_CropsFallow_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_CropsFallow_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.133 Edges\_CropsFallow\_r3000

**filename:** Edges\_CropsFallow\_r3000.tif

**layername:** evg\_133

**English name:** Edge pixels of Cropland, Fallow land within the 3 km landscape

**Latvian name:** Aramzemu malu pikselu skaits 3 km ainavā

**Procedure:** The total edge within a 3000 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path  = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   = c("./RasterGrids_100m/2024/RAW/Edges_CropsFallow_cell.tif"),
  layer_prefixes = c("Edges_CropsFallow"),
  output_dir     = "./RasterGrids_100m/2024/RAW/",
  n_workers      = 12,
  radii          = c("r3000"),
  radius_mode    = "sparse",
  extract_fun    = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,

```

```

future_max_size = 20 * 1024^3)

# Edges_CropsFallow_r3000.tif  egv_133 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_CropsFallow_r3000.tif")
names(slanis)="egv_133"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Edges_CropsFallow_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_CropsFallow_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.134 Edges\_CropsFallow\_r10000

**filename:** Edges\_CropsFallow\_r10000.tif

**layername:** egv\_134

**English name:** Edge pixels of Cropland, Fallow land within the 10 km landscape

**Latvian name:** Aramzemju malu pikselu skaits 10 km ainavā

**Procedure:** The total edge within a 10000 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_CropsFallow_cell.tif"),
  layer_prefixes = c("Edges_CropsFallow"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r10000"),
  radius_mode   = "sparse",

```

```

extract_fun = "sum",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 20 * 1024^3)

# Edges_CropsFallow_r10000.tif egv_134 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_CropsFallow_r10000.tif")
names(slanis)="egv_134"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Edges_CropsFallow_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_CropsFallow_r10000.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.135 Edges\_FarmlandShrubs-Trees\_cell

**filename:** Edges\_FarmlandShrubs-Trees\_cell.tif

**layername:** egv\_135

**English name:** Edge pixels of Farmland, Clear-Cuts, Shrubs bordering with Trees within the analysis cell (1 ha)

**Latvian name:** Lauksaimniecības zemju, izcirtumu, krūmu malu ar kokiem pikseļu skaits analīzes šūnā (1 ha)

**Procedure:** First, values between 300 and 400 and between 600 and 630 from [Landscape classification](#) are coded as 0, and all other values as NA. Then values larger than or equal to 630 but smaller than 700 from the [Landscape classification](#) are coded as 1, and all other values as NA. Then, the first layer (0 = presence) is covered over the second layer (presence = 1) and written to file (matching the input). Next, using the workflow `egvtools::landscape_function()` total edge between the two classes is calculated. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Templates ----

```

```

template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")

# simple landscape ----
simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# Edges_FarmlandShrubs-Trees_input.tif ----
farmshrub=ifel((simple_landscape>300 & simple_landscape<400) |
  (simple_landscape>600 & simple_landscape<630), 0, NA)

trees_from630=ifel(simple_landscape>=630 & simple_landscape<700, 1, NA)
plot(trees_from630)

farmshrub_trees630=cover(farmshrub, trees_from630)
plot(farmshrub_trees630)

edge_farmshrub_trees630=project(farmshrub_trees630, template10,
  filename="./RasterGrids_10m/2024/Edges_FarmlandShrubs-Trees_input.tif",
  overwrite=TRUE)
rm(edge_farmshrub_trees630)
rm(farmshrub_trees630)

# Edges_FarmlandShrubs-Trees_cell.tif    egv_135 ----
landscape_function(
  landscape = "./RasterGrids_10m/2024/Edges_FarmlandShrubs-Trees_input.tif",
  zones     = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  id_field  = "id",
  tile_field = "tks50km",
  template   = "./Templates/TemplateRasters/LV100m_10km.tif",
  out_dir    = "./RasterGrids_100m/2024/RAW",
  out_filename = "Edges_FarmlandShrubs-Trees_cell.tif",
  out_layername = "egv_135",
  what      = "lsm_l_te",
  lm_args    = list(count_boundary = FALSE),
  rasterize_engine = "fasterize",
  n_workers  = 12,
  future_max_size = 20 * 1024^3,
  fill_gaps   = TRUE,
  plot_gaps   = FALSE,
  plot_result = FALSE
)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_FarmlandShrubs-Trees_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/", nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/", nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis, fun="mean", na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets, fun="rms", na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.136 Edges\_FarmlandShrubs-Trees\_r500

**filename:** Edges\_FarmlandShrubs-Trees\_r500.tif

**layername:** egv\_136

**English name:** Edge pixels of Farmland, Clear-Cuts, Shrubs bordering with Trees within the 0.5 km landscape

**Latvian name:** Lauksaimniecības zemju, izcirtumu, krūmu malu ar kokiem pikseļu skaits 0,5 km ainavā

**Procedure:** The total edge within a 500 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_FarmlandShrubs-Trees_cell.tif"),
  layer_prefixes = c("Edges_FarmlandShrubs-Trees"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

# Edges_FarmlandShrubs-Trees_r500.tif  egv_136 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_FarmlandShrubs-Trees_r500.tif")
names(slanis)="egv_136"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_FarmlandShrubs-Trees_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_FarmlandShrubs-Trees_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.137 Edges\_FarmlandShrubs-Trees\_r1250

**filename:** Edges\_FarmlandShrubs-Trees\_r1250.tif

**layername:** evg\_137

**English name:** Edge pixels of Farmland, Clear-Cuts, Shrubs bordering with Trees within the 1.25 km landscape

**Latvian name:** Lauksaimniecības zemju, izcirtumu, krūmu malu ar kokiem pikselu skaits 1,25 km ainavā

**Procedure:** The total edge within a 1250 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_FarmlandShrubs-Trees_cell.tif"),
  layer_prefixes = c("Edges_FarmlandShrubs-Trees"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

# Edges_FarmlandShrubs-Trees_r1250.tif evg_137 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_FarmlandShrubs-Trees_r1250.tif")
names(slanis)="evg_137"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Edges_FarmlandShrubs-Trees_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_FarmlandShrubs-Trees_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
```

```
overwrite=TRUE)
```

## 6.138 Edges\_FarmlandShrubs-Trees\_r3000

**filename:** Edges\_FarmlandShrubs-Trees\_r3000.tif

**layername:** evg\_138

**English name:** Edge pixels of Farmland, Clear-Cuts, Shrubs bordering with Trees within the 3 km landscape

**Latvian name:** Lauksaimniecības zemju, izcirtumu, krūmu malu ar kokiem pikselu skaits 3 km ainavā

**Procedure:** The total edge within a 3000 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_FarmlandShrubs-Trees_cell.tif"),
  layer_prefixes = c("Edges_FarmlandShrubs-Trees"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_FarmlandShrubs-Trees_r3000.tif evg_138 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_FarmlandShrubs-Trees_r3000.tif")
names(slanis)="evg_138"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_FarmlandShrubs-Trees_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_FarmlandShrubs-Trees_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
```

```

merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.139 Edges\_FarmlandShrubs-Trees\_r10000

**filename:** Edges\_FarmlandShrubs-Trees\_r10000.tif

**layername:** egv\_139

**English name:** Edge pixels of Farmland, Clear-Cuts, Shrubs bordering with Trees within the 10 km landscape

**Latvian name:** Lauksaimniecības zemju, izcirtumu, krūmu malu ar kokiem pikselu skaits 10 km ainavā

**Procedure:** The total edge within a 10000 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadратi_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/Edges_FarmlandShrubs-Trees_cell.tif"),
  layer_prefixes = c("Edges_FarmlandShrubs-Trees"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 12,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

# Edges_FarmlandShrubs-Trees_r10000.tif egv_139 ----
slanis=rast("./RasterGrids_100m/2024/Raw/Edges_FarmlandShrubs-Trees_r10000.tif")
names(slanis)="egv_139"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/Raw/Edges_FarmlandShrubs-Trees_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_FarmlandShrubs-Trees_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)

```

```

saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=ticta::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.140 Edges\_Grasslands\_cell

**filename:** Edges\_Grasslands\_cell.tif

**layername:** egv\_140

**English name:** Edge pixels of Grassland within the analysis cell (1 ha)

**Latvian name:** Zālāju malu pikselu skaits analīzes šūnā (1 ha)

**Procedure:** First, values equal to 330 from the [Landscape classification](#) are coded as 1, and all other values as NA. Then, the layer (1 = presence) is covered over the nulls layer (presence = 0) and written to file (matching the input). Then, using the workflow `egvtools::landscape_function()` total edge between the two classes is calculated. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")

# simple landscape ----
simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# Edges_Grasslands_input.tif ----
grassland=ifel(simple_landscape==330,1,NA)
plot(grassland)
grassland=cover(grassland,nulls10)
plot(grassland)

edge_grassland=project(grassland,template10,
    filename="./RasterGrids_10m/2024/Edges_Grasslands_input.tif",
    overwrite=TRUE)
rm(edge_grassland)

# Edges_Grasslands_cell.tif egv_140 ----
landscape_function(
  landscape   = "./RasterGrids_10m/2024/Edges_Grasslands_input.tif",
  zones       = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  id_field   = "id",
  tile_field  = "tks50km",

```

```

template      = "./Templates/TemplateRasters/LV100m_10km.tif",
out_dir      = "./RasterGrids_100m/2024/RAW",
out_filename = "Edges_Grasslands_cell.tif",
out_layername = "egv_140",
what         = "lsm_l_te",
lm_args       = list(count_boundary = FALSE),
rasterize_engine = "fasterize",
n_workers    = 12,
future_max_size = 20 * 1024^3,
fill_gaps    = TRUE,
plot_gaps    = FALSE,
plot_result  = FALSE
)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Grasslands_cell.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.141 Edges\_Grasslands\_r500

**filename:** Edges\_Grasslands\_r500.tif

**layername:** egv\_141

**English name:** Edge pixels of Grassland within the 0.5 km landscape

**Latvian name:** Zālāju malu pikselu skaits 0,5 km ainavā

**Procedure:** The total edge within a 500 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Grasslands_cell.tif"),
  layer_prefixes = c("Edges_Grasslands"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",

```

```

n_workers    = 12,
radii        = c("r500"),
radius_mode  = "sparse",
extract_fun   = "sum",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 20 * 1024^3)

# Edges_Grasslands_r500.tif egv_141 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Grasslands_r500.tif")
names(slanis)="egv_141"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Edges_Grasslands_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Grasslands_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.142 Edges\_Grasslands\_r1250

**filename:** Edges\_Grasslands\_r1250.tif

**layername:** egv\_142

**English name:** Edge pixels of Grassland within the 1.25 km landscape

**Latvian name:** Zālāju malu pikseļu skaits 1,25 km ainavā

**Procedure:** The total edge within a 1250 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",

```

```

input_layers = c("./RasterGrids_100m/2024/RAW/Edges_Grasslands_cell.tif"),
layer_prefixes = c("Edges_Grasslands"),
output_dir = "./RasterGrids_100m/2024/RAW/",
n_workers = 12,
radii = c("r1250"),
radius_mode = "sparse",
extract_fun = "sum",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 20 * 1024^3)

# Edges_Grasslands_r1250.tif    egv_142 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Grasslands_r1250.tif")
names(slanis)="egv_142"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Edges_Grasslands_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Grasslands_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.143 Edges\_Grasslands\_r3000

**filename:** Edges\_Grasslands\_r3000.tif

**layername:** egv\_143

**English name:** Edge pixels of Grassland within the 3 km landscape

**Latvian name:** Zālāju malu pikseļu skaits 3 km ainavā

**Procedure:** The total edge within a 3000 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",

```

```

radii_path = "./Templates/TemplateGridPoints/tiles/",
tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers = c("./RasterGrids_100m/2024/RAW/Edges_Grasslands_cell.tif"),
layer_prefixes = c("Edges_Grasslands"),
output_dir = "./RasterGrids_100m/2024/RAW/",
n_workers = 12,
radii = c("r3000"),
radius_mode = "sparse",
extract_fun = "sum",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 20 * 1024^3)

# Edges_Grasslands_r3000.tif    egv_143 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Grasslands_r3000.tif")
names(slanis)="egv_143"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Edges_Grasslands_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Grasslands_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.144 Edges\_Grasslands\_r10000

**filename:** Edges\_Grasslands\_r10000.tif

**layername:** egv\_144

**English name:** Edge pixels of Grassland within the 10 km landscape

**Latvian name:** Zālāju malu pikselu skaits 10 km ainavā

**Procedure:** The total edge within a 10000 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

```

```

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Grasslands_cell.tif"),
  layer_prefixes = c("Edges_Grasslands"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii        = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Grasslands_r10000.tif  evg_144 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Grasslands_r10000.tif")
names(slanis)="evg_144"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Grasslands_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Grasslands_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.145 Edges\_OldForests\_cell

**filename:** Edges\_OldForests\_cell.tif

**layername:** evg\_145

**English name:** Edge pixels of Forests Over Rotation Age within the analysis cell (1 ha)

**Latvian name:** Pieaugušo un pāraugušo mežaudžu malu pikseļu skaits analīzes šūnā (1 ha)

**Procedure:** First, the raster layer with forest stands from the **MVR** at age groups 4 and 5 is prepared (presence = 1, everything else = NA). Then, the layer (1 = presence) is covered over the nulls layer (presence = 0) and written to file (matching the input). Then, using the workflow `egvtools::landscape_function()` total edge between the two classes is calculated. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

```

```

if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")

# simple landscape ----
simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# Edges_OldForests_input.tif ----
mvr=sfarrow::st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr2=mvr %>%
  mutate(forest_age=ifelse(vgr=="4" | vgr=="5", 1, NA)) %>%
  filter(!is.na(forest_age))

rast_old=fasterize(mvr2,raster(template10),field="forest_age")
terra_old=rast(rast_old)
plot(terra_old)
terra_old=cover(terra_old,nulls10)
plot(terra_old)

edge_old=project(terra_old,template10,
                 filename="./RasterGrids_10m/2024/Edges_OldForests_input.tif",
                 overwrite=TRUE)
rm(mvr)
rm(mvr2)
rm(rast_old)
rm(terra_old)
rm(edge_old)

# Edges_OldForests_cell.tif evg_145 ----
landscape_function(
  landscape   = "./RasterGrids_10m/2024/Edges_OldForests_input.tif",
  zones       = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  id_field    = "id",
  tile_field  = "tks50km",
  template    = "./Templates/TemplateRasters/LV100m_10km.tif",
  out_dir     = "./RasterGrids_100m/2024/RAW",
  out_filename = "Edges_OldForests_cell.tif",
  out_layername = "evg_145",
  what        = "lsm_l_te",
  lm_args     = list(count_boundary = FALSE),
  rasterize_engine = "fasterize",
  n_workers   = 12,
  future_max_size = 20 * 1024^3,
  fill_gaps   = TRUE,
  plot_gaps   = FALSE,
  plot_result = FALSE
)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_OldForests_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)

```

```

centrets=slanis$videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.146 Edges\_OldForests\_r500

**filename:** Edges\_OldForests\_r500.tif

**layername:** egv\_146

**English name:** Edge pixels of Forests Over Rotation Age within the 0.5 km landscape

**Latvian name:** Pieaugušo un pāraugušo mežaudžu malu pikseļu skaits 0,5 km ainavā

**Procedure:** The total edge within a 500 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_OldForests_cell.tif"),
  layer_prefixes = c("Edges_OldForests"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_OldForests_r500.tif egv_146 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_OldForests_r500.tif")
names(slanis)="egv_146"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_OldForests_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_OldForests_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)

```

```

saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.147 Edges\_OldForests\_r1250

**filename:** Edges\_OldForests\_r1250.tif

**layername:** egv\_147

**English name:** Edge pixels of Forests Over Rotation Age within the 1.25 km landscape

**Latvian name:** Pieaugušo un pāraugušo mežaudžu malu pikselu skaits 1,25 km ainavā

**Procedure:** The total edge within a 1250 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/Edges_OldForests_cell.tif"),
  layer_prefixes = c("Edges_OldForests"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 12,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_OldForests_r1250.tif    egv_147 ----
slanis=rast("./RasterGrids_100m/2024/Raw/Edges_OldForests_r1250.tif")
names(slanis)="egv_147"
slanis2=project(slanis,template100)
writeRaster(slanis2,
    "./RasterGrids_100m/2024/Raw/Edges_OldForests_r1250.tif",
    overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

nosaukums="Edges_OldForests_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.148 Edges\_OldForests\_r3000

**filename:** Edges\_OldForests\_r3000.tif

**layername:** egv\_148

**English name:** Edge pixels of Forests Over Rotation Age within the 3 km landscape

**Latvian name:** Pieaugušo un pāraugušo mežaudžu malu pikselu skaits 3 km ainavā

**Procedure:** The total edge within a 3000 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_OldForests_cell.tif"),
  layer_prefixes = c("Edges_OldForests"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_OldForests_r3000.tif      egv_148 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_OldForests_r3000.tif")
names(slanis)="egv_148"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_OldForests_r3000.tif",
  overwrite=TRUE)

```

```

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_OldForests_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.149 Edges\_OldForests\_r10000

**filename:** Edges\_OldForests\_r10000.tif

**layername:** egv\_149

**English name:** Edge pixels of Forests Over Rotation Age within the 10 km landscape

**Latvian name:** Pieaugušo un pāraugušo mežaudžu malu pikselu skaits 10 km ainavā

**Procedure:** The total edge within a 10000 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_OldForests_cell.tif"),
  layer_prefixes = c("Edges_OldForests"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_OldForests_r10000.tif  egv_149 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_OldForests_r10000.tif")
names(slanis)="egv_149"
slanis2=project(slanis,template100)
writeRaster(slanis2,

```

```

"./RasterGrids_100m/2024/Raw/Edges_OldForests_r10000.tif",
overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_OldForests_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.150 Edges\_Roads\_cell

**filename:** Edges\_Roads\_cell.tif

**layername:** egv\_150

**English name:** Edge pixels of Roads within the analysis cell (1 ha)

**Latvian name:** Ceļu malu pikselu skaits analīzēs šūnā (1 ha)

**Procedure:** First, values equal to 100 from the [Landscape classification](#) are coded as 1, and other values as NA. Then, the layer (1 = presence) is covered over the nulls layer (presence = 0) and written to file (matching the input). Next, with the workflow `egvtools::landscape_function()` total edge between the two classes is calculated. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")

# simple landscape ----
simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# Edges_Roads_input.tif ----
roads=ifel(simple_landscape==100,1,NA)
plot(roads)
roads=cover(roads,nulls10)
plot(roads)

edge_roads=project(roads,template10,
    filename="./RasterGrids_10m/2024/Edges_Roads_input.tif",
    overwrite=TRUE)

```

```

rm(edge_roads)

# Edges_Roads_cell.tif  evg_150 ----
landscape_function(
  landscape  = "./RasterGrids_10m/2024/Edges_Roads_input.tif",
  zones      = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  id_field   = "id",
  tile_field  = "tks50km",
  template    = "./Templates/TemplateRasters/LV100m_10km.tif",
  out_dir     = "./RasterGrids_100m/2024/Raw",
  out_filename = "Edges_Roads_cell.tif",
  out_layername = "evg_150",
  what        = "lsm_l_te",
  lm_args     = list(count_boundary = FALSE),
  rasterize_engine = "fasterize",
  n_workers   = 12,
  future_max_size = 20 * 1024^3,
  fill_gaps   = TRUE,
  plot_gaps   = FALSE,
  plot_result  = FALSE
)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Roads_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.151 Edges\_Roads\_r500

**filename:** Edges\_Roads\_r500.tif

**layername:** evg\_151

**English name:** Edge pixels of Roads within the 0.5 km landscape

**Latvian name:** Ceļu malu pikselu skaits 0,5 km ainavā

**Procedure:** The total edge within a 500 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

```

```

# radii ----
radius_function(
  kvadri_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Roads_cell.tif"),
  layer_prefixes = c("Edges_Roads"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii        = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Roads_r500.tif  egv_151 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Roads_r500.tif")
names(slanis)="egv_151"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Roads_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Roads_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.152 Edges\_Roads\_r1250

**filename:** Edges\_Roads\_r1250.tif

**layername:** egv\_152

**English name:** Edge pixels of Roads within the 1.25 km landscape

**Latvian name:** Ceļu malu pikselu skaits 1,25 km ainavā

**Procedure:** The total edge within a 1250 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

```

```

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Roads_cell.tif"),
  layer_prefixes = c("Edges_Roads"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

# Edges_Roads_r1250.tif egv_152 -----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Roads_r1250.tif")
names(slanis)="egv_152"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Roads_r1250.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Roads_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.153 Edges\_Roads\_r3000

**filename:** Edges\_Roads\_r3000.tif

**layername:** egv\_153

**English name:** Edge pixels of Roads within the 3 km landscape

**Latvian name:** Ceļu malu pikselu skaits 3 km ainavā

**Procedure:** The total edge within a 3000 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs -----
if(!require(terra)) {install.packages("terra"); require(terra)}

```

```

if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Roads_cell.tif"),
  layer_prefixes = c("Edges_Roads"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Roads_r3000.tif egv_153 -----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Roads_r3000.tif")
names(slanis)="egv_153"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Roads_r3000.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Roads_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.154 Edges\_Roads\_r10000

**filename:** Edges\_Roads\_r10000.tif

**layername:** egv\_154

**English name:** Edge pixels of Roads within the 10 km landscape

**Latvian name:** Ceļu malu pikseļu skaits 10 km ainavā

**Procedure:** The total edge within a 10000 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Roads_cell.tif"),
  layer_prefixes = c("Edges_Roads"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

# Edges_Roads_r10000.tif    egv_154 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Roads_r10000.tif")
names(slanis)="egv_154"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Roads_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Roads_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.155 Edges\_Trees\_cell

**filename:** Edges\_Trees\_cell.tif

**layername:** egv\_155

**English name:** Edge pixels of Trees within the analysis cell (1 ha)

**Latvian name:** Koku malu pikseļu skaits analīzes šūnā (1 ha)

**Procedure:** First, values larger or equal to 630 and smaller than 700 from [Landscape classification](#) are coded as 1, and all other values as NA. Then, the layer (1 = presence) is covered over the nulls layer (presence = 0) and written to file (matching the input). Next, with the workflow `egvtools::landscape_function()` total

edge between the two classes is calculated. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")

# simple landscape ----
simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# Edges_Trees_input.tif ----
trees_from630=ifel(simple_landscape>=630 & simple_landscape<700, 1, NA)
plot(trees_from630)
trees_from630=cover(trees_from630,nulls10)
plot(trees_from630)

edge_trees_from630=project(trees_from630,template10,
                           filename="./RasterGrids_10m/2024/Edges_Trees_input.tif",
                           overwrite=TRUE)
rm(edge_trees_from630)

# Edges_Trees_cell.tif  egv_155
landscape_function(
  landscape  = "./RasterGrids_10m/2024/Edges_Trees_input.tif",
  zones      = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  id_field   = "id",
  tile_field = "tks50km",
  template   = "./Templates/TemplateRasters/LV100m_10km.tif",
  out_dir    = "./RasterGrids_100m/2024/Raw",
  out_filename = "Edges_Trees_cell.tif",
  out_layername = "egv_155",
  what       = "lsm_l_te",
  lm_args    = list(count_boundary = FALSE),
  rasterize_engine = "fasterize",
  n_workers  = 12,
  future_max_size = 20 * 1024^3,
  fill_gaps  = TRUE,
  plot_gaps  = FALSE,
  plot_result = FALSE
)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Trees_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
```

```

merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.156 Edges\_Trees\_r500

**filename:** Edges\_Trees\_r500.tif

**layername:** egv\_156

**English name:** Edge pixels of Trees within the 0.5 km landscape

**Latvian name:** Koku malu pikselu skaits 0,5 km ainavā

**Procedure:** The total edge within a 500 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Trees_cell.tif"),
  layer_prefixes = c("Edges_Trees"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Trees_r500.tif egv_156 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Trees_r500.tif")
names(slanis)="egv_156"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Edges_Trees_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Trees_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)

```

```

videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.157 Edges\_Trees\_r1250

**filename:** Edges\_Trees\_r1250.tif

**layername:** egv\_157

**English name:** Edge pixels of Trees within the 1.25 km landscape

**Latvian name:** Koku malu pikseļu skaits 1,25 km ainavā

**Procedure:** The total edge within a 1250 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/Edges_Trees_cell.tif"),
  layer_prefixes = c("Edges_Trees"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 12,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

# Edges_Trees_r1250.tif egv_157 ----
slanis=rast("./RasterGrids_100m/2024/Raw/Edges_Trees_r1250.tif")
names(slanis)="egv_157"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/Edges_Trees_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Trees_r1250.tif"

```

```

ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.158 Edges\_Trees\_r3000

**filename:** Edges\_Trees\_r3000.tif

**layername:** evg\_158

**English name:** Edge pixels of Trees within the 3 km landscape

**Latvian name:** Koku malu pikseļu skaits 3 km ainavā

**Procedure:** The total edge within a 3000 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Trees_cell.tif"),
  layer_prefixes = c("Edges_Trees"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Trees_r3000.tif evg_158 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Trees_r3000.tif")
names(slanis)="evg_158"
slanis2=project(slanis,template100)
writeRaster(slanis2,
    "./RasterGrids_100m/2024/RAW/Edges_Trees_r3000.tif",
    overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}

```

```

if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Trees_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.159 Edges\_Trees\_r10000

**filename:** Edges\_Trees\_r10000.tif

**layername:** egv\_159

**English name:** Edge pixels of Trees within the 10 km landscape

**Latvian name:** Koku malu pikselu skaits 10 km ainavā

**Procedure:** The total edge within a 10000 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/Edges_Trees_cell.tif"),
  layer_prefixes = c("Edges_Trees"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 12,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Trees_r10000.tif      egv_159 ----
slanis=rast("./RasterGrids_100m/2024/Raw/Edges_Trees_r10000.tif")
names(slanis)="egv_159"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/Raw/Edges_Trees_r10000.tif",
            overwrite=TRUE)

```

```

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Trees_r1000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.160 Edges\_Water\_cell

**filename:** Edges\_Water\_cell.tif

**layername:** egv\_160

**English name:** Edge pixels of Water within the analysis cell (1 ha)

**Latvian name:** Ūdenstilpu malu pikseļu skaits analīzes šūnā (1 ha)

**Procedure:** First, values equal to 200 from the [Landscape classification](#) are coded as 1 and everything else as NA. Then, the layer (1 = presence) is covered over the nulls layer (presence = 0) and written to file (matching the input). Next, with the workflow `egvtools::landscape_function()` total edge between the two classes is calculated. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")

# simple landscape ----
simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# Edges_Water_input.tif ----
water=ifel(simple_landscape==200,1,0)
plot(water)
water=cover(water,nulls10)
plot(water)

edge_water=project(water,template10,
                    filename="./RasterGrids_10m/2024/Edges_Water_input.tif",
                    overwrite=TRUE)

```

```

# Edges_Water_cell.tif  egv_160 ----
landscape_function(
  landscape = "./RasterGrids_10m/2024/Edges_Water_input.tif",
  zones = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  id_field = "id",
  tile_field = "tks50km",
  template = "./Templates/TemplateRasters/LV100m_10km.tif",
  out_dir = "./RasterGrids_100m/2024/RAW",
  out_filename = "Edges_Water_cell.tif",
  out_layername = "egv_160",
  what = "lsm_l_te",
  lm_args = list(count_boundary = FALSE),
  rasterize_engine = "fasterize",
  n_workers = 12,
  future_max_size = 20 * 1024^3,
  fill_gaps = TRUE,
  plot_gaps = FALSE,
  plot_result = FALSE
)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Water_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.161 Edges\_Water\_r500

**filename:** Edges\_Water\_r500.tif

**layername:** egv\_161

**English name:** Edge pixels of Water within the 0.5 km landscape

**Latvian name:** Ūdenstilpju malu pikselu skaits 0,5 km ainavā

**Procedure:** The total edge within a 500 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",

```

```

radii_path = "./Templates/TemplateGridPoints/tiles/",
tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers = c("./RasterGrids_100m/2024/RAW/Edges_Water_cell.tif"),
layer_prefixes = c("Edges_Water"),
output_dir = "./RasterGrids_100m/2024/RAW/",
n_workers = 12,
radii = c("r500"),
radius_mode = "sparse",
extract_fun = "sum",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 20 * 1024^3)

# Edges_Water_r500.tif evg_161 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Water_r500.tif")
names(slanis)="evg_161"
slanis2=project(slanis,template100)
writeRaster(slanis2,
           "./RasterGrids_100m/2024/RAW/Edges_Water_r500.tif",
           overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Water_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.162 Edges\_Water\_r1250

**filename:** Edges\_Water\_r1250.tif

**layername:** evg\_162

**English name:** Edge pixels of Water within the 1.25 km landscape

**Latvian name:** Ūdenstilpu malu pikseļu skaits 1,25 km ainavā

**Procedure:** The total edge within a 1250 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

```

```

# radii ----
radius_function(
  kvadri_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Water_cell.tif"),
  layer_prefixes = c("Edges_Water"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii        = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Water_r1250.tif egv_162 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Water_r1250.tif")
names(slanis)="egv_162"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Water_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Water_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.163 Edges\_Water\_r3000

**filename:** Edges\_Water\_r3000.tif

**layername:** egv\_163

**English name:** Edge pixels of Water within the 3 km landscape

**Latvian name:** Ūdenstilpu malu pikselu skaits 3 km ainavā

**Procedure:** The total edge within a 3000 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

```

```

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Water_cell.tif"),
  layer_prefixes = c("Edges_Water"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

# Edges_Water_r3000.tif egv_163 -----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Water_r3000.tif")
names(slanis)="egv_163"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Water_r3000.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Water_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.164 Edges\_Water\_r10000

**filename:** Edges\_Water\_r10000.tif

**layername:** egv\_164

**English name:** Edge pixels of Water within the 10 km landscape

**Latvian name:** Ūdenstilpu malu pikseļu skaits 10 km ainavā

**Procedure:** The total edge within a 10000 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs -----
if(!require(terra)) {install.packages("terra"); require(terra)}

```

```

if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Water_cell.tif"),
  layer_prefixes = c("Edges_Water"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Water_r10000.tif      egv_164 -----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Water_r10000.tif")
names(slanis)="egv_164"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Water_r10000.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Water_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.165 Edges\_Water-Farmland\_cell

**filename:** Edges\_Water-Farmland\_cell.tif

**layername:** egv\_165

**English name:** Edge pixels of Water bordering with Farmland within the analysis cell (1 ha)

**Latvian name:** Ūdenstilpju malu ar lauksaimniecības zemēm pikseļu skaits analīzes šūnā (1 ha)

**Procedure:** First, values larger than 300 and smaller than 400 from [Landscape classification](#) are coded as 1, and all other values as NA. Then values equal to 200 from the [Landscape classification](#) are coded as 0, and all other values as NA. Then, the first layer (1 = presence) is covered over the second layer (presence = 0) and written to file (matching the input). Next, using the workflow `egvtools::landscape_function()` total edge between the two classes is calculated. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")

# simple landscape ----
simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# Edges_Water-Farmland_input.tif ----
water=ifel(simple_landscape==200,0,NA)
plot(water)

farmland=ifel(simple_landscape>300 & simple_landscape<400,1,NA)
plot(farmland)

water_farmland=cover(water,farmland)
plot(water_farmland)

edge_water_farmland=project(water_farmland,template10,
                           filename="./RasterGrids_10m/2024/Edges_Water-Farmland_input.tif",
                           overwrite=TRUE)
rm(edge_water_farmland)
rm(water_farmland)

# Edges_Water-Farmland_cell.tif egv_165 ----
landscape_function(
  landscape   = "./RasterGrids_10m/2024/Edges_Water-Farmland_input.tif",
  zones       = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  id_field    = "id",
  tile_field   = "tks50km",
  template    = "./Templates/TemplateRasters/LV100m_10km.tif",
  out_dir     = "./RasterGrids_100m/2024/RAW",
  out_filename = "Edges_Water-Farmland_cell.tif",
  out_layername = "egv_165",
  what        = "lsm_l_te",
  lm_args     = list(count_boundary = FALSE),
  rasterize_engine = "fasterize",
  n_workers   = 12,
  future_max_size = 20 * 1024^3,
  fill_gaps   = TRUE,
  plot_gaps   = FALSE,
  plot_result = FALSE
)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Water-Farmland_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]

```

```

standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.166 Edges\_Water-Farmland\_r500

**filename:** Edges\_Water-Farmland\_r500.tif

**layername:** evg\_166

**English name:** Edge pixels of Water bordering with Farmland within the 0.5 km landscape

**Latvian name:** Ūdenstilpu malu ar lauksaimniecības zemēm pikseļu skaits 0,5 km ainavā

**Procedure:** The total edge within a 500 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Water-Farmland_cell.tif"),
  layer_prefixes = c("Edges_Water-Farmland"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Water-Farmland_r500.tif evg_166 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Water-Farmland_r500.tif")
names(slanis)="evg_166"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Water-Farmland_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Water-Farmland_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)

```

```

slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.167 Edges\_Water-Farmland\_r1250

**filename:** Edges\_Water-Farmland\_r1250.tif

**layername:** egv\_167

**English name:** Edge pixels of Water bordering with Farmland within the 1.25 km landscape

**Latvian name:** Ūdenstilpu malu ar lauksaimniecības zemēm pikselu skaits 1,25 km ainavā

**Procedure:** The total edge within a 1250 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow [egvtools::radius\\_function\(\)](#). During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/Edges_Water-Farmland_cell.tif"),
  layer_prefixes = c("Edges_Water-Farmland"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 12,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Water-Farmland_r1250.tif    egv_167 ----
slanis=rast("./RasterGrids_100m/2024/Raw/Edges_Water-Farmland_r1250.tif")
names(slanis)="egv_167"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/Edges_Water-Farmland_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

nosaukums="Edges_Water-Farmland_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.168 Edges\_Water-Farmland\_r3000

**filename:** Edges\_Water-Farmland\_r3000.tif

**layername:** evg\_168

**English name:** Edge pixels of Water bordering with Farmland within the 3 km landscape

**Latvian name:** Ūdenstilpju malu ar lauksaimniecības zemēm pikseļu skaits 3 km ainavā

**Procedure:** The total edge within a 3000 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/Edges_Water-Farmland_cell.tif"),
  layer_prefixes = c("Edges_Water-Farmland"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 12,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Water-Farmland_r3000.tif    evg_168 ----
slanis=rast("./RasterGrids_100m/2024/Raw/Edges_Water-Farmland_r3000.tif")
names(slanis)="evg_168"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/Edges_Water-Farmland_r3000.tif",
  overwrite=TRUE)

# standardisation ----

```

```

if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Water-Farmland_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.169 Edges\_Water-Farmland\_r10000

**filename:** Edges\_Water-Farmland\_r10000.tif

**layername:** egv\_169

**English name:** Edge pixels of Water bordering with Farmland within the 10 km landscape

**Latvian name:** Ūdenstilpu malu ar lauksaimniecības zemēm pikselu skaits 10 km ainavā

**Procedure:** The total edge within a 10000 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadri_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/Edges_Water-Farmland_cell.tif"),
  layer_prefixes = c("Edges_Water-Farmland"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 12,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Water-Farmland_r10000.tif egv_169 ----
slanis=rast("./RasterGrids_100m/2024/Raw/Edges_Water-Farmland_r10000.tif")
names(slanis)="egv_169"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/Edges_Water-Farmland_r10000.tif",

```

```

overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Water-Farmland_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.170 Edges\_Water-Grassland\_cell

**filename:** Edges\_Water-Grassland\_cell.tif

**layername:** egv\_170

**English name:** Edge pixels of Water bordering with Grassland within the analysis cell (1 ha)

**Latvian name:** Ūdenstilpu malu ar zālājiem pikseļu skaits analīzes šūnā (1 ha)

**Procedure:** First, values equal to 330 from the [Landscape classification](#) are coded as 1, and all other values as NA. Then values equal to 200 from the [Landscape classification](#) are coded as 0, and all other values as NA. Then, the first layer (1 = presence) is covered over the second layer (presence = 0) and written to file (matching the input). Next, with the workflow `egvtools::landscape_function()` total edge between the two classes is calculated. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")

# simple landscape ----
simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# Edges_Water-Grassland_input.tif ----
water=ifel(simple_landscape==200,0,NA)
plot(water)

grassland=ifel(simple_landscape==330,1,NA)
plot(grassland)

water_grassland=cover(water,grassland)
plot(water_grassland)

```

```

edge_water_grassland=project(water_grassland,template10,
                               filename=".~/RasterGrids_10m/2024/Edges_Water–Grassland_input.tif",
                               overwrite=TRUE)
rm(edge_water_grassland)
rm(water_grassland)

# Edges_Water–Grassland_cell.tif    egv_170 ----
landscape_function(
  landscape   = ".~/RasterGrids_10m/2024/Edges_Water–Grassland_input.tif",
  zones       = ".~/Templates/TemplateGrids/tikls100_sauzeme.parquet",
  id_field    = "id",
  tile_field   = "tks50km",
  template    = ".~/Templates/TemplateRasters/LV100m_10km.tif",
  out_dir     = ".~/RasterGrids_100m/2024/RAW",
  out_filename = "Edges_Water–Grassland_cell.tif",
  out_layername = "egv_170",
  what        = "lsm_l_te",
  lm_args     = list(count_boundary = FALSE),
  rasterize_engine = "fasterize",
  n_workers   = 12,
  future_max_size = 20 * 1024^3,
  fill_gaps   = TRUE,
  plot_gaps   = FALSE,
  plot_result  = FALSE
)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Water–Grassland_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.171 Edges\_Water–Grassland\_r500

**filename:** Edges\_Water–Grassland\_r500.tif

**layername:** egv\_171

**English name:** Edge pixels of Water bordering with Grassland within the 0.5 km landscape

**Latvian name:** Ūdenstilpu malu ar zālājiem pikselu skaits 0,5 km ainavā

**Procedure:** The total edge within a 500 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

```

```

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Water-Grassland_cell.tif"),
  layer_prefixes = c("Edges_Water-Grassland"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Water-Grassland_r500.tif    egv_171 -----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Water-Grassland_r500.tif")
names(slanis)="egv_171"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Water-Grassland_r500.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Water-Grassland_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.172 Edges\_Water-Grassland\_r1250

**filename:** Edges\_Water-Grassland\_r1250.tif

**layername:** egv\_172

**English name:** Edge pixels of Water bordering with Grassland within the 1.25 km landscape

**Latvian name:** Ūdenstilpju malu ar zālājiem pikselu skaits 1,25 km ainavā

**Procedure:** The total edge within a 1250 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/Edges_Water-Grassland_cell.tif"),
  layer_prefixes = c("Edges_Water-Grassland"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 12,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

# Edges_Water-Grassland_r1250.tif egv_172 ----
slanis=rast("./RasterGrids_100m/2024/Raw/Edges_Water-Grassland_r1250.tif")
names(slanis)="egv_172"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/Edges_Water-Grassland_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Water-Grassland_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.173 Edges\_Water-Grassland\_r3000

**filename:** Edges\_Water-Grassland\_r3000.tif

**layername:** egv\_173

**English name:** Edge pixels of Water bordering with Grassland within the 3 km landscape

**Latvian name:** Ūdenstilpju malu ar zālājiem pikseļu skaits 3 km ainavā

**Procedure:** The total edge within a 3000 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the

output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/Edges_Water-Grassland_cell.tif"),
  layer_prefixes = c("Edges_Water-Grassland"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 12,
  radii        = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_Water-Grassland_r3000.tif  egv_173 ----
slanis=rast("./RasterGrids_100m/2024/Raw/Edges_Water-Grassland_r3000.tif")
names(slanis)="egv_173"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/Edges_Water-Grassland_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Water-Grassland_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.174 Edges\_Water-Grassland\_r10000

**filename:** Edges\_Water-Grassland\_r10000.tif

**layername:** egv\_174

**English name:** Edge pixels of Water bordering with Grassland within the 10 km landscape

**Latvian name:** Ūdenstilpju malu ar zālājiem pikseļu skaits 10 km ainavā

**Procedure:** The total edge within a 10000 m radius around the analysis grid cell is calculated as the area-weighted sum of the `analysis cells` inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_Water-Grassland_cell.tif"),
  layer_prefixes = c("Edges_Water-Grassland"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

# Edges_Water-Grassland_r10000.tif egv_174 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_Water-Grassland_r10000.tif")
names(slanis)="egv_174"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Edges_Water-Grassland_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_Water-Grassland_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.175 Edges\_ReedSedgeRushBeds-Water\_cell

**filename:** Edges\_ReedSedgeRushBeds-Water\_cell.tif

**layername:** egv\_175

**English name:** Edge pixels of Reed-, Sedge-, Rush- Beds bordering with Water within the analysis cell (1 ha)

**Latvian name:** Niedrāju, grīslāju, meldrāju malu ar ūdeni pikseļu skaits analīzē šūnā (1 ha)

**Procedure:** First, values equal to 720 from the [Landscape classification](#) are coded as 1, and all other values as NA. Then values equal to 200 from the [Landscape classification](#) are coded as 0, and all other values as NA. Then, the first layer (1 = presence) is covered over the second layer (presence = 0) and written to file (matching the input). Next, with the workflow `egvtools::landscape_function()` total edge between the two classes is calculated. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

# Templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")

# simple landscape ----
simple_landscape=rast("./RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# Edges_ReedSedgeRushBeds-Water_input.tif ----
water=ifel(simple_landscape==200,0,NA)
plot(water)

reedssedgerush=ifel(simple_landscape==720,1,NA)
plot(reedssedgerush)

reedssedgerush_water=cover(reedssedgerush,water)
plot(reedssedgerush_water)

edge_reedssedgerush_water=project(reedssedgerush_water,template10,
                                    filename="./RasterGrids_10m/2024/Edges_ReedSedgeRushBeds-Water_input.tif",
                                    overwrite=TRUE)
rm(edge_reedssedgerush_water)
rm(reedssedgerush_water)

# Edges_ReedSedgeRushBeds-Water_cell.tif      egv_175 ----
landscape_function(
  landscape   = "./RasterGrids_10m/2024/Edges_ReedSedgeRushBeds-Water_input.tif",
  zones       = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  id_field    = "id",
  tile_field   = "tks50km",
  template     = "./Templates/TemplateRasters/LV100m_10km.tif",
  out_dir     = "./RasterGrids_100m/2024/RAW",
  out_filename = "Edges_ReedSedgeRushBeds-Water_cell.tif",
  out_layername = "egv_175",
  what        = "lsm_l_te",
  lm_args     = list(count_boundary = FALSE),
  rasterize_engine = "fasterize",
  n_workers   = 12,
  future_max_size = 20 * 1024^3,
  fill_gaps   = TRUE,
  plot_gaps   = FALSE,
```

```

    plot_result  = FALSE
  )

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_ReedSedgeRushBeds-Water_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.176 Edges\_ReedSedgeRushBeds-Water\_r500

**filename:** Edges\_ReedSedgeRushBeds-Water\_r500.tif

**layername:** egv\_176

**English name:** Edge pixels of Reed-, Sedge-, Rush- Beds bordering with Water within the 0.5 km landscape

**Latvian name:** Niedrāju, grīslāju, meldrāju malu ar ūdeni pikselu skaits 0,5 km ainavā

**Procedure:** The total edge within a 500 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path  = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   = c("./RasterGrids_100m/2024/Raw/Edges_ReedSedgeRushBeds-Water_cell.tif"),
  layer_prefixes = c("Edges_ReedSedgeRushBeds-Water"),
  output_dir     = "./RasterGrids_100m/2024/Raw/",
  n_workers      = 12,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "sum",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Edges_ReedSedgeRushBeds-Water_r500.tif    egv_176 ----
slanis=rast("./RasterGrids_100m/2024/Raw/Edges_ReedSedgeRushBeds-Water_r500.tif")

```

```

names(slanis)="egv_176"
slanis2=project(slanis,template100)
writeRaster(slanis2,
    "./RasterGrids_100m/2024/RAW/Edges_ReedSedgeRushBeds-Water_r500.tif",
    overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_ReedSedgeRushBeds-Water_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.177 Edges\_ReedSedgeRushBeds-Water\_r1250

**filename:** Edges\_ReedSedgeRushBeds-Water\_r1250.tif

**layername:** egv\_177

**English name:** Edge pixels of Reed-, Sedge-, Rush- Beds bordering with Water within the 1.25 km landscape

**Latvian name:** Niedrāju, grīslāju, meldrāju malu ar ūdeni pikseļu skaits 1,25 km ainavā

**Procedure:** The total edge within a 1250 m radius around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
    kvadrati_path = "./Templates/TemplateGrids/tiles/",
    radii_path    = "./Templates/TemplateGridPoints/tiles/",
    tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
    template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
    input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_ReedSedgeRushBeds-Water_cell.tif"),
    layer_prefixes = c("Edges_ReedSedgeRushBeds-Water"),
    output_dir    = "./RasterGrids_100m/2024/RAW/",
    n_workers     = 12,
    radii         = c("r1250"),
    radius_mode   = "sparse",
    extract_fun   = "sum",
    fill_missing   = TRUE,
    IDW_weight    = 2,

```

```

future_max_size = 20 * 1024^3)

# Edges_ReedSedgeRushBeds-Water_r1250.tif  egv_177 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_ReedSedgeRushBeds-Water_r1250.tif")
names(slanis)="egv_177"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Edges_ReedSedgeRushBeds-Water_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_ReedSedgeRushBeds-Water_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.178 Edges\_ReedSedgeRushBeds-Water\_r3000

**filename:** Edges\_ReedSedgeRushBeds-Water\_r3000.tif

**layername:** egv\_178

**English name:** Edge pixels of Reed-, Sedge-, Rush- Beds bordering with Water within the 3 km landscape

**Latvian name:** Niedrāju, grīslāju, meldrāju malu ar ūdeni pikseļu skaits 3 km ainavā

**Procedure:** The total edge within a 3000 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_ReedSedgeRushBeds-Water_cell.tif"),
  layer_prefixes = c("Edges_ReedSedgeRushBeds-Water"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r3000"),
  radius_mode   = "sparse",

```

```

extract_fun  = "sum",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 20 * 1024^3)

# Edges_ReedSedgeRushBeds-Water_r3000.tif  egv_178 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_ReedSedgeRushBeds-Water_r3000.tif")
names(slanis)="egv_178"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Edges_ReedSedgeRushBeds-Water_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_ReedSedgeRushBeds-Water_r3000.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.179 Edges\_ReedSedgeRushBeds-Water\_r10000

**filename:** Edges\_ReedSedgeRushBeds-Water\_r10000.tif

**layername:** egv\_179

**English name:** Edge pixels of Reed-, Sedge-, Rush- Beds bordering with Water within the 10 km landscape

**Latvian name:** Niedrāju, grīslāju, meldrāju malu ar ūdeni pikselu skaits 10 km ainavā

**Procedure:** The total edge within a 10000 m radius around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Edges_ReedSedgeRushBeds-Water_cell.tif"),
  layer_prefixes = c("Edges_ReedSedgeRushBeds-Water"),
  output_dir    = "./RasterGrids_100m/2024/RAW/"
)

```

```

n_workers    = 12,
radii        = c("r10000"),
radius_mode  = "sparse",
extract_fun   = "sum",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 20 * 1024^3)

# Edges_ReedSedgeRushBeds-Water_r10000.tif evg_179 ----
slanis=rast("./RasterGrids_100m/2024/RAW/Edges_ReedSedgeRushBeds-Water_r10000.tif")
names(slanis)="evg_179"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Edges_ReedSedgeRushBeds-Water_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Edges_ReedSedgeRushBeds-Water_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.180 FarmlandCrops\_CropsAll\_cell

**filename:** FarmlandCrops\_CropsAll\_cell.tif

**layername:** evg\_180

**English name:** Fractional cover of Crops (all types) within the analysis cell (1 ha)

**Latvian name:** Aramzemju (dažādu lauksaimniecības kultūraugu) platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, agricultural parcels with any type of crops are selected from the [Rural Support Service's information on declared fields](#). These geometries are then rasterised to input resolution, ensuring value 1 at the polygon locations and value 0 elsewhere. Rasterisation is performed with the workflow `egvtools::polygon2input()`. Once rasterised, the layer is aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----

```

```

template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# FarmlandCrops_CropsAll_cell.tif    evg_180 ----
aramzemes=lad %>%
  filter(str_detect(SDM_grupa_sakums,"aramz"))

p2i_rez=egvtools::polygon2input(vector_data = aramzemes,
                                 template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                 out_path = "./RasterGrids_10m/2024/",
                                 file_name = "FarmlandCrops_CropsAll_input.tif",
                                 value_field = "yes",
                                 prepare=FALSE,
                                 background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                 plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "FarmlandCrops_CropsAll_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/Raw/",
                             outfilename = "FarmlandCrops_CropsAll_cell.tif",
                             layername = "evg_180",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(p2i_rez)
rm(i2e_rez)
rm(aramzemes)
unlink("./RasterGrids_10m/2024/FarmlandCrops_CropsAll_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsAll_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.181 FarmlandCrops\_CropsAll\_r500

**filename:** FarmlandCrops\_CropsAll\_r500.tif

**layername:** egv\_181

**English name:** Fractional cover of Crops (all types) within the 0.5 km landscape

**Latvian name:** Aramzemju (dažādu lauksaimniecības kultūraugu) platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsAll_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsAll"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsAll_r500.tif  egv_181 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsAll_r500.tif")
names(slanis)="egv_181"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsAll_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsAll_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
```

```
overwrite=TRUE)
```

## 6.182 FarmlandCrops\_CropsAll\_r1250

**filename:** FarmlandCrops\_CropsAll\_r1250.tif

**layername:** evg\_182

**English name:** Fractional cover of Crops (all types) within the 1.25 km landscape

**Latvian name:** Aramzemju (dažādu lauksaimniecības kultūraugu) platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path   = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers    = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsAll_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsAll"),
  output_dir      = "./RasterGrids_100m/2024/RAW/",
  n_workers       = 6,
  radii          = c("r1250"),
  radius_mode    = "sparse",
  extract_fun    = "mean",
  fill_missing   = TRUE,
  IDW_weight     = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsAll_r1250.tif evg_182 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsAll_r1250.tif")
names(slanis)="evg_182"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsAll_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsAll_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
```

```

standartnovirze=terra::global(centrets, fun="rms", na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.183 FarmlandCrops\_CropsAll\_r3000

**filename:** FarmlandCrops\_CropsAll\_r3000.tif

**layername:** egv\_183

**English name:** Fractional cover of Crops (all types) within the 3 km landscape

**Latvian name:** Aramzemju (dažādu lauksaimniecības kultūraugu) platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsAll_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsAll"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsAll_r3000.tif egv_183 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsAll_r3000.tif")
names(slanis)="egv_183"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsAll_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsAll_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)

```

```

saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.184 FarmlandCrops\_CropsAll\_r10000

**filename:** FarmlandCrops\_CropsAll\_r10000.tif

**layername:** egv\_184

**English name:** Fractional cover of Crops (all types) within the 10 km landscape

**Latvian name:** Aramzemju (dažādu lauksaimniecības kultūraugu) platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsAll_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsAll"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing  = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsAll_r10000.tif egv_184 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsAll_r10000.tif")
names(slanis)="egv_184"
slanis2=project(slanis,template100)
writeRaster(slanis2,
    "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsAll_r10000.tif",
    overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}

```

```

if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsAll_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.185 FarmlandCrops\_CropsHoed\_cell

**filename:** FarmlandCrops\_CropsHoed\_cell.tif

**layername:** egv\_185

**English name:** Fractional cover of Hoed Crops within the analysis cell (1 ha)

**Latvian name:** Vagu un rušināmkultūru platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, agricultural parcels declared as hoed crops are selected from the [Rural Support Service's information on declared fields](#). These geometries are then rasterised to input resolution, ensuring value 1 at the polygon locations and value 0 elsewhere. Rasterisation is performed using the workflow `egvtools::polygon2input()`. Once rasterised, the layer is aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

```

```

# FarmlandCrops_CropsHoed_cell.tif  egv_185 ----
dati=lad %>%
  filter(str_detect(SDM_grupa_sakums,"ruši"))
table(dati$SDM_grupa_sakums,useNA="always")

p2i_rez=egvtools::polygon2input(vector_data = dati,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "FarmlandCrops_CropsHoed_input.tif",
  value_field = "yes",
  prepare=FALSE,
  background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
  plot_result = TRUE)

p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
  "FarmlandCrops_CropsHoed_input.tif"),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "FarmlandCrops_CropsHoed_cell.tif",
  layername = "egv_185",
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(p2i_rez)
rm(i2e_rez)
rm(dati)
unlink("./RasterGrids_10m/2024/FarmlandCrops_CropsHoed_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsHoed_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.186 FarmlandCrops\_CropsHoed\_r500

**filename:** FarmlandCrops\_CropsHoed\_r500.tif

**layername:** egv\_186

**English name:** Fractional cover of Hoed Crops within the 0.5 km landscape

**Latvian name:** Vagu un rušināmkultūru platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsHoed_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsHoed"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsHoed_r500.tif egv_186 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsHoed_r500.tif")
names(slanis)="egv_186"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsHoed_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsHoed_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.187 FarmlandCrops\_CropsHoed\_r1250

**filename:** FarmlandCrops\_CropsHoed\_r1250.tif

**layername:** egv\_187

**English name:** Fractional cover of Hoed Crops within the 1.25 km landscape

**Latvian name:** Vagu un rušināmkultūru platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the

output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsHoed_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsHoed"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsHoed_r1250.tif egv_187 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsHoed_r1250.tif")
names(slanis)="egv_187"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsHoed_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsHoed_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.188 FarmlandCrops\_CropsHoed\_r3000

**filename:** FarmlandCrops\_CropsHoed\_r3000.tif

**layername:** egv\_188

**English name:** Fractional cover of Hoed Crops within the 3 km landscape

**Latvian name:** Vagu un rušināmkultūru platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsHoed_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsHoed"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsHoed_r3000.tif egv_188 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsHoed_r3000.tif")
names(slanis)="egv_188"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsHoed_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsHoed_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.189 FarmlandCrops\_CropsHoed\_r10000

**filename:** FarmlandCrops\_CropsHoed\_r10000.tif

**layername:** egv\_189

**English name:** Fractional cover of Hoed Crops within the 10 km landscape

**Latvian name:** Vagu un rušināmkultūru platības īpatsvarts 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes:::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsHoed_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsHoed"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsHoed_r10000.tif    egv_189 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsHoed_r10000.tif")
names(slanis)="egv_189"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsHoed_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsHoed_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.190 FarmlandCrops\_CropsOther\_cell

**filename:** FarmlandCrops\_CropsOther\_cell.tif

**layername:** evg\_190

**English name:** Fractional cover of Other Crops within the analysis cell (1 ha)

**Latvian name:** Citu lauksaimniecības kultūraugu aramzemes platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, agricultural parcels with otherwise not differentiated crops are selected from the [Rural Support Service's information on declared fields](#). These geometries are then rasterised to input resolution, ensuring value 1 at the polygon locations and value 0 elsewhere. Rasterisation is performed using the workflow `egvtools::polygon2input()`. Once rasterised, the layer is aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# FarmlandCrops_CropsOther_cell.tif evg_190 ----
dati=lad %>%
  filter(str_detect(SDM_grupa_sakums,"citur neie"))
table(dati$SDM_grupa_sakums,useNA="always")

p2i_rez=egvtools::polygon2input(vector_data = dati,
                                template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                out_path = "./RasterGrids_10m/2024/",
                                file_name = "FarmlandCrops_CropsOther_input.tif",
                                value_field = "yes",
                                prepare=FALSE,
                                background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "FarmlandCrops_CropsOther_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
```

```

        outlocation = "./RasterGrids_100m/2024/RAW/",
        outfilename = "FarmlandCrops_CropsOther_cell.tif",
        layername = "egv_190",
        idw_weight = 2,
        plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(p2i_rez)
rm(i2e_rez)
rm(dati)
unlink("./RasterGrids_10m/2024/FarmlandCrops_CropsOther_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsOther_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.191 FarmlandCrops\_CropsOther\_r500

**filename:** FarmlandCrops\_CropsOther\_r500.tif

**layername:** egv\_191

**English name:** Fractional cover of Other Crops within the 0.5 km landscape

**Latvian name:** Citu lauksaimniecības kultūraugu aramzemēs platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsOther_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsOther"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",

```

```

extract_fun  = "mean",
fill_missing = TRUE,
IDW_weight  = 2,
future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsOther_r500.tif egv_191 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsOther_r500.tif")
names(slanis)="egv_191"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsOther_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsOther_r500.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.192 FarmlandCrops\_CropsOther\_r1250

**filename:** FarmlandCrops\_CropsOther\_r1250.tif

**layername:** egv\_192

**English name:** Fractional cover of Other Crops within the 1.25 km landscape

**Latvian name:** Citu lauksaimniecības kultūraugu aramzemes platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path  = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsOther_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsOther"),
  output_dir     = "./RasterGrids_100m/2024/RAW/",

```

```

n_workers    = 6,
radii        = c("r1250"),
radius_mode  = "sparse",
extract_fun   = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsOther_r1250.tif    egv_192 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsOther_r1250.tif")
names(slanis)="egv_192"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsOther_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsOther_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.193 FarmlandCrops\_CropsOther\_r3000

**filename:** FarmlandCrops\_CropsOther\_r3000.tif

**layername:** egv\_193

**English name:** Fractional cover of Other Crops within the 3 km landscape

**Latvian name:** Citu lauksaimniecības kultūraugu aramzemēs platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",

```

```

input_layers = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsOther_cell.tif"),
layer_prefixes = c("FarmlandCrops_CropsOther"),
output_dir = "./RasterGrids_100m/2024/RAW/",
n_workers = 6,
radii = c("r3000"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsOther_r3000.tif    egv_193 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsOther_r3000.tif")
names(slanis)="egv_193"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsOther_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsOther_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.194 FarmlandCrops\_CropsOther\_r10000

**filename:** FarmlandCrops\_CropsOther\_r10000.tif

**layername:** egv\_194

**English name:** Fractional cover of Other Crops within the 10 km landscape

**Latvian name:** Citu lauksaimniecības kultūraugu aramzemēs platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",

```

```

radii_path = "./Templates/TemplateGridPoints/tiles/",
tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsOther_cell.tif"),
layer_prefixes = c("FarmlandCrops_CropsOther"),
output_dir = "./RasterGrids_100m/2024/RAW/",
n_workers = 6,
radii = c("r10000"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsOther_r10000.tif  egv_194 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsOther_r10000.tif")
names(slanis)="egv_194"
slanis2=project(slanis,template100)
writeRaster(slanis2,
           "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsOther_r10000.tif",
           overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsOther_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.195 FarmlandCrops\_CropsSpring\_cell

**filename:** FarmlandCrops\_CropsSpring\_cell.tif

**layername:** egv\_195

**English name:** Fractional cover of Spring Sown Crops within the analysis cell (1 ha)

**Latvian name:** Vasarāju aramzemēs platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, agricultural parcels declared as spring sown crops are selected from the [Rural Support Service's information on declared fields](#). These geometries are then rasterised to input resolution, ensuring value 1 at the polygon locations and value 0 elsewhere. Rasterisation is performed using the workflow `egvtools::polygon2input()`. Once rasterised, the layer is aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# FarmlandCrops_CropsSpring_cell.tif    egv_195 ----
dati=lad %>%
  filter(str_detect(SDM_grupa_sakums,"labiba-vasarāji"))
table(dati$SDM_grupa_sakums,useNA="always")

p2i_rez=egvtools::polygon2input(vector_data = dati,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "FarmlandCrops_CropsSpring_input.tif",
  value_field = "yes",
  prepare=FALSE,
  background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
  plot_result = TRUE)

p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
  "FarmlandCrops_CropsSpring_input.tif"),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "FarmlandCrops_CropsSpring_cell.tif",
  layername = "egv_195",
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(p2i_rez)
rm(i2e_rez)
rm(dati)
unlink("./RasterGrids_10m/2024/FarmlandCrops_CropsSpring_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsSpring_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)

```

```

videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.196 FarmlandCrops\_CropsSpring\_r500

**filename:** FarmlandCrops\_CropsSpring\_r500.tif

**layername:** egv\_196

**English name:** Fractional cover of Spring Sown Crops within the 0.5 km landscape

**Latvian name:** Vasarāju aramzemēs platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsSpring_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsSpring"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsSpring_r500.tif    egv_196 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsSpring_r500.tif")
names(slanis)="egv_196"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsSpring_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsSpring_r500.tif"

```

```

ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.197 FarmlandCrops\_CropsSpring\_r1250

**filename:** FarmlandCrops\_CropsSpring\_r1250.tif

**layername:** egv\_197

**English name:** Fractional cover of Spring Sown Crops within the 1.25 km landscape

**Latvian name:** Vasarāju aramzemes platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsSpring_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsSpring"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsSpring_r1250.tif  egv_197 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsSpring_r1250.tif")
names(slanis)="egv_197"
slanis2=project(slanis,template100)
writeRaster(slanis2,
    "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsSpring_r1250.tif",
    overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}

```

```

if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsSpring_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.198 FarmlandCrops\_CropsSpring\_r3000

**filename:** FarmlandCrops\_CropsSpring\_r3000.tif

**layername:** egv\_198

**English name:** Fractional cover of Spring Sown Crops within the 3 km landscape

**Latvian name:** Vasarāju aramzemes platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsSpring_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsSpring"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsSpring_r3000.tif  egv_198 -----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsSpring_r3000.tif")
names(slanis)="egv_198"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsSpring_r3000.tif",
            overwrite=TRUE)

```

```

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsSpring_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovidze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovidze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.199 FarmlandCrops\_CropsSpring\_r10000

**filename:** FarmlandCrops\_CropsSpring\_r10000.tif

**layername:** egv\_199

**English name:** Fractional cover of Spring Sown Crops within the 10 km landscape

**Latvian name:** Vasarāju aramzemes platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsSpring_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsSpring"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsSpring_r10000.tif egv_199 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsSpring_r10000.tif")
names(slanis)="egv_199"
slanis2=project(slanis,template100)

```

```

writeRaster(slanis2,
    "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsSpring_r10000.tif",
    overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsSpring_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.200 FarmlandCrops\_CropsWinter\_cell

**filename:** FarmlandCrops\_CropsWinter\_cell.tif

**layername:** egv\_200

**English name:** Fractional cover of Winter Crops within the analysis cell (1 ha)

**Latvian name:** Ziemāju aramzemēs platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, agricultural parcels declared as winter crops are selected from the [Rural Support Service's information on declared fields](#). These geometries are then rasterised to input resolution, ensuring value 1 at the polygon locations and value 0 elsewhere. Rasterisation is performed using the workflow `egvtools::polygon2input()`. Once rasterised, the layer is aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1

```

```

lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# FarmlandCrops_CropsWinter_cell.tif    egv_200 ----
dati=lad %>%
  filter(str_detect(SDM_grupa_sakums,"labība-ziemāji"))
table(dati$SDM_grupa_sakums,useNA="always")

p2i_rez=egvtools::polygon2input(vector_data = dati,
                                 template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                 out_path = "./RasterGrids_10m/2024/",
                                 file_name = "FarmlandCrops_CropsWinter_input.tif",
                                 value_field = "yes",
                                 prepare=FALSE,
                                 background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                 plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "FarmlandCrops_CropsWinter_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/Raw/",
                             outfilename = "FarmlandCrops_CropsWinter_cell.tif",
                             layername = "egv_200",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(p2i_rez)
rm(i2e_rez)
rm(dati)
unlink("./RasterGrids_10m/2024/FarmlandCrops_CropsWinter_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsWinter_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.201 FarmlandCrops\_CropsWinter\_r500

**filename:** FarmlandCrops\_CropsWinter\_r500.tif

**layername:** egv\_201

**English name:** Fractional cover of Winter Crops within the 0.5 km landscape

**Latvian name:** Ziemāju aramzemēs platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsWinter_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsWinter"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsWinter_r500.tif    egv_201 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsWinter_r500.tif")
names(slanis)="egv_201"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsWinter_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsWinter_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.202 FarmlandCrops\_CropsWinter\_r1250

**filename:** FarmlandCrops\_CropsWinter\_r1250.tif

**layername:** egv\_202

**English name:** Fractional cover of Winter Crops within the 1.25 km landscape

**Latvian name:** Ziemāju aramzemes platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes:::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsWinter_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsWinter"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsWinter_r1250.tif  egv_202 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsWinter_r1250.tif")
names(slanis)="egv_202"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsWinter_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsWinter_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.203 FarmlandCrops\_CropsWinter\_r3000

**filename:** FarmlandCrops\_CropsWinter\_r3000.tif

**layername:** egv\_203

**English name:** Fractional cover of Winter Crops within the 3 km landscape

**Latvian name:** Ziemāju aramzemēs platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsWinter_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsWinter"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsWinter_r3000.tif  egv_203 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsWinter_r3000.tif")
names(slanis)="egv_203"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsWinter_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsWinter_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.204 FarmlandCrops\_CropsWinter\_r10000

**filename:** FarmlandCrops\_CropsWinter\_r10000.tif

**layername:** egv\_204

**English name:** Fractional cover of Winter Crops within the 10 km landscape

**Latvian name:** Ziemāju aramzemēs platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsWinter_cell.tif"),
  layer_prefixes = c("FarmlandCrops_CropsWinter"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_CropsWinter_r10000.tif egv_204 ----
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsWinter_r10000.tif")
names(slanis)="egv_204"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandCrops_CropsWinter_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_CropsWinter_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.205 FarmlandCrops\_RapeseedsSpring\_cell

**filename:** FarmlandCrops\_RapeseedsSpring\_cell.tif

**layername:** egv\_205

**English name:** Fractional cover of Spring Sown Rapeseed, Turnip, Corn within the analysis cell (1 ha)

**Latvian name:** Vasaras rapša, ripša, kukurūzas platība analīzes šūnā (1 ha)

**Procedure:** First, agricultural parcels declared as spring sown rapeseed, turnip or corn are selected from the [Rural Support Service's information on declared fields](#). These geometries are then rasterised to input resolution, ensuring value 1 at the polygon locations and value 0 elsewhere. Rasterisation is performed using the workflow `egvtools::polygon2input()`. Once rasterised, the layer is aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# FarmlandCrops_RapeseedsSpring_cell.tif      egv_205 ----
dati=lad %>%
  filter(str_detect(SDM_grupa_sakums,"vasaras rapsis"))
table(dati$SDM_grupa_sakums,useNA="always")

p2i_rez=egvtools::polygon2input(vector_data = dati,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "FarmlandCrops_RapeseedsSpring_input.tif",
  value_field = "yes",
  prepare=FALSE,
  background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
  plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
```

```

        "FarmlandCrops_RapeseedsSpring_input.tif"),
egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
summary_function = "average",
missing_job = "FillOutput",
outlocation = "./RasterGrids_100m/2024/RAW/",
outfile = "FarmlandCrops_RapeseedsSpring_cell.tif",
layername = "egv_205",
idw_weight = 2,
plot_gaps = FALSE, plot_final = TRUE)

i2e_rez
rm(p2i_rez)
rm(i2e_rez)
rm(dati)
unlink("./RasterGrids_10m/2024/FarmlandCrops_RapeseedsSpring_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_RapeseedsSpring_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.206 FarmlandCrops\_RapeseedsSpring\_r500

**filename:** FarmlandCrops\_RapeseedsSpring\_r500.tif

**layername:** egv\_206

**English name:** Fractional cover of Spring Sown Rapeseed, Turnip, Corn within the 0.5 km landscape

**Latvian name:** Vasaras rapša, ripša, kukurūzas platība 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",

```

```

input_layers = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsSpring_cell.tif"),
layer_prefixes = c("FarmlandCrops_RapeseedsSpring"),
output_dir = "./RasterGrids_100m/2024/RAW/",
n_workers = 6,
radii = c("r500"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# FarmlandCrops_RapeseedsSpring_r500.tif    egv_206
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsSpring_r500.tif")
names(slanis)="egv_206"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsSpring_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_RapeseedsSpring_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.207 FarmlandCrops\_RapeseedsSpring\_r1250

**filename:** FarmlandCrops\_RapeseedsSpring\_r1250.tif

**layername:** egv\_207

**English name:** Fractional cover of Spring Sown Rapeseed, Turnip, Corn within the 1.25 km landscape

**Latvian name:** Vasaras rapša, ripša, kukurūzas platība 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",

```

```

radii_path = "./Templates/TemplateGridPoints/tiles/",
tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsSpring_cell.tif"),
layer_prefixes = c("FarmlandCrops_RapeseedsSpring"),
output_dir = "./RasterGrids_100m/2024/RAW/",
n_workers = 6,
radii = c("r1250"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# FarmlandCrops_RapeseedsSpring_r1250.tif  egv_207
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsSpring_r1250.tif")
names(slanis)="egv_207"
slanis2=project(slanis,template100)
writeRaster(slanis2,
           "./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsSpring_r1250.tif",
           overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_RapeseedsSpring_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.208 FarmlandCrops\_RapeseedsSpring\_r3000

**filename:** FarmlandCrops\_RapeseedsSpring\_r3000.tif

**layername:** egv\_208

**English name:** Fractional cover of Spring Sown Rapeseed, Turnip, Corn within the 3 km landscape

**Latvian name:** Vasaras rapša, ripša, kukurūzas platība 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

```

```

# radii ----
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path  = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsSpring_cell.tif"),
  layer_prefixes = c("FarmlandCrops_RapeseedsSpring"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_RapeseedsSpring_r3000.tif  egv_208
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsSpring_r3000.tif")
names(slanis)="egv_208"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsSpring_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_RapeseedsSpring_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.209 FarmlandCrops\_RapeseedsSpring\_r10000

**filename:** FarmlandCrops\_RapeseedsSpring\_r10000.tif

**layername:** egv\_209

**English name:** Fractional cover of Spring Sown Rapeseed, Turnip, Corn within the 10 km landscape

**Latvian name:** Vasaras rapša, ripša, kukurūzas platība 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

```

```

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsSpring_cell.tif"),
  layer_prefixes = c("FarmlandCrops_RapeseedsSpring"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_RapeseedsSpring_r10000.tif egv_209
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsSpring_r10000.tif")
names(slanis)="egv_209"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsSpring_r10000.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_RapeseedsSpring_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.210 FarmlandCrops\_RapeseedsWinter\_cell

**filename:** FarmlandCrops\_RapeseedsWinter\_cell.tif

**layername:** egv\_210

**English name:** Fractional cover of Winter Rapeseed, Turnip within the analysis cell (1 ha)

**Latvian name:** Ziemas rapša, ripša platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, agricultural parcels declared as winter rapeseed or turnip are selected from the [Rural Support Service's information on declared fields](#). These geometries are then rasterised to input resolution, ensuring value 1 at the polygon locations and value 0 elsewhere. Rasterisation is performed using the workflow `egvtools::polygon2input()`. Once rasterised, the layer is aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# FarmlandCrops_RapeseedsWinter_cell.tif      egv_210 ----
dati=lad %>%
  filter(str_detect(SDM_grupa_sakums,"ziemas rapsis"))
table(dati$SDM_grupa_sakums,useNA="always")

p2i_rez=egvtools::polygon2input(vector_data = dati,
                                 template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                 out_path = "./RasterGrids_10m/2024/",
                                 file_name = "FarmlandCrops_RapeseedsWinter_input.tif",
                                 value_field = "yes",
                                 prepare=FALSE,
                                 background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                 plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "FarmlandCrops_RapeseedsWinter_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/Raw/",
                             outfilename = "FarmlandCrops_RapeseedsWinter_cell.tif",
                             layername = "egv_210",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(p2i_rez)
rm(i2e_rez)
rm(dati)
unlink("./RasterGrids_10m/2024/FarmlandCrops_RapeseedsWinter_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}

```

```

if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_RapeseedsWinter_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.211 FarmlandCrops\_RapeseedsWinter\_r500

**filename:** FarmlandCrops\_RapeseedsWinter\_r500.tif

**layername:** egv\_211

**English name:** Fractional cover of Winter Rapeseed, Turnip within the 0.5 km landscape

**Latvian name:** Ziemas rapša, ripša platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsWinter_cell.tif"),
  layer_prefixes = c("FarmlandCrops_RapeseedsWinter"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_RapeseedsWinter_r500.tif      egv_211
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsWinter_r500.tif")
names(slanis)="egv_211"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsWinter_r500.tif",
            overwrite=TRUE)

```

```

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_RapeseedsWinter_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovidze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovidze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.212 FarmlandCrops\_RapeseedsWinter\_r1250

**filename:** FarmlandCrops\_RapeseedsWinter\_r1250.tif

**layername:** egv\_212

**English name:** Fractional cover of Winter Rapeseed, Turnip within the 1.25 km landscape

**Latvian name:** Ziemas rapša, ripša platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsWinter_cell.tif"),
  layer_prefixes = c("FarmlandCrops_RapeseedsWinter"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_RapeseedsWinter_r1250.tif  egv_212
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsWinter_r1250.tif")
names(slanis)="egv_212"
slanis2=project(slanis,template100)

```

```

writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsWinter_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_RapeseedsWinter_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.213 FarmlandCrops\_RapeseedsWinter\_r3000

**filename:** FarmlandCrops\_RapeseedsWinter\_r3000.tif

**layername:** egv\_213

**English name:** Fractional cover of Winter Rapeseed, Turnip within the 3 km landscape

**Latvian name:** Ziemas rapša, ripša platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsWinter_cell.tif"),
  layer_prefixes = c("FarmlandCrops_RapeseedsWinter"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandCrops_RapeseedsWinter_r3000.tif  egv_213

```

```

slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsWinter_r3000.tif")
names(slanis)="egv_213"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsWinter_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_RapeseedsWinter_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.214 FarmlandCrops\_RapeseedsWinter\_r10000

**filename:** FarmlandCrops\_RapeseedsWinter\_r10000.tif

**layername:** egv\_214

**English name:** Fractional cover of Winter Rapeseed, Turnip within the 10 km landscape

**Latvian name:** Ziemas rapša, ripša platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsWinter_cell.tif"),
  layer_prefixes = c("FarmlandCrops_RapeseedsWinter"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

```

```

# FarmlandCrops_RapeseedsWinter_r10000.tif  egv_214
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsWinter_r10000.tif")
names(slanis)="egv_214"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandCrops_RapeseedsWinter_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandCrops_RapeseedsWinter_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.215 FarmlandGrassland\_GrasslandsAbandoned\_cell

**filename:** FarmlandGrassland\_GrasslandsAbandoned\_cell.tif

**layername:** egv\_215

**English name:** Fractional cover of Abandoned Grassland within the analysis cell (1 ha)

**Latvian name:** Neapsaimniekotu zālāju platības īpatsvars analīzē šūnā (1 ha)

**Procedure:** First, the grasslands from the [Landscape classification](#) are selected (value 330 reclassified as value 1, others as NA). Next, agricultural parcels declared as grasslands are selected from the [Rural Support Service's information on declared fields](#). Next, cells with grasslands in [Landscape classification](#) but not in the [Rural Support Service's information on declared fields](#) are selected and matched to input layer. Once matched, the layer is aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

```

```

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# FarmlandGrassland_GrasslandsAbandoned_cell.tif      egv_215 ----
landscape_grasslands=ifel(simple_landscape==330,1,0)

dati=lad %>%
  filter(str_detect(SDM_grupa_sakums,"zālāji"))
table(dati$SDM_grupa_sakums,useNA="always")

lad_zalajiem=fasterize(dati,rastrs10,field="yes",fun="first")
lad_zalaji=rast(lad_zalajiem)

abandoned=ifel(landscape_grasslands==1&is.na(lad_zalaji),1,0)
plot(abandoned)

i2e_rez=egvtools::input2egv(input=abandoned,
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/RAW/",
  outfilename = "FarmlandGrassland_GrasslandsAbandoned_cell.tif",
  layername = "egv_215",
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(i2e_rez)
rm(dati)
rm(lad_zalajiem)
rm(lad_zalaji)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsAbandoned_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.216 FarmlandGrassland\_GrasslandsAbandoned\_r500

**filename:** FarmlandGrassland\_GrasslandsAbandoned\_r500.tif

**layername:** egv\_216

**English name:** Fractional cover of Abandoned Grassland within the 0.5 km landscape

**Latvian name:** Neapsaimniekotu zālāju platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   =
    c("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAbandoned_cell.tif"),
  layer_prefixes = c("FarmlandGrassland_GrasslandsAbandoned"),
  output_dir     = "./RasterGrids_100m/2024/RAW/",
  n_workers      = 6,
  radii          = c("r500"),
  radius_mode    = "sparse",
  extract_fun    = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandGrassland_GrasslandsAbandoned_r500.tif    egv_216
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAbandoned_r500.tif")
names(slanis)="egv_216"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAbandoned_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsAbandoned_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.217 FarmlandGrassland\_GrasslandsAbandoned\_r1250

**filename:** FarmlandGrassland\_GrasslandsAbandoned\_r1250.tif

**layername:** egv\_217

**English name:** Fractional cover of Abandoned Grassland within the 1.25 km landscape

**Latvian name:** Neapsaimniekotu zālāju platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAbandoned_cell.tif"),
  layer_prefixes = c("FarmlandGrassland_GrasslandsAbandoned"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandGrassland_GrasslandsAbandoned_r1250.tif  egv_217
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAbandoned_r1250.tif")
names(slanis)="egv_217"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAbandoned_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsAbandoned_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
```

```
overwrite=TRUE
```

## 6.218 FarmlandGrassland\_GrasslandsAbandoned\_r3000

**filename:** FarmlandGrassland\_GrasslandsAbandoned\_r3000.tif

**layername:** egv\_218

**English name:** Fractional cover of Abandoned Grassland within the 3 km landscape

**Latvian name:** Neapsaimniekotu zālāju platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAbandoned_cell.tif"),
  layer_prefixes = c("FarmlandGrassland_GrasslandsAbandoned"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandGrassland_GrasslandsAbandoned_r3000.tif  egv_218
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAbandoned_r3000.tif")
names(slanis)="egv_218"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAbandoned_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsAbandoned_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
```

```

standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.219 FarmlandGrassland\_GrasslandsAbandoned\_r10000

**filename:** FarmlandGrassland\_GrasslandsAbandoned\_r10000.tif

**layername:** egv\_219

**English name:** Fractional cover of Abandoned Grassland within the 10 km landscape

**Latvian name:** Neapsaimniekotu zālāju platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAbandoned_cell.tif"),
  layer_prefixes = c("FarmlandGrassland_GrasslandsAbandoned"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing  = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandGrassland_GrasslandsAbandoned_r10000.tif egv_219
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAbandoned_r10000.tif")
names(slanis)="egv_219"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAbandoned_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsAbandoned_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)

```

```

saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.220 FarmlandGrassland\_GrasslandsAll\_cell

**filename:** FarmlandGrassland\_GrasslandsAll\_cell.tif

**layername:** egv\_220

**English name:** Fractional cover of any Grassland within the analysis cell (1 ha)

**Latvian name:** Zālāju (visu veidu) platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, the grasslands from the [Landscape classification](#) are selected (value 330 reclassified to value 1, others as 0). Once selected, the layer is aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# FarmlandGrassland_GrasslandsAll_cell.tif  egv_220 ----
landscape_grasslands=ifel(simple_landscape==330,1,0)

i2e_rez=egvtools::input2egv(input=landscape_grasslands,
                            egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",

```

```

        summary_function = "average",
        missing_job = "FillOutput",
        outlocation = "./RasterGrids_100m/2024/RAW/",
        outfilename = "FarmlandGrassland_GrasslandsAll_cell.tif",
        layername = "egv_220",
        idw_weight = 2,
        plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(i2e_rez)
rm(landscape_grasslands)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsAll_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.221 FarmlandGrassland\_GrasslandsAll\_r500

**filename:** FarmlandGrassland\_GrasslandsAll\_r500.tif

**layername:** egv\_221

**English name:** Fractional cover of any Grassland within the 0.5 km landscape

**Latvian name:** Zālāju (visu veidu) platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAll_cell.tif"),
  layer_prefixes = c("FarmlandGrassland_GrasslandsAll"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",

```

```

extract_fun  = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3)

# FarmlandGrassland_GrasslandsAll_r500.tif  egv_221
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAll_r500.tif")
names(slanis)="egv_221"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAll_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsAll_r500.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.222 FarmlandGrassland\_GrasslandsAll\_r1250

**filename:** FarmlandGrassland\_GrasslandsAll\_r1250.tif

**layername:** egv\_222

**English name:** Fractional cover of any Grassland within the 1.25 km landscape

**Latvian name:** Zālāju (visu veidu) platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAll_cell.tif"),
  layer_prefixes = c("FarmlandGrassland_GrasslandsAll"),
  output_dir    = "./RasterGrids_100m/2024/RAW/")

```

```

n_workers    = 6,
radii        = c("r1250"),
radius_mode  = "sparse",
extract_fun   = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3)

# FarmlandGrassland_GrasslandsAll_r1250.tif evg_222
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAll_r1250.tif")
names(slanis)="evg_222"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAll_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsAll_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.223 FarmlandGrassland\_GrasslandsAll\_r3000

**filename:** FarmlandGrassland\_GrasslandsAll\_r3000.tif

**layername:** evg\_223

**English name:** Fractional cover of any Grassland within the 3 km landscape

**Latvian name:** Zālāju (visu veidu) platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",

```

```

input_layers = c("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAll_cell.tif"),
layer_prefixes = c("FarmlandGrassland_GrasslandsAll"),
output_dir = "./RasterGrids_100m/2024/RAW/",
n_workers = 6,
radii = c("r3000"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# FarmlandGrassland_GrasslandsAll_r3000.tif evg_223
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAll_r3000.tif")
names(slanis)="evg_223"
slanis2=project(slanis,template100)
writeRaster(slanis2,
    "./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAll_r3000.tif",
    overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsAll_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.224 FarmlandGrassland\_GrasslandsAll\_r10000

**filename:** FarmlandGrassland\_GrasslandsAll\_r10000.tif

**layername:** evg\_224

**English name:** Fractional cover of any Grassland within the 10 km landscape

**Latvian name:** Zālāju (visu veidu) platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",

```

```

radii_path = "./Templates/TemplateGridPoints/tiles/",
tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers = c("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAll_cell.tif"),
layer_prefixes = c("FarmlandGrassland_GrasslandsAll"),
output_dir = "./RasterGrids_100m/2024/RAW/",
n_workers = 6,
radii = c("r10000"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# FarmlandGrassland_GrasslandsAll_r10000.tif    egv_224
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAll_r10000.tif")
names(slanis)="egv_224"
slanis2=project(slanis,template100)
writeRaster(slanis2,
           "./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsAll_r10000.tif",
           overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsAll_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.225 FarmlandGrassland\_GrasslandsPermanent\_cell

**filename:** FarmlandGrassland\_GrasslandsPermanent\_cell.tif

**layername:** egv\_225

**English name:** Fractional cover of Permanent Grassland within the analysis cell (1 ha)

**Latvian name:** Ilggadīgu zālāju platības īpatsvars analīzē šūnā (1 ha)

**Procedure:** First, agricultural parcels declared as permanent grasslands are selected from the [Rural Support Service's information on declared fields](#). These geometries are then rasterised to input resolution, ensuring value 1 at the polygon locations and value 0 elsewhere. Rasterisation is performed with the workflow `egvtools::polygon2input()`. Once rasterised, the layer is aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# FarmlandGrassland_GrasslandsPermanent_cell.tif      egv_225 ----
dati=lad %>%
  filter(SDM_grupa_sakums=="zālāji (ilggadīgie)")

table(dati$SDM_grupa_sakums,useNA="always")

p2i_rez=egvtools::polygon2input(vector_data = dati,
                                 template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                 out_path = "./RasterGrids_10m/2024/",
                                 file_name = "FarmlandGrassland_GrasslandsPermanent_input.tif",
                                 value_field = "yes",
                                 prepare=FALSE,
                                 background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                 plot_result = TRUE)

p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "FarmlandGrassland_GrasslandsPermanent_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/Raw/",
                             outfilename = "FarmlandGrassland_GrasslandsPermanent_cell.tif",
                             layername = "egv_225",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(p2i_rez)
rm(i2e_rez)
rm(dati)
unlink("./RasterGrids_10m/2024/FarmlandGrassland_GrasslandsPermanent_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsPermanent_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)

```

```

videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.226 FarmlandGrassland\_GrasslandsPermanent\_r500

**filename:** FarmlandGrassland\_GrasslandsPermanent\_r500.tif

**layername:** egv\_226

**English name:** Fractional cover of Permanent Grassland within the 0.5 km landscape

**Latvian name:** Ilggadīgu zālāju platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsPermanent_cell.tif"),
  layer_prefixes = c("FarmlandGrassland_GrasslandsPermanent"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing  = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandGrassland_GrasslandsPermanent_r500.tif      egv_226
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsPermanent_r500.tif")
names(slanis)="egv_226"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsPermanent_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

nosaukums="FarmlandGrassland_GrasslandsPermanent_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.227 FarmlandGrassland\_GrasslandsPermanent\_r1250

**filename:** FarmlandGrassland\_GrasslandsPermanent\_r1250.tif

**layername:** egv\_227

**English name:** Fractional cover of Permanent Grassland within the 1.25 km landscape

**Latvian name:** Ilggadīgu zālāju platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsPermanent_cell.tif"),
  layer_prefixes = c("FarmlandGrassland_GrasslandsPermanent"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandGrassland_GrasslandsPermanent_r1250.tif  egv_227
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsPermanent_r1250.tif")
names(slanis)="egv_227"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsPermanent_r1250.tif",
  overwrite=TRUE)

```

```

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsPermanent_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.228 FarmlandGrassland\_GrasslandsPermanent\_r3000

**filename:** FarmlandGrassland\_GrasslandsPermanent\_r3000.tif

**layername:** egv\_228

**English name:** Fractional cover of Permanent Grassland within the 3 km landscape

**Latvian name:** Ilggadīgu zālāju platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsPermanent_cell.tif"),
  layer_prefixes = c("FarmlandGrassland_GrasslandsPermanent"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandGrassland_GrasslandsPermanent_r3000.tif  egv_228
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsPermanent_r3000.tif")
names(slanis)="egv_228"
slanis2=project(slanis,template100)

```

```

writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsPermanent_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsPermanent_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.229 FarmlandGrassland\_GrasslandsPermanent\_r10000

**filename:** FarmlandGrassland\_GrasslandsPermanent\_r10000.tif

**layername:** evg\_229

**English name:** Fractional cover of Permanent Grassland within the 10 km landscape

**Latvian name:** Ilggadīgu zālāju platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsPermanent_cell.tif"),
  layer_prefixes = c("FarmlandGrassland_GrasslandsPermanent"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing  = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

```

```

# FarmlandGrassland_GrasslandsPermanent_r10000.tif  egv_229
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsPermanent_r10000.tif")
names(slanis)="egv_229"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsPermanent_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsPermanent_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.230 FarmlandGrassland\_GrasslandsTemporary\_cell

**filename:** FarmlandGrassland\_GrasslandsTemporary\_cell.tif

**layername:** egv\_230

**English name:** Fractional cover of Temporary Grassland within the analysis cell (1 ha)

**Latvian name:** Zālāju-aramzemē platības īpatsvars analīzēs šūnā (1 ha)

**Procedure:** First, agricultural parcels declared as grasslands in arable lands are selected from the [Rural Support Service's information on declared fields](#). These geometries are then rasterised to input resolution, ensuring value 1 at the polygon locations and value 0 elsewhere. Rasterisation is performed with the workflow `egvtools::polygon2input()`. Once rasterised, the layer is aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")

```

```

kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# FarmlandGrassland_GrasslandsTemporary_cell.tif      egv_230 ----
dati=lad %>%
  filter(str_detect(SDM_grupa_sakums,"kultivēt"))
table(dati$SDM_grupa_sakums,useNA="always")

p2i_rez=egvtools::polygon2input(vector_data = dati,
                                 template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                 out_path = "./RasterGrids_10m/2024/",
                                 file_name = "FarmlandGrassland_GrasslandsTemporary_input.tif",
                                 value_field = "yes",
                                 prepare=FALSE,
                                 background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                 plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "FarmlandGrassland_GrasslandsTemporary_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/Raw/",
                             outfilename = "FarmlandGrassland_GrasslandsTemporary_cell.tif",
                             layername = "egv_230",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(p2i_rez)
rm(i2e_rez)
rm(dati)
unlink("./RasterGrids_10m/2024/FarmlandGrassland_GrasslandsTemporary_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsTemporary_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.231 FarmlandGrassland\_GrasslandsTemporary\_r500

**filename:** FarmlandGrassland\_GrasslandsTemporary\_r500.tif

**layername:** egv\_231

**English name:** Fractional cover of Temporary Grassland within the 0.5 km landscape

**Latvian name:** Zālāju-aramzemē platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   =
    c("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsTemporary_cell.tif"),
  layer_prefixes = c("FarmlandGrassland_GrasslandsTemporary"),
  output_dir     = "./RasterGrids_100m/2024/RAW/",
  n_workers      = 6,
  radii          = c("r500"),
  radius_mode    = "sparse",
  extract_fun    = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandGrassland_GrasslandsTemporary_r500.tif    egv_231
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsTemporary_r500.tif")
names(slanis)="egv_231"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsTemporary_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsTemporary_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.232 FarmlandGrassland\_GrasslandsTemporary\_r1250

**filename:** FarmlandGrassland\_GrasslandsTemporary\_r1250.tif

**layername:** evg\_232

**English name:** Fractional cover of Temporary Grassland within the 1.25 km landscape

**Latvian name:** Zālāju-aramzemē platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsTemporary_cell.tif"),
  layer_prefixes = c("FarmlandGrassland_GrasslandsTemporary"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandGrassland_GrasslandsTemporary_r1250.tif  evg_232
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsTemporary_r1250.tif")
names(slanis)="evg_232"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsTemporary_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsTemporary_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
```

```
overwrite=TRUE)
```

## 6.233 FarmlandGrassland\_GrasslandsTemporary\_r3000

**filename:** FarmlandGrassland\_GrasslandsTemporary\_r3000.tif

**layername:** egv\_233

**English name:** Fractional cover of Temporary Grassland within the 3 km landscape

**Latvian name:** Zālāju-aramzemē platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsTemporary_cell.tif"),
  layer_prefixes = c("FarmlandGrassland_GrasslandsTemporary"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandGrassland_GrasslandsTemporary_r3000.tif  egv_233
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsTemporary_r3000.tif")
names(slanis)="egv_233"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsTemporary_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsTemporary_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
```

```

standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.234 FarmlandGrassland\_GrasslandsTemporary\_r10000

**filename:** FarmlandGrassland\_GrasslandsTemporary\_r10000.tif

**layername:** egv\_234

**English name:** Fractional cover of Temporary Grassland within the 10 km landscape

**Latvian name:** Zālāju-aramzemē platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsTemporary_cell.tif"),
  layer_prefixes = c("FarmlandGrassland_GrasslandsTemporary"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing  = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandGrassland_GrasslandsTemporary_r10000.tif egv_234
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsTemporary_r10000.tif")
names(slanis)="egv_234"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandGrassland_GrasslandsTemporary_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandGrassland_GrasslandsTemporary_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)

```

```

saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.235 FarmlandParcels\_FieldsActive\_cell

**filename:** FarmlandParcels\_FieldsActive\_cell.tif

**layername:** egv\_235

**English name:** Fractional cover of Agricultural Land Parcels within the analysis cell (1 ha)

**Latvian name:** Lauku bloku platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, agricultural parcels from the [Rural Support Service's information on declared fields](#) are rasterised to input resolution, ensuring value 1 at the polygon locations and value 0 elsewhere. Rasterisation is performed with the workflow `egvtools::polygon2input()`. Once rasterised, the layer is aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# FarmlandParcels_FieldsActive_cell.tif egv_235 ----
p2i_rez=egvtools::polygon2input(vector_data = lad,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
```

```

        file_name = "FarmlandParcels_FieldsActive_input.tif",
        value_field = "yes",
        prepare=FALSE,
        background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
        plot_result = TRUE)

p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "FarmlandParcels_FieldsActive_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/Raw/",
                             outfilename = "FarmlandParcels_FieldsActive_cell.tif",
                             layername = "egv_235",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(p2i_rez)
rm(i2e_rez)
rm(dat)
unlink("./RasterGrids_10m/2024/FarmlandParcels_FieldsActive_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandParcels_FieldsActive_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.236 FarmlandParcels\_FieldsActive\_r500

**filename:** FarmlandParcels\_FieldsActive\_r500.tif

**layername:** egv\_236

**English name:** Fractional cover of Agricultural Land Parcels within the 0.5 km landscape

**Latvian name:** Lauku bloku platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

```

```

# radii ----
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path  = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   = c("./RasterGrids_100m/2024/RAW/FarmlandParcels_FieldsActive_cell.tif"),
  layer_prefixes = c("FarmlandParcels_FieldsActive"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandParcels_FieldsActive_r500.tif egv_236
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandParcels_FieldsActive_r500.tif")
names(slanis)="egv_236"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandParcels_FieldsActive_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandParcels_FieldsActive_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.237 FarmlandParcels\_FieldsActive\_r1250

**filename:** FarmlandParcels\_FieldsActive\_r1250.tif

**layername:** egv\_237

**English name:** Fractional cover of Agricultural Land Parcels within the 1.25 km landscape

**Latvian name:** Lauku bloku platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

```

```

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandParcels_FieldsActive_cell.tif"),
  layer_prefixes = c("FarmlandParcels_FieldsActive"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandParcels_FieldsActive_r1250.tif    egv_237
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandParcels_FieldsActive_r1250.tif")
names(slanis)="egv_237"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandParcels_FieldsActive_r1250.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandParcels_FieldsActive_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.238 FarmlandParcels\_FieldsActive\_r3000

**filename:** FarmlandParcels\_FieldsActive\_r3000.tif

**layername:** egv\_238

**English name:** Fractional cover of Agricultural Land Parcels within the 3 km landscape

**Latvian name:** Lauku bloku platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs -----
if(!require(terra)) {install.packages("terra"); require(terra)}

```

```

if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandParcels_FieldsActive_cell.tif"),
  layer_prefixes = c("FarmlandParcels_FieldsActive"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandParcels_FieldsActive_r3000.tif      egv_238
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandParcels_FieldsActive_r3000.tif")
names(slanis)="egv_238"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandParcels_FieldsActive_r3000.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandParcels_FieldsActive_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.239 FarmlandParcels\_FieldsActive\_r10000

**filename:** FarmlandParcels\_FieldsActive\_r10000.tif

**layername:** egv\_239

**English name:** Fractional cover of Agricultural Land Parcels within the 10 km landscape

**Latvian name:** Lauku bloku platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandParcels_FieldsActive_cell.tif"),
  layer_prefixes = c("FarmlandParcels_FieldsActive"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandParcels_FieldsActive_r10000.tif  egv_239
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandParcels_FieldsActive_r10000.tif")
names(slanis)="egv_239"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandParcels_FieldsActive_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandParcels_FieldsActive_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.240 FarmlandPloughed\_CropsFallow\_cell

**filename:** FarmlandPloughed\_CropsFallow\_cell.tif

**layername:** egv\_240

**English name:** Fractional cover of Crop-, Fallow- Land within the analysis cell (1 ha)

**Latvian name:** Aramzemju, papuvju platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, agricultural parcels declared as crops or fallow land are selected from the [Rural Support Service's information on declared fields](#). Geometries are then rasterised to input resolution, ensuring value 1 at the polygon locations and value 0 elsewhere. Rasterisation is performed using the workflow `egvtools::polygon2input()`. Once rasterised, the layer is aggregated to EGV resolution using the workflow

`egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# FarmlandPloughed_CropsFallow_cell.tif egv_240 ----
dati=lad %>%
  filter(SDM_grupa_sakums %in% c("aramzemes (citur neiekļautās)",
                                   "aramzemes (labība-vasarāji)",
                                   "aramzemes (labība-ziemāji)",
                                   "aramzemes (vagu un rušināmkultūru)",
                                   "aramzemes (vasaras rapsis un risspis, kukurūzas, zirņi un pupas, soja,
                                   ← kaņepes)",
                                   "aramzemes (ziemas rapsis un risspis)",
                                   "papuves"))
table(dati$SDM_grupa_sakums,useNA="always")

p2i_rez=egvtools::polygon2input(vector_data = dati,
                                 template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                 out_path = "./RasterGrids_10m/2024/",
                                 file_name = "FarmlandPloughed_CropsFallow_input.tif",
                                 value_field = "yes",
                                 prepare=FALSE,
                                 background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                 plot_result = TRUE)

p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "FarmlandPloughed_CropsFallow_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/RAW/",
                             outfilename = "FarmlandPloughed_CropsFallow_cell.tif",
                             layername = "egv_240",
```

```

        idw_weight = 2,
        plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(p2i_rez)
rm(i2e_rez)
rm(dat)
unlink("./RasterGrids_10m/2024/FarmlandPloughed_CropsFallow_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandPloughed_CropsFallow_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.241 FarmlandPloughed\_CropsFallow\_r500

**filename:** FarmlandPloughed\_CropsFallow\_r500.tif

**layername:** evg\_241

**English name:** Fractional cover of Crop-, Fallow- Land within the 0.5 km landscape

**Latvian name:** Aramzemju, papuvju platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_10m/2024/RAW/FarmlandPloughed_CropsFallow_cell.tif"),
  layer_prefixes = c("FarmlandPloughed_CropsFallow"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing  = TRUE,

```

```

IDW_weight = 2,
future_max_size = 40 * 1024^3)

# FarmlandPloughed_CropsFallow_r500.tif egv_241
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallow_r500.tif")
names(slanis)="egv_241"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallow_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandPloughed_CropsFallow_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.242 FarmlandPloughed\_CropsFallow\_r1250

**filename:** FarmlandPloughed\_CropsFallow\_r1250.tif

**layername:** egv\_242

**English name:** Fractional cover of Crop-, Fallow- Land within the 1.25 km landscape

**Latvian name:** Aramzemju, papuvju platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path  = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   = c("./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallow_cell.tif"),
  layer_prefixes = c("FarmlandPloughed_CropsFallow"),
  output_dir     = "./RasterGrids_100m/2024/RAW/",
  n_workers      = 6,
  radii         = c("r1250"),

```

```

radius_mode  = "sparse",
extract_fun  = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3)

# FarmlandPloughed_CropsFallow_r1250.tif    egv_242
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallow_r1250.tif")
names(slanis)="egv_242"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallow_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandPloughed_CropsFallow_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.243 FarmlandPloughed\_CropsFallow\_r3000

**filename:** FarmlandPloughed\_CropsFallow\_r3000.tif

**layername:** egv\_243

**English name:** Fractional cover of Crop-, Fallow- Land within the 3 km landscape

**Latvian name:** Aramzemju, papuvju platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path  = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   = c("./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallow_cell.tif"),
  layer_prefixes = c("FarmlandPloughed_CropsFallow"),

```

```

output_dir  = "./RasterGrids_100m/2024/RAW/",
n_workers   = 6,
radii       = c("r3000"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight  = 2,
future_max_size = 40 * 1024^3)

# FarmlandPloughed_CropsFallow_r3000.tif    egv_243
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallow_r3000.tif")
names(slanis)="egv_243"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallow_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandPloughed_CropsFallow_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.244 FarmlandPloughed\_CropsFallow\_r10000

**filename:** FarmlandPloughed\_CropsFallow\_r10000.tif

**layername:** egv\_244

**English name:** Fractional cover of Crop-, Fallow- Land within the 10 km landscape

**Latvian name:** Aramzemju, papuvju platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",

```

```

template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers = c("./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallow_cell.tif"),
layer_prefixes = c("FarmlandPloughed_CropsFallow"),
output_dir = "./RasterGrids_100m/2024/RAW/",
n_workers = 6,
radii = c("r10000"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# FarmlandPloughed_CropsFallow_r10000.tif  egv_244
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallow_r10000.tif")
names(slanis)="egv_244"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallow_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandPloughed_CropsFallow_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.245 FarmlandPloughed\_CropsFallowTempGrass\_cell

**filename:** FarmlandPloughed\_CropsFallowTempGrass\_cell.tif

**layername:** egv\_245

**English name:** Fractional cover of Crop-, Fallow-, Temporary Grass- Lands within the analysis cell (1 ha)

**Latvian name:** Aramzemju, papuvju, zālāju-aramzemē platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, agricultural parcels declared as crops, fallow land or grasslands in arable land are selected from the [Rural Support Service's information on declared fields](#). Geometries are then rasterised to input resolution, ensuring value 1 at the polygon locations and value 0 elsewhere. Rasterisation is performed using the workflow `egvtools::polygon2input()`. Once rasterised, the layer is aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}

```

```

if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# FarmlandPloughed_CropsFallowTempGrass_cell.tif      egv_245 ----
dati=lad %>%
  filter(SDM_grupa_sakums %in% c("aramzemes (citur neiekļautās)",
                                   "aramzemes (labiba-vasarāji)",
                                   "aramzemes (labiba-ziemāji)",
                                   "aramzemes (vagu un rušināmkultūru)",
                                   "aramzemes (vasaras rapsis un rīspsis, kukurūzas, zirņi un pupas, soja,
                                   ← kaņepes)",
                                   "aramzemes (ziemas rapsis un rīpsis)",
                                   "papuves",
                                   "zālāji (kultivētie)"))
table(dati$SDM_grupa_sakums,useNA="always")

p2i_rez=egvtools::polygon2input(vector_data = dati,
                                 template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                 out_path = "./RasterGrids_10m/2024/",
                                 file_name = "FarmlandPloughed_CropsFallowTempGrass_input.tif",
                                 value_field = "yes",
                                 prepare=FALSE,
                                 background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                 plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "FarmlandPloughed_CropsFallowTempGrass_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/Raw/",
                             outfilename = "FarmlandPloughed_CropsFallowTempGrass_cell.tif",
                             layername = "egv_245",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(p2i_rez)
rm(i2e_rez)
rm(dati)
unlink("./RasterGrids_10m/2024/FarmlandPloughed_CropsFallowTempGrass_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

nosaukums="FarmlandPloughed_CropsFallowTempGrass_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.246 FarmlandPloughed\_CropsFallowTempGrass\_r500

**filename:** FarmlandPloughed\_CropsFallowTempGrass\_r500.tif

**layername:** egv\_246

**English name:** Fractional cover of Crop-, Fallow-, Temporary Grass- Lands within the 0.5 km landscape

**Latvian name:** Aramzemju, papuvju, zālāju-aramzemē platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers =
    c("./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallowTempGrass_cell.tif"),
  layer_prefixes = c("FarmlandPloughed_CropsFallowTempGrass"),
  output_dir   = "./RasterGrids_100m/2024/RAW/",
  n_workers    = 6,
  radii        = c("r500"),
  radius_mode  = "sparse",
  extract_fun  = "mean",
  fill_missing  = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandPloughed_CropsFallowTempGrass_r500.tif      egv_246
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallowTempGrass_r500.tif")
names(slanis)="egv_246"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallowTempGrass_r500.tif",
            overwrite=TRUE)

```

```

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandPloughed_CropsFallowTempGrass_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovidze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovidze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.247 FarmlandPloughed\_CropsFallowTempGrass\_r1250

**filename:** FarmlandPloughed\_CropsFallowTempGrass\_r1250.tif

**layername:** egv\_247

**English name:** Fractional cover of Crop-, Fallow-, Temporary Grass- Lands within the 1.25 km landscape

**Latvian name:** Aramzemju, papuvju, zālāju-aramzemē platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/Raw/FarmlandPloughed_CropsFallowTempGrass_cell.tif"),
  layer_prefixes = c("FarmlandPloughed_CropsFallowTempGrass"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandPloughed_CropsFallowTempGrass_r1250.tif  egv_247
slanis=rast("./RasterGrids_100m/2024/Raw/FarmlandPloughed_CropsFallowTempGrass_r1250.tif")
names(slanis)="egv_247"

```

```

slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallowTempGrass_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandPloughed_CropsFallowTempGrass_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.248 FarmlandPloughed\_CropsFallowTempGrass\_r3000

**filename:** FarmlandPloughed\_CropsFallowTempGrass\_r3000.tif

**layername:** evg\_248

**English name:** Fractional cover of Crop-, Fallow-, Temporary Grass- Lands within the 3 km landscape

**Latvian name:** Aramzemju, papuvju, zālāju-aramzemē platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallowTempGrass_cell.tif"),
  layer_prefixes = c("FarmlandPloughed_CropsFallowTempGrass"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

```

```

# FarmlandPloughed_CropsFallowTempGrass_r3000.tif    egv_248
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallowTempGrass_r3000.tif")
names(slanis)="egv_248"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallowTempGrass_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandPloughed_CropsFallowTempGrass_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.249 FarmlandPloughed\_CropsFallowTempGrass\_r10000

**filename:** FarmlandPloughed\_CropsFallowTempGrass\_r10000.tif

**layername:** egv\_249

**English name:** Fractional cover of Crop-, Fallow-, Temporary Grass- Lands within the 10 km landscape

**Latvian name:** Aramzemju, papuvju, zālāju-aramzemē plātības īpatsvars 10 km aimavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallowTempGrass_cell.tif"),
  layer_prefixes = c("FarmlandPloughed_CropsFallowTempGrass"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",

```

```

fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# FarmlandPloughed_CropsFallowTempGrass_r10000.tif egv_249
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallowTempGrass_r10000.tif")
names(slanis)="egv_249"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandPloughed_CropsFallowTempGrass_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandPloughed_CropsFallowTempGrass_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.250 FarmlandPloughed\_Fallow\_cell

**filename:** FarmlandPloughed\_Fallow\_cell.tif

**layername:** egv\_250

**English name:** Fractional cover of Fallow Land within the analysis cell (1 ha)

**Latvian name:** Papuvju platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, agricultural parcels declared as fallow land are selected from the [Rural Support Service's information on declared fields](#). Geometries are then rasterised to input resolution, ensuring value 1 at the polygon locations and value 0 elsewhere. Rasterisation is performed using the workflow `egvtools::polygon2input()`. Once rasterised, the layer is aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

```

```

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# FarmlandPloughed_Fallow_cell.tif  egv_250 ----
dati=lad %>%
  filter(SDM_grupa_sakums == "papuves")
table(dati$SDM_grupa_sakums,useNA="always")

p2i_rez=egvtools::polygon2input(vector_data = dati,
                                 template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                 out_path = "./RasterGrids_10m/2024/",
                                 file_name = "FarmlandPloughed_Fallow_input.tif",
                                 value_field = "yes",
                                 prepare=FALSE,
                                 background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                 plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "FarmlandPloughed_Fallow_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/RAW/",
                             outfilename = "FarmlandPloughed_Fallow_cell.tif",
                             layername = "egv_250",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(p2i_rez)
rm(i2e_rez)
rm(dati)
unlink("./RasterGrids_10m/2024/FarmlandPloughed_Fallow_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandPloughed_Fallow_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.251 FarmlandPloughed\_Fallow\_r500

**filename:** FarmlandPloughed\_Fallow\_r500.tif

**layername:** evg\_251

**English name:** Fractional cover of Fallow Land within the 0.5 km landscape

**Latvian name:** Papuvju platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandPloughed_Fallow_cell.tif"),
  layer_prefixes = c("FarmlandPloughed_Fallow"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandPloughed_Fallow_r500.tif evg_251
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandPloughed_Fallow_r500.tif")
names(slanis)="evg_251"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandPloughed_Fallow_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandPloughed_Fallow_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.252 FarmlandPloughed\_Fallow\_r1250

**filename:** FarmlandPloughed\_Fallow\_r1250.tif

**layername:** evg\_252

**English name:** Fractional cover of Fallow Land within the 1.25 km landscape

**Latvian name:** Papuvju platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandPloughed_Fallow_cell.tif"),
  layer_prefixes = c("FarmlandPloughed_Fallow"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandPloughed_Fallow_r1250.tif evg_252
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandPloughed_Fallow_r1250.tif")
names(slanis)="evg_252"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandPloughed_Fallow_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandPloughed_Fallow_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.253 FarmlandPloughed\_Fallow\_r3000

**filename:** FarmlandPloughed\_Fallow\_r3000.tif

**layername:** evg\_253

**English name:** Fractional cover of Fallow Land within the 3 km landscape

**Latvian name:** Papuvju platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandPloughed_Fallow_cell.tif"),
  layer_prefixes = c("FarmlandPloughed_Fallow"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandPloughed_Fallow_r3000.tif evg_253
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandPloughed_Fallow_r3000.tif")
names(slanis)="evg_253"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandPloughed_Fallow_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandPloughed_Fallow_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.254 FarmlandPloughed\_Fallow\_r10000

**filename:** FarmlandPloughed\_Fallow\_r10000.tif

**layername:** egv\_254

**English name:** Fractional cover of Fallow Land within the 10 km landscape

**Latvian name:** Papuvju platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandPloughed_Fallow_cell.tif"),
  layer_prefixes = c("FarmlandPloughed_Fallow"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandPloughed_Fallow_r10000.tif    egv_254
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandPloughed_Fallow_r10000.tif")
names(slanis)="egv_254"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandPloughed_Fallow_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandPloughed_Fallow_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.255 FarmlandSubsidies\_BiologicalSubsidies\_cell

**filename:** FarmlandSubsidies\_BiologicalSubsidies\_cell.tif

**layername:** evg\_255

**English name:** Fractional cover of Farmland receiving Subsidies for Biological Agriculture within the analysis cell (1 ha)

**Latvian name:** Bioloģiskās lauksaimniecības atbalstam pieteikto lauksaimniecības platību īpatsvars analīzē šūnā (1 ha)

**Procedure:** First, agricultural parcels declared as receiving subsidies for biological agriculture are selected from the [Rural Support Service's information on declared fields](#). Geometries are then rasterised to input resolution, ensuring value 1 at the polygon locations and value 0 elsewhere. Rasterisation is performed using the workflow `egvtools::polygon2input()`. Once rasterised, the layer is aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# FarmlandSubsidies_BiologicalSubsidies_cell.tif      evg_255 ----
dati=lad %>%
  filter(str_detect(AID_FORMS,"BLA"))
table(dati$AID_FORMS,useNA="always")

p2i_rez=egvtools::polygon2input(vector_data = dati,
                                template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                out_path = "./RasterGrids_10m/2024/",
                                file_name = "FarmlandSubsidies_BiologicalSubsidies_input.tif",
                                value_field = "yes",
                                prepare=FALSE,
                                background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                plot_result = TRUE)
p2i_rez
```

```

i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "FarmlandSubsidies_BiologicalSubsidies_input.tif"),
                            egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                            summary_function = "average",
                            missing_job = "FillOutput",
                            outlocation = "./RasterGrids_100m/2024/Raw/",
                            outfilename = "FarmlandSubsidies_BiologicalSubsidies_cell.tif",
                            layername = "egv_255",
                            idw_weight = 2,
                            plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(p2i_rez)
rm(i2e_rez)
rm(dat1)
unlink("./RasterGrids_10m/2024/FarmlandSubsidies_BiologicalSubsidies_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandSubsidies_BiologicalSubsidies_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.256 FarmlandSubsidies\_BiologicalSubsidies\_r500

**filename:** FarmlandSubsidies\_BiologicalSubsidies\_r500.tif

**layername:** egv\_256

**English name:** Fractional cover of Farmland receiving Subsidies for Biological Agriculture within the 0.5 km landscape

**Latvian name:** Bioloģiskās lauksaimniecības atbalstam pieteikto lauksaimniecības platību īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",

```

```

tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers =
  c("./RasterGrids_100m/2024/RAW/FarmlandSubsidies_BiologicalSubsidies_cell.tif"),
layer_prefixes = c("FarmlandSubsidies_BiologicalSubsidies"),
output_dir   = "./RasterGrids_100m/2024/RAW/",
n_workers    = 6,
radii        = c("r500"),
radius_mode  = "sparse",
extract_fun  = "mean",
fill_missing = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3)

# FarmlandSubsidies_BiologicalSubsidies_r500.tif    egv_256
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandSubsidies_BiologicalSubsidies_r500.tif")
names(slanis)="egv_256"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandSubsidies_BiologicalSubsidies_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandSubsidies_BiologicalSubsidies_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.257 FarmlandSubsidies\_BiologicalSubsidies\_r1250

**filename:** FarmlandSubsidies\_BiologicalSubsidies\_r1250.tif

**layername:** egv\_257

**English name:** Fractional cover of Farmland receiving Subsidies for Biological Agriculture within the 1.25 km landscape

**Latvian name:** Bioloģiskās lauksaimniecības atbalstam pieteikto lauksaimniecības platību īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

```

```

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    ↳ c("./RasterGrids_100m/2024/RAW/FarmlandSubsidies_BiologicalSubsidies_cell.tif"),
  layer_prefixes = c("FarmlandSubsidies_BiologicalSubsidies"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandSubsidies_BiologicalSubsidies_r1250.tif  egv_257
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandSubsidies_BiologicalSubsidies_r1250.tif")
names(slanis)="egv_257"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandSubsidies_BiologicalSubsidies_r1250.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandSubsidies_BiologicalSubsidies_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.258 FarmlandSubsidies\_BiologicalSubsidies\_r3000

**filename:** FarmlandSubsidies\_BiologicalSubsidies\_r3000.tif

**layername:** egv\_258

**English name:** Fractional cover of Farmland receiving Subsidies for Biological Agriculture within the 3 km landscape

**Latvian name:** Bioloģiskās lauksaimniecības atbalstam pieteikto lauksaimniecības platību īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   =
    ↳ c("./RasterGrids_100m/2024/Raw/FarmlandSubsidies_BiologicalSubsidies_cell.tif"),
  layer_prefixes = c("FarmlandSubsidies_BiologicalSubsidies"),
  output_dir     = "./RasterGrids_100m/2024/Raw/",
  n_workers      = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandSubsidies_BiologicalSubsidies_r3000.tif  egv_258
slanis=rast("./RasterGrids_100m/2024/Raw/FarmlandSubsidies_BiologicalSubsidies_r3000.tif")
names(slanis)="egv_258"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/FarmlandSubsidies_BiologicalSubsidies_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandSubsidies_BiologicalSubsidies_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.259 FarmlandSubsidies\_BiologicalSubsidies\_r10000

**filename:** FarmlandSubsidies\_BiologicalSubsidies\_r10000.tif

**layername:** egv\_259

**English name:** Fractional cover of Farmland receiving Subsidies for Biological Agriculture within the 10 km landscape

**Latvian name:** Bioloģiskās lauksaimniecības atbalstam pieteikto lauksaimniecības platību īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   =
    ↳ c("./RasterGrids_100m/2024/RAW/FarmlandSubsidies_BiologicalSubsidies_cell.tif"),
  layer_prefixes = c("FarmlandSubsidies_BiologicalSubsidies"),
  output_dir     = "./RasterGrids_100m/2024/RAW/",
  n_workers      = 6,
  radii          = c("r10000"),
  radius_mode    = "sparse",
  extract_fun    = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandSubsidies_BiologicalSubsidies_r10000.tif  egv_259
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandSubsidies_BiologicalSubsidies_r10000.tif")
names(slanis)="egv_259"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandSubsidies_BiologicalSubsidies_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandSubsidies_BiologicalSubsidies_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.260 FarmlandTrees\_PermanentCrops\_cell

**filename:** FarmlandTrees\_PermanentCrops\_cell.tif

**layername:** egv\_260

**English name:** Fractional cover of Permanent Crops within the analysis cell (1 ha)

**Latvian name:** Ilggadīgo kultūraugu platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, agricultural parcels declared as permanent crops are selected from the [Rural Support Service's information on declared fields](#). Geometries are then rasterised to input resolution, ensuring value 1 at the polygon locations and value 0 elsewhere. Rasterisation is performed using the workflow `egvtools::polygon2input()`. Once rasterised, the layer is aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# FarmlandTrees_PermanentCrops_cell.tif egv_260 ----
dati=lad %>%
  filter(SDM_grupa_sakums == "augludārzi")
table(dati$SDM_grupa_sakums,useNA="always")
dati=dati %>%
  dplyr::select(yes)

topo=sfarrow::st_read_parquet("./Geodata/2024/TopographicMap/LandusA_COMB.parquet")
dati_topo=topo %>%
  filter(FNAME %in% c("poligons_Augludarzs","poligons_Augludarzs",
  "poligons_Ogulājs","poligons_Ogulajs")) %>%
  mutate(yes=1) %>%
  dplyr::select(yes)
abidati=rbind(dati,dati_topo)

p2i_rez=egvtools::polygon2input(vector_data = abidati,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "FarmlandTrees_PermanentCrops_input.tif",
  value_field = "yes",
  prepare=FALSE,
  background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
```

```

            plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "FarmlandTrees_PermanentCrops_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/RAW/",
                             outfilename = "FarmlandTrees_PermanentCrops_cell.tif",
                             layername = "egv_260",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(p2i_rez)
rm(i2e_rez)
rm(dat)
rm(topo)
rm(dat_topo)
rm(abidat)
unlink("./RasterGrids_10m/2024/FarmlandTrees_PermanentCrops_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandTrees_PermanentCrops_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.261 FarmlandTrees\_PermanentCrops\_r500

**filename:** FarmlandTrees\_PermanentCrops\_r500.tif

**layername:** egv\_261

**English name:** Fractional cover of Permanent Crops within the 0.5 km landscape

**Latvian name:** Ilggadīgo kultūraugu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function()

```

```

kvadrati_path = "./Templates/TemplateGrids/tiles/",
radii_path    = "./Templates/TemplateGridPoints/tiles/",
tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandTrees_PermanentCrops_cell.tif"),
layer_prefixes = c("FarmlandTrees_PermanentCrops"),
output_dir   = "./RasterGrids_100m/2024/RAW/",
n_workers    = 6,
radii        = c("r500"),
radius_mode  = "sparse",
extract_fun  = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3)

# FarmlandTrees_PermanentCrops_r500.tif egv_261
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandTrees_PermanentCrops_r500.tif")
names(slanis)="egv_261"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandTrees_PermanentCrops_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandTrees_PermanentCrops_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.262 FarmlandTrees\_PermanentCrops\_r1250

**filename:** FarmlandTrees\_PermanentCrops\_r1250.tif

**layername:** egv\_262

**English name:** Fractional cover of Permanent Crops within the 1.25 km landscape

**Latvian name:** Ilggadīgo kultūraugu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

```

```

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandTrees_PermanentCrops_cell.tif"),
  layer_prefixes = c("FarmlandTrees_PermanentCrops"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandTrees_PermanentCrops_r1250.tif    egv_262
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandTrees_PermanentCrops_r1250.tif")
names(slanis)="egv_262"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandTrees_PermanentCrops_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandTrees_PermanentCrops_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.263 FarmlandTrees\_PermanentCrops\_r3000

**filename:** FarmlandTrees\_PermanentCrops\_r3000.tif

**layername:** egv\_263

**English name:** Fractional cover of Permanent Crops within the 3 km landscape

**Latvian name:** Ilggadīgo kultūraugu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

```

```

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandTrees_PermanentCrops_cell.tif"),
  layer_prefixes = c("FarmlandTrees_PermanentCrops"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandTrees_PermanentCrops_r3000.tif      egv_263
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandTrees_PermanentCrops_r3000.tif")
names(slanis)="egv_263"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandTrees_PermanentCrops_r3000.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandTrees_PermanentCrops_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.264 FarmlandTrees\_PermanentCrops\_r10000

**filename:** FarmlandTrees\_PermanentCrops\_r10000.tif

**layername:** egv\_264

**English name:** Fractional cover of Permanent Crops within the 10 km landscape

**Latvian name:** Ilggadīgo kultūraugu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs -----
```

```

if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/FarmlandTrees_PermanentCrops_cell.tif"),
  layer_prefixes = c("FarmlandTrees_PermanentCrops"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# FarmlandTrees_PermanentCrops_r10000.tif  egv_264
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandTrees_PermanentCrops_r10000.tif")
names(slanis)="egv_264"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandTrees_PermanentCrops_r10000.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandTrees_PermanentCrops_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.265 FarmlandTrees\_ShortRotationCoppice\_cell

**filename:** FarmlandTrees\_ShortRotationCoppice\_cell.tif

**layername:** egv\_265

**English name:** Fractional cover of Short-rotation Coppice and Other Woody Energy Crops within the analysis cell (1 ha)

**Latvian name:** Īscirtmeta atvasāju un enerģijai audzētu kokaugu platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, agricultural parcels declared as short rotation coppice are selected from the [Rural Support Service's information on declared fields](#). Geometries are then rasterised to input resolution, ensuring value 1 at the polygon locations and value 0 elsewhere. Rasterisation is performed using the workflow `egvtools::polygon2input()`. Once rasterised, the layer is aggregated to EGV resolution using the workflow

`egvtools::input2egv()`, which calculates the arithmetic mean and thus results in a cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# FarmlandTrees_ShortRotationCoppice_cell.tif    egv_265 ----
dati=lad %>%
  filter(SDM_grupa_sakums == "krūmveida ilggadīgie stādījumi")
table(dati$SDM_grupa_sakums,useNA="always")

p2i_rez=egvtools::polygon2input(vector_data = dati,
                                template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                out_path = "./RasterGrids_10m/2024/",
                                file_name = "FarmlandTrees_ShortRotationCoppice_input.tif",
                                value_field = "yes",
                                prepare=FALSE,
                                background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "FarmlandTrees_ShortRotationCoppice_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/Raw/",
                             outfilename = "FarmlandTrees_ShortRotationCoppice_cell.tif",
                             layername = "egv_265",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(p2i_rez)
rm(i2e_rez)
rm(dati)
unlink("./RasterGrids_10m/2024/FarmlandTrees_ShortRotationCoppice_input.tif")
```

```

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandTrees_ShortRotationCoppice_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovidze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovidze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.266 FarmlandTrees\_ShortRotationCoppice\_r500

**filename:** FarmlandTrees\_ShortRotationCoppice\_r500.tif

**layername:** egv\_266

**English name:** Fractional cover of Short-rotation Coppice and Other Woody Energy Crops within the 0.5 km landscape

**Latvian name:** Īscirtmeta atvasāju un enerģijai audzētu kokaugu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/FarmlandTrees_ShortRotationCoppice_cell.tif"),
  layer_prefixes = c("FarmlandTrees_ShortRotationCoppice"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# FarmlandTrees_ShortRotationCoppice_r500.tif    egv_266

```

```

slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandTrees_ShortRotationCoppice_r500.tif")
names(slanis)="egv_266"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandTrees_ShortRotationCoppice_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandTrees_ShortRotationCoppice_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.267 FarmlandTrees\_ShortRotationCoppice\_r1250

**filename:** FarmlandTrees\_ShortRotationCoppice\_r1250.tif

**layername:** egv\_267

**English name:** Fractional cover of Short-rotation Coppice and Other Woody Energy Crops within the 1.25 km landscape

**Latvian name:** Īscirtmeta atvasāju un enerģijai audzētu kokaugu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/FarmlandTrees_ShortRotationCoppice_cell.tif"),
  layer_prefixes = c("FarmlandTrees_ShortRotationCoppice"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",

```

```

fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# FarmlandTrees_ShortRotationCoppice_r1250.tif egv_267
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandTrees_ShortRotationCoppice_r1250.tif")
names(slanis)="egv_267"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/FarmlandTrees_ShortRotationCoppice_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandTrees_ShortRotationCoppice_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.268 FarmlandTrees\_ShortRotationCoppice\_r3000

**filename:** FarmlandTrees\_ShortRotationCoppice\_r3000.tif

**layername:** egv\_268

**English name:** Fractional cover of Short-rotation Coppice and Other Woody Energy Crops within the 3 km landscape

**Latvian name:** Īscirtmeta atvasāju un enerģijai audzētu kokaugu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadрати_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/FarmlandTrees_ShortRotationCoppice_cell.tif"),

```

```

layer_prefixes = c("FarmlandTrees_ShortRotationCoppice"),
output_dir    = "./RasterGrids_100m/2024/RAW/",
n_workers     = 6,
radii         = c("r3000"),
radius_mode   = "sparse",
extract_fun   = "mean",
fill_missing   = TRUE,
IDW_weight    = 2,
future_max_size = 40 * 1024^3)

# FarmlandTrees_ShortRotationCoppice_r3000.tif  egv_268
slanis=rast("./RasterGrids_100m/2024/RAW/FarmlandTrees_ShortRotationCoppice_r3000.tif")
names(slanis)="egv_268"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/FarmlandTrees_ShortRotationCoppice_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
nosaukums="FarmlandTrees_ShortRotationCoppice_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.269 FarmlandTrees\_ShortRotationCoppice\_r10000

**filename:** FarmlandTrees\_ShortRotationCoppice\_r10000.tif

**layername:** egv\_269

**English name:** Fractional cover of Short-rotation Coppice and Other Woody Energy Crops within the 10 km landscape

**Latvian name:** Īscirtmeta atvasāju un enerģijai audzētu kokaugu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",

```

```

radii_path = "./Templates/TemplateGridPoints/tiles/",
tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers =
  ↳ c("./RasterGrids_100m/2024/Raw/FarmlandTrees_ShortRotationCoppice_cell.tif"),
layer_prefixes = c("FarmlandTrees_ShortRotationCoppice"),
output_dir = "./RasterGrids_100m/2024/Raw/",
n_workers = 6,
radii = c("r10000"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# FarmlandTrees_ShortRotationCoppice_r10000.tif egv_269
slanis=rast("./RasterGrids_100m/2024/Raw/FarmlandTrees_ShortRotationCoppice_r10000.tif")
names(slanis)="egv_269"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/Raw/FarmlandTrees_ShortRotationCoppice_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="FarmlandTrees_ShortRotationCoppice_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.270 ForestsAge\_ClearcutsLowStands\_cell

**filename:** ForestsAge\_ClearcutsLowStands\_cell.tif

**layername:** egv\_270

**English name:** Fractional cover of Clearcuts and Forest Stands lower than 5 m within the analysis cell (1 ha)

**Latvian name:** Izcirtumu un mežaudžu līdz 5 m augstumam platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** To prepare this EGV, stands in land category 10 with a height of less than 5 m are selected from the [State Forest Service's State Forest Registry](#) and rasterised. After rasterisation, this layer is covered by a clearcut mask. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}

```

```

if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# ForestsAge_ClearcutsLowStands_cell.tif      egv_270 ----
zemas_audzes=mvr %>%
  filter(zkat=="10") %>%
  filter(h10<5) %>%
  dplyr::select(yes)
r_zemasaudzes=fasterize(zemas_audzes,rastrs10,field="yes")
t_zemasaudzes=rast(r_zemasaudzes)
plot(t_zemasaudzes)

cleacuts_low=cover(t_zemasaudzes,clearcut_mask)
plot(cleacuts_low)

i2e_rez=egvtools::input2egv(input=cleacuts_low,
                           egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                           summary_function = "average",
                           missing_job = "FillOutput",
                           outlocation = "./RasterGrids_100m/2024/RAW/",
                           outfilename = "ForestsAge_ClearcutsLowStands_cell.tif",
                           layername = "egv_270",
                           idw_weight = 2,
                           plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(i2e_rez)
rm(zemas_audzes)
rm(r_zemasaudzes)
rm(t_zemasaudzes)
rm(cleacuts_low)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_ClearcutsLowStands_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,

```

```
overwrite=TRUE)
```

## 6.271 ForestsAge\_ClearcutsLowStands\_r500

**filename:** ForestsAge\_ClearcutsLowStands\_r500.tif

**layername:** egv\_271

**English name:** Fractional cover of Clearcuts and Forest Stands lower than 5 m within the 0.5 km landscape

**Latvian name:** Izcirtumu un mežaudžu līdz 5 m augstumam platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsAge_ClearcutsLowStands_cell.tif"),
  layer_prefixes = c("ForestsAge_ClearcutsLowStands"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsAge_ClearcutsLowStands_r500.tif    egv_271
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsAge_ClearcutsLowStands_r500.tif")
names(slanis)="egv_271"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsAge_ClearcutsLowStands_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_ClearcutsLowStands_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
```

```

merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.272 ForestsAge\_ClearcutsLowStands\_r1250

**filename:** ForestsAge\_ClearcutsLowStands\_r1250.tif

**layername:** egv\_272

**English name:** Fractional cover of Clearcuts and Forest Stands lower than 5 m within the 1.25 km landscape

**Latvian name:** Izcirtumu un mežaudžu līdz 5 m augstumam platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsAge_ClearcutsLowStands_cell.tif"),
  layer_prefixes = c("ForestsAge_ClearcutsLowStands"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsAge_ClearcutsLowStands_r1250.tif  egv_272
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsAge_ClearcutsLowStands_r1250.tif")
names(slanis)="egv_272"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsAge_ClearcutsLowStands_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_ClearcutsLowStands_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)

```

```

videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.273 ForestsAge\_ClearcutsLowStands\_r3000

**filename:** ForestsAge\_ClearcutsLowStands\_r3000.tif

**layername:** egv\_273

**English name:** Fractional cover of Clearcuts and Forest Stands lower than 5 m within the 3 km landscape

**Latvian name:** Izcirtumu un mežaudžu līdz 5 m augstumam platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsAge_ClearcutsLowStands_cell.tif"),
  layer_prefixes = c("ForestsAge_ClearcutsLowStands"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsAge_ClearcutsLowStands_r3000.tif  egv_273
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsAge_ClearcutsLowStands_r3000.tif")
names(slanis)="egv_273"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsAge_ClearcutsLowStands_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_ClearcutsLowStands_r3000.tif"

```

```

ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.274 ForestsAge\_ClearcutsLowStands\_r10000

**filename:** ForestsAge\_ClearcutsLowStands\_r10000.tif

**layername:** egv\_274

**English name:** Fractional cover of Clearcuts and Forest Stands lower than 5 m within the 10 km landscape

**Latvian name:** Izcirtumu un mežaudžu līdz 5 m augstumam platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsAge_ClearcutsLowStands_cell.tif"),
  layer_prefixes = c("ForestsAge_ClearcutsLowStands"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsAge_ClearcutsLowStands_r10000.tif egv_274
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsAge_ClearcutsLowStands_r10000.tif")
names(slanis)="egv_274"
slanis2=project(slanis,template100)
writeRaster(slanis2,
    "./RasterGrids_100m/2024/RAW/ForestsAge_ClearcutsLowStands_r10000.tif",
    overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}

```

```

if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_ClearcutsLowStands_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.275 ForestsAge\_Middle\_cell

**filename:** ForestsAge\_Middle\_cell.tif

**layername:** egv\_275

**English name:** Fractional cover of Middle-Aged Forest Stands within the analysis cell (1 ha)

**Latvian name:** Vidēja vecuma un briestaudžu platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

To prepare this EGV, stands in land category 10 and age groups two and three are selected from [State Forest Service's State Forest Registry](#) and rasterised. Rasterisation is performed using the workflow `egvtools::polygon2input()` (presence = 1, absence = 0), restricting presence locations only outside the clearcut mask. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

```

```

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsAge_Middle_cell.tif      egv_275 ----
videjas_audzes=mvr %>%
  filter(zkat=="10") %>%
#filter(h10>=5) %>%
  filter(vgr %in% c("2","3")) %>%
  dplyr::select(yes)

p2i_rez=egvtools::polygon2input(vector_data = videjas_audzes,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsAge_Middle_input.tif",
  value_field = "yes",
  restrict_to = clearcut_mask,
  restrict_values = 0,
  prepare=FALSE,
  background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
  plot_result = TRUE)

p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
  "ForestsAge_Middle_input.tif"),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "ForestsAge_Middle_cell.tif",
  layername = "egv_275",
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(videjas_audzes)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsAge_Middle_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_Middle_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)

```

```

slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.276 ForestsAge\_Middle\_r500

**filename:** ForestsAge\_Middle\_r500.tif

**layername:** egv\_276

**English name:** Fractional cover of Middle-Aged Forest Stands within the 0.5 km landscape

**Latvian name:** Vidēja vecuma un briestaudžu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path   = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers    = c("./RasterGrids_100m/2024/RAW/ForestsAge_Middle_cell.tif"),
  layer_prefixes = c("ForestsAge_Middle"),
  output_dir     = "./RasterGrids_100m/2024/RAW/",
  n_workers      = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsAge_Middle_r500.tif    egv_276
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsAge_Middle_r500.tif")
names(slanis)="egv_276"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsAge_Middle_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

nosaukums="ForestsAge_Middle_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.277 ForestsAge\_Middle\_r1250

**filename:** ForestsAge\_Middle\_r1250.tif

**layername:** egv\_277

**English name:** Fractional cover of Middle-Aged Forest Stands within the 1.25 km landscape

**Latvian name:** Vidēja vecuma un briestaudžu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsAge_Middle_cell.tif"),
  layer_prefixes = c("ForestsAge_Middle"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsAge_Middle_r1250.tif  egv_277
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsAge_Middle_r1250.tif")
names(slanis)="egv_277"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsAge_Middle_r1250.tif",
  overwrite=TRUE)

# standardisation ----

```

```

if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_Middle_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.278 ForestsAge\_Middle\_r3000

**filename:** ForestsAge\_Middle\_r3000.tif

**layername:** egv\_278

**English name:** Fractional cover of Middle-Aged Forest Stands within the 3 km landscape

**Latvian name:** Vidēja vecuma un briestaudžu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadri_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/ForestsAge_Middle_cell.tif"),
  layer_prefixes = c("ForestsAge_Middle"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsAge_Middle_r3000.tif  egv_278
slanis=rast("./RasterGrids_100m/2024/Raw/ForestsAge_Middle_r3000.tif")
names(slanis)="egv_278"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/Raw/ForestsAge_Middle_r3000.tif",
            overwrite=TRUE)

```

```

overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_Middle_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.279 ForestsAge\_Middle\_r10000

**filename:** ForestsAge\_Middle\_r10000.tif

**layername:** egv\_279

**English name:** Fractional cover of Middle-Aged Forest Stands within the 10 km landscape

**Latvian name:** Vidēja vecuma un briestaudžu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/ForestsAge_Middle_cell.tif"),
  layer_prefixes = c("ForestsAge_Middle"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsAge_Middle_r10000.tif egv_279
slanis=rast("./RasterGrids_100m/2024/Raw/ForestsAge_Middle_r10000.tif")
names(slanis)="egv_279"

```

```

slanis2=project(slanis,template100)
writeRaster(slanis2,
    "./RasterGrids_100m/2024/RAW/ForestsAge_Middle_r10000.tif",
    overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_Middle_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.280 ForestsAge\_Old\_cell

**filename:** ForestsAge\_Old\_cell.tif

**layername:** egv\_280

**English name:** Fractional cover of Old (over rotation age) Forest Stands within the analysis cell (1 ha)

**Latvian name:** Vecu (kopš cirtmeta) mežaudžu platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

To prepare this EGV, stands in land category 10 and age groups four and five are selected from [State Forest Service's State Forest Registry](#) and rasterised. Rasterisation is performed using the workflow `egvtools::polygon2input()` (presence = 1, absence = 0), restricting presence locations only outside the clearcut mask. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

```

```

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,raster10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsAge_Old_cell.tif    egv_280 ----
vecas=mvr %>%
  filter(zkat=="10") %>%
  #filter(h10>=5) %>%
  filter(vgr %in% c("4","5")) %>%
  dplyr::select(yes)

p2i_rez=egvtools::polygon2input(vector_data = vecas,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsAge_Old_input.tif",
  value_field = "yes",
  restrict_to = clearcut_mask,
  restrict_values = 0,
  prepare=FALSE,
  background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
  plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
  "ForestsAge_Old_input.tif"),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "ForestsAge_Old_cell.tif",
  layername = "egv_280",
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(vecas)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsAge_Old_input.tif")

```

```

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_Old_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.281 ForestsAge\_Old\_r500

**filename:** ForestsAge\_Old\_r500.tif

**layername:** egv\_281

**English name:** Fractional cover of Old (over rotation age) Forest Stands within the 0.5 km landscape

**Latvian name:** Vecu (kopš cirtmeta) mežaudžu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsAge_Old_cell.tif"),
  layer_prefixes = c("ForestsAge_Old"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsAge_Old_r500.tif  egv_281
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsAge_Old_r500.tif")
names(slanis)="egv_281"
slanis2=project(slanis,template100)
writeRaster(slanis2,

```

```

"./RasterGrids_100m/2024/RAW/ForestsAge_Old_r500.tif",
overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_Old_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.282 ForestsAge\_Old\_r1250

**filename:** ForestsAge\_Old\_r1250.tif

**layername:** egv\_282

**English name:** Fractional cover of Old (over rotation age) Forest Stands within the 1.25 km landscape

**Latvian name:** Vecu (kopš cirtmeta) mežaudžu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path  = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   = c("./RasterGrids_100m/2024/RAW/ForestsAge_Old_cell.tif"),
  layer_prefixes = c("ForestsAge_Old"),
  output_dir     = "./RasterGrids_100m/2024/RAW/",
  n_workers      = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsAge_Old_r1250.tif egv_282
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsAge_Old_r1250.tif")

```

```

names(slanis)="egv_282"
slanis2=project(slanis,template100)
writeRaster(slanis2,
    "./RasterGrids_100m/2024/RAW/ForestsAge_Old_r1250.tif",
    overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_Old_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.283 ForestsAge\_Old\_r3000

**filename:** ForestsAge\_Old\_r3000.tif

**layername:** egv\_283

**English name:** Fractional cover of Old (over rotation age) Forest Stands within the 3 km landscape

**Latvian name:** Vecu (kopš cirtmeta) mežaudžu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadрати_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsAge_Old_cell.tif"),
  layer_prefixes = c("ForestsAge_Old"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

```

```

# ForestsAge_Old_r3000.tif  evg_283
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsAge_Old_r3000.tif")
names(slanis)="evg_283"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsAge_Old_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_Old_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.284 ForestsAge\_Old\_r10000

**filename:** ForestsAge\_Old\_r10000.tif

**layername:** evg\_284

**English name:** Fractional cover of Old (over rotation age) Forest Stands within the 10 km landscape

**Latvian name:** Vecu (kopš cirtmeta) mežaudžu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsAge_Old_cell.tif"),
  layer_prefixes = c("ForestsAge_Old"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing  = TRUE,

```

```

IDW_weight = 2,
future_max_size = 40 * 1024^3

# ForestsAge_Old_r10000.tif egv_284
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsAge_Old_r10000.tif")
names(slanis)="egv_284"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsAge_Old_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_Old_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.285 ForestsAge\_YoungTallStandsShrubs\_cell

**filename:** ForestsAge\_YoungTallStandsShrubs\_cell.tif

**layername:** egv\_285

**English name:** Fractional cover of Shrubs, Young Forest Stands (at least 5 m tall) within the analysis cell (1 ha)

**Latvian name:** Krūmāju un jaunaudžu (no 5 m augstuma) platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

To prepare this EGV, stands in land category 10 and age group 1 with height above 5 m are selected from the [State Forest Service's State Forest Registry](#) and rasterised (presence = 1, NA otherwise). This layer is then combined with the category 620 from the [Landscape classification](#) (presence = 1, 0 otherwise). Values in pixels matching the clearcut mask are set to 0. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

```

```

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsAge_YoungTallStandsShrubs_cell.tif egv_285 ----
jaunasaugstas=mvr %>%
  filter(zkat=="10") %>%
  filter(h10>5) %>%
  filter(vgr %in% c("1")) %>%
  dplyr::select(yes)
r_jaunasaugstas=fasterize(jaunasaugstas,rastrs10,field="yes")
t_jaunasaugstas=rast(r_jaunasaugstas)
plot(t_jaunasaugstas)

shrubs=ifel(simple_landscape==620,1,0)

younshrubbs=cover(t_jaunasaugstas,shrubs)
plot(younshrubbs)

younshrubbs2=ifel(younshrubbs==1&clearcut_mask==0,1,0)
plot(younshrubbs2)

i2e_rez=egvtools::input2egv(input=younshrubbs2,
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "ForestsAge_YoungTallStandsShrubs_cell.tif",
  layername = "egv_285",

```

```

        idw_weight = 2,
        plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(jaunasaugstas)
rm(r_jaunasaugstas)
rm(t_jaunasaugstas)
rm(shrubs)
rm(younshrubs)
rm(younshrubs2)
rm(i2e_rez)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_YoungTallStandsShrubs_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.286 ForestsAge\_YoungTallStandsShrubs\_r500

**filename:** ForestsAge\_YoungTallStandsShrubs\_r500.tif

**layername:** egv\_286

**English name:** Fractional cover of Shrubs, Young Forest Stands (at least 5 m tall) within the 0.5 km landscape

**Latvian name:** Krūmāju un jaunaudžu (no 5 m augstuma) platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/ForestsAge_YoungTallStandsShrubs_cell.tif"),
  layer_prefixes = c("ForestsAge_YoungTallStandsShrubs"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,

```

```

radii      = c("r500"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# ForestsAge_YoungTallStandsShrubs_r500.tif egv_286
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsAge_YoungTallStandsShrubs_r500.tif")
names(slanis)="egv_286"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsAge_YoungTallStandsShrubs_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_YoungTallStandsShrubs_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.287 ForestsAge\_YoungTallStandsShrubs\_r1250

**filename:** ForestsAge\_YoungTallStandsShrubs\_r1250.tif

**layername:** egv\_287

**English name:** Fractional cover of Shrubs, Young Forest Stands (at least 5 m tall) within the 1.25 km landscape

**Latvian name:** Krūmāju un jaunaudžu (no 5 m augstuma) platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",

```

```

template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers = c("./RasterGrids_100m/2024/RAW/ForestsAge_YoungTallStandsShrubs_cell.tif"),
layer_prefixes = c("ForestsAge_YoungTallStandsShrubs"),
output_dir = "./RasterGrids_100m/2024/RAW/",
n_workers = 6,
radii = c("r1250"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# ForestsAge_YoungTallStandsShrubs_r1250.tif    egv_287
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsAge_YoungTallStandsShrubs_r1250.tif")
names(slanis)="egv_287"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsAge_YoungTallStandsShrubs_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_YoungTallStandsShrubs_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.288 ForestsAge\_YoungTallStandsShrubs\_r3000

**filename:** ForestsAge\_YoungTallStandsShrubs\_r3000.tif

**layername:** egv\_288

**English name:** Fractional cover of Shrubs, Young Forest Stands (at least 5 m tall) within the 3 km landscape

**Latvian name:** Krūmāju un jaunaudžu (no 5 m augstuma) platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function()

```

```

kvadrati_path = "./Templates/TemplateGrids/tiles/",
radii_path    = "./Templates/TemplateGridPoints/tiles/",
tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsAge_YoungTallStandsShrubs_cell.tif"),
layer_prefixes = c("ForestsAge_YoungTallStandsShrubs"),
output_dir   = "./RasterGrids_100m/2024/RAW/",
n_workers    = 6,
radii        = c("r3000"),
radius_mode  = "sparse",
extract_fun  = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3

# ForestsAge_YoungTallStandsShrubs_r3000.tif      egv_288
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsAge_YoungTallStandsShrubs_r3000.tif")
names(slanis)="egv_288"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsAge_YoungTallStandsShrubs_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_YoungTallStandsShrubs_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.289 ForestsAge\_YoungTallStandsShrubs\_r10000

**filename:** ForestsAge\_YoungTallStandsShrubs\_r10000.tif

**layername:** egv\_289

**English name:** Fractional cover of Shrubs, Young Forest Stands (at least 5 m tall) within the 10 km landscape

**Latvian name:** Krūmāju un jaunaudžu (no 5 m augstuma) platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

```

```

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsAge_YoungTallStandsShrubs_cell.tif"),
  layer_prefixes = c("ForestsAge_YoungTallStandsShrubs"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsAge_YoungTallStandsShrubs_r10000.tif  egv_289
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsAge_YoungTallStandsShrubs_r10000.tif")
names(slanis)="egv_289"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsAge_YoungTallStandsShrubs_r10000.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsAge_YoungTallStandsShrubs_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.290 ForestsQuant\_AgeProp-average\_cell

**filename:** ForestsQuant\_AgeProp-average\_cell.tif

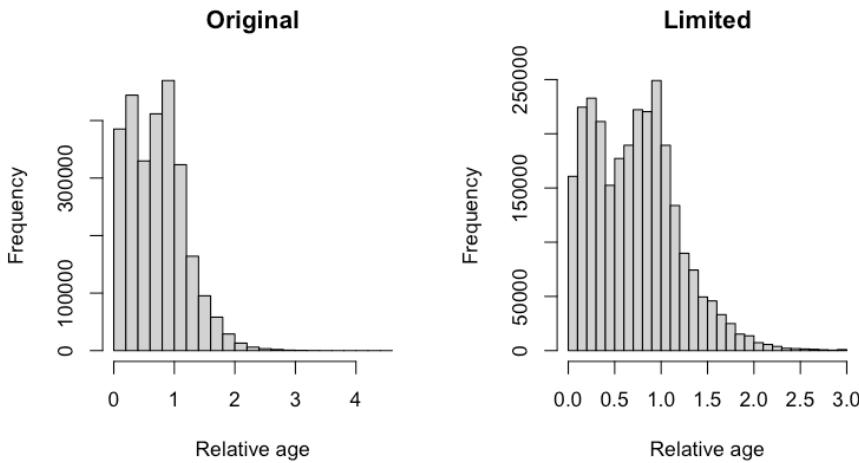
**layername:** egv\_290

**English name:** Average stand age relative to rotation age within the analysis cell (1 ha)

**Latvian name:** Mežaudzēs vecuma attiecība pret cirtmetu, vidējais analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

To prepare this EGV, every forest stand had assigned [legal rotation age](#), based on dominant tree species and bonity class as registered in the [State Forest Service's State Forest Registry](#). We assumed 35 years as the rotation age for grey alder. The registered age of the dominant tree group is then divided by the stand specific legal rotation age. This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised with the workflow `egvtools::polygon2input()`, restricting to pixels outside the clearcut mask. No background values are assigned during rasterisation. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating arithmetic mean. After the aggregation, cells with no forest information are filled with value 0. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
```

```

    overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# Forest law https://likumi.lv/ta/id/2825#p9
ozoli=c("10","61")
priedes_lapegles=c("1","13","14","22")
eolgvk=c("3","15","23","11","64","12","62","16","65","24","63")
berzi=c("4")
melnalksni=c("6")
apses=c("8","19","68")

bonA=c("0","1")
bonB=c("2","3")
bonC=c("4","5","6")
bonAB=c("0","1","2","3")

nogabali=mvr %>%
  mutate(cirtmets=ifelse((s10 %in% ozoli)&(bon %in% bonA),101,
                        ifelse((s10 %in% ozoli),121,NA))) %>%
  mutate(cirtmets=ifelse((s10 %in% priedes_lapegles)&(bon %in% bonAB),101,
                        ifelse((s10 %in% priedes_lapegles),121,cirtmets))) %>%
  mutate(cirtmets=ifelse((s10 %in% eolgvk),81,cirtmets)) %>%
  mutate(cirtmets=ifelse((s10 %in% berzi)&(bon %in% bonAB),71,
                        ifelse((s10 %in% berzi),51,cirtmets))) %>%
  mutate(cirtmets=ifelse((s10 %in% melnalksni),71,cirtmets)) %>%
  mutate(cirtmets=ifelse((s10 %in% apses),41,cirtmets)) %>%
  mutate(cirtmets=ifelse(is.na(cirtmets)&zkat=="10",35,cirtmets)) %>%
  mutate(nogvec=a10/cirtmets) %>%
  mutate(nogvec2=ifelse(nogvec>3,3,nogvec)) %>%
  filter(!is.na(nogvec))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$nogvec,main="Original",xlab="Relative age")
hist(nogabali$nogvec2,main="Limited",xlab="Relative age")
par(mfrow=c(1,1))
options(scipen=0)
# 700*400

p2i_rez=polygon2input(vector_data=nogabali,
                       template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                       out_path = "./RasterGrids_10m/2024/",
                       file_name = "ForestQuant_AgeProp.tif",
                       value_field = "nogvec2",
                       fun="max",
                       prepare=FALSE,
                       restrict_to = clearcut_mask,
                       restrict_values = 0,
                       plot_result=TRUE)

p2i_rez
i2e_rez=input2egv(input="./RasterGrids_10m/2024/ForestQuant_AgeProp.tif",
                    egv_template = "./Templates/TemplateRasters/LV100m_10km.tif",
                    summary_function = "average",
                    missing_job = "CoverOutput",
                    output_bg = "./Templates/TemplateRasters/nulls_LV100m_10km.tif",
                    outlocation = "./RasterGrids_100m/2024/Raw/",
                    outfilename = "ForestsQuant_AgeProp-average_cell.tif",

```

```

    layername = "egv_290",
    plot_final=TRUE)
i2e_rez
rm(nogabali)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestQuant_AgeProp.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsQuant_AgeProp-average_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.291 ForestsQuant\_DominantDiameter-max\_cell

**filename:** ForestsQuant\_DominantDiameter-max\_cell.tif

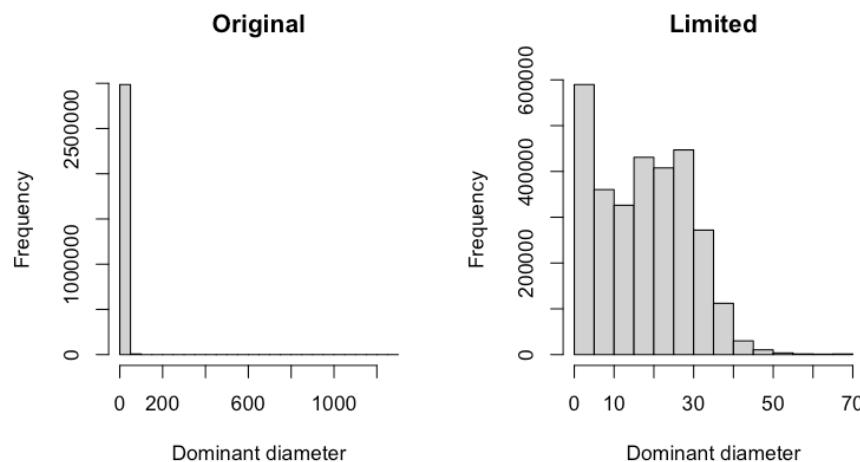
**layername:** egv\_291

**English name:** Dominant tree trunk diameter, maximum within the analysis cell (1 ha)

**Latvian name:** Koku stumbra diametrs, valdaudzes maksimālais analīzes šūnā (1 ha)

**Procedure:** Most of forests describing EGVs are spatially restricted outside clearcuts and dead stands. Mask for this is created from the [State Forest Service's State Forest Registry](#) land category 12 and 14 combined with [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

This EGV is prepared based on the information of dominant tree species per inventoried forest stand - [State Forest Service's State Forest Registry](#). This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised with the workflow `egvtools::polygon2input()`, restricting to pixels outside the clearcut mask. No background values are assigned during rasterisation. The

resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating maximum value. After the aggregation, cells with no forest information are filled with value 0. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsQuant_DominantDiameter-max_cell.tif      egv_291 ----
nogabali=mvr %>%
  mutate(valddiam=ifelse(d10>70,70,d10)) %>%
  filter(!is.na(valddiam))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$d10,main="Original",xlab="Dominant diameter")
hist(nogabali$valddiam,main="Limited",xlab="Dominant diameter")
par(mfrow=c(1,1))
```

```

options(scipen=0)

p2i_rez=polygon2input(vector_data=nogabali,
                       template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                       out_path = "./RasterGrids_10m/2024/",
                       file_name = "ForestQuant_DominantDiameter.tif",
                       value_field = "valddiam",
                       fun="max",
                       prepare=FALSE,
                       restrict_to = clearcut_mask,
                       restrict_values = 0,
                       plot_result=TRUE,
                       overwrite=TRUE)

p2i_rez
i2e_rez=input2egv(input="./RasterGrids_10m/2024/ForestQuant_DominantDiameter.tif",
                    evg_template = "./Templates/TemplateRasters/LV100m_10km.tif",
                    summary_function = "max",
                    missing_job = "CoverOutput",
                    output_bg = "./Templates/TemplateRasters/nulls_LV100m_10km.tif",
                    outlocation = "./RasterGrids_100m/2024/Raw/",
                    outfilename = "ForestsQuant_DominantDiameter-max_cell.tif",
                    layername = "egv_291",
                    plot_final=TRUE)

i2e_rez
rm(p2i_rez)
rm(nogabali)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestQuant_DominantDiameter.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsQuant_DominantDiameter-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.292 ForestsQuant\_LargestDiameter-max\_cell

**filename:** ForestsQuant\_LargestDiameter-max\_cell.tif

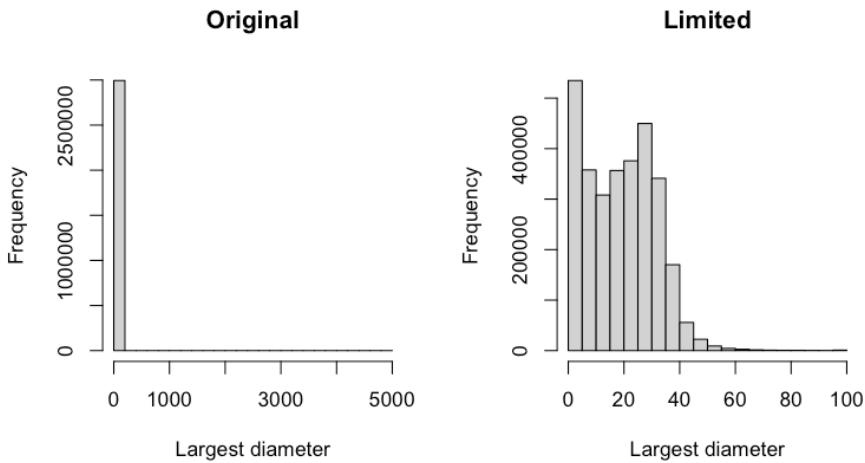
**layername:** egv\_292

**English name:** Largest tree trunk diameter within the analysis cell (1 ha)

**Latvian name:** Lielākais koka stumbra diametrs analīzē šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

This EGV is prepared based on the information of the largest tree diameter per inventoried forest stand - [State Forest Service's State Forest Registry](#). This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised with the workflow `egvtools::polygon2input()`, restricting to pixels outside the clearcut mask. No background values are assigned during rasterisation. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating maximum value. After the aggregation, cells with no forest information are filled with value 0. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
```

```

    overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsQuant_LargestDiameter-max_cell.tif egv_292 ----
nogabali=mvr %>%
  rowwise() %>%
  mutate(maxDiam=max(c(d10,d11,d12,d13,d14,d22,d23,d24),na.rm=TRUE)) %>%
  ungroup() %>%
  mutate(maxDiam2=ifelse(maxDiam>100,100,maxDiam)) %>%
  filter(!is.na(maxDiam2))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$maxDiam,main="Original",xlab="Largest diameter")
hist(nogabali$maxDiam2,main="Limited",xlab="Largest diameter")
par(mfrow=c(1,1))
options(scipen=0)

p2i_rez=polygon2input(vector_data=nogabali,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsQuant_LargestDiameter.tif",
  value_field = "maxDiam2",
  fun="max",
  prepare=FALSE,
  restrict_to = clearcut_mask,
  restrict_values = 0,
  plot_result=TRUE,
  overwrite=TRUE)

p2i_rez
i2e_rez=input2egv(input="./RasterGrids_10m/2024/ForestsQuant_LargestDiameter.tif",
  egv_template = "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "max",
  missing_job = "CoverOutput",
  output_bg = "./Templates/TemplateRasters/nulls_LV100m_10km.tif",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "ForestsQuant_LargestDiameter-max_cell.tif",
  layername = "egv_292",
  plot_final=TRUE)

i2e_rez
rm(p2i_rez)
rm(nogabali)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsQuant_LargestDiameter.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsQuant_LargestDiameter-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,

```

```
overwrite=TRUE
```

## 6.293 ForestsQuant\_TimeSinceDisturbance-average\_cell

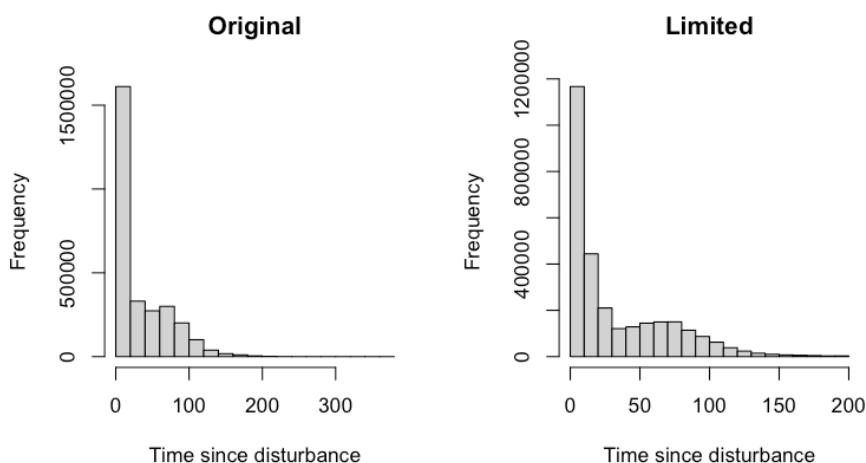
**filename:** ForestsQuant\_TimeSinceDisturbance-average\_cell.tif

**layername:** egv\_293

**English name:** Time since last disturbance affecting tree growing within the analysis cell (1 ha)

**Latvian name:** Laiks kopš pēdējā ar koku augšanu saistītā traucējuma analīzes šūnā (1 ha)

**Procedure:** This EGV is prepared primarily based on the information of the forestry related disturbances as registered per inventoried forest stand - [State Forest Service's State Forest Registry](#). The register, however, includes obvious errors - values later than 2024 and earlier than 1500 that are set to NA. Remaining values are subtracted from 2024. In stands with no disturbance registered, the age of dominant tree group is used to calculate minimum difference (age of time since disturbance) from the year 2024. This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised (presence only). This raster layer is then overlaid with reclassified year of tree cover loss (reclassified to difference from the year 2024) and per pixel minimum value is retained. As not all the forests or tree covered areas are inventoried, classes from the [Landscape classification](#) are used to impute assumption of time since tree growing disturbance - for class 620 we assume five years, whereas for classes 630 and 640 - 50 years and 0 otherwise. Per pixel minimum layer is then overlaid the assumed time since disturbance layer. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating arithmetic mean value. After the aggregation, inverse distance weighted (power = 2) gap filling is applied to avoid possible gaps at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)
```

```

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsQuant_TimeSinceDisturbance-average_cell.tif      egv_293 ----
nogabali=mvr %>%
  mutate(new_PDG=ifelse(p_darbg>2024,NA,
    ifelse(p_darbv %in% c("1","4","5","6","7","10","11"),p_darbg,NA)),
  new_PDG2=ifelse(new_PDG<1500,NA,new_PDG),
  new_PCG=ifelse(p_cirg>2024,NA,p_cirg),
  new_PCG2=ifelse(new_PCG<1500,NA,new_PCG),
  vecumam=ifelse(a10==0,NA,a10),
  new_PCG3=2024-new_PCG2,
  new_PDG3=2024-new_PDG2) %>%
  rowwise() %>%
  mutate(Laikam=min(c(vecumam,new_PDG3,new_PCG3),na.rm=TRUE)) %>%
  ungroup() %>%
  mutate(KopsTraucejuma=ifelse(is.infinite(Laikam),NA,Laikam)) %>%
  mutate(KopsTraucejuma2=ifelse(KopsTraucejuma>200,200,KopsTraucejuma)) %>%
  filter(!is.na(KopsTraucejuma2))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$KopsTraucejuma,main="Original",xlab="Time since disturbance")
hist(nogabali$KopsTraucejuma2,main="Limited",xlab="Time since disturbance")
par(mfrow=c(1,1))
options(scipen=0)

mvr_trauclaiks=fasterize::fasterize(nogabali,rastrs10,field="KopsTraucejuma2",fun = "min")
t_MVRtrauclaiks=rast(mvr_trauclaiks)
plot(t_MVRtrauclaiks)

gfw=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")

```

```

plot(gfw)
gfw2=ifel(gfw>=0,24-gfw,NA)
plot(gfw2)

# No ainavas:
## Mežaudzes un koki = 50
## Krūmāji un parki = 5
## pārējais = 0
aizpildisanai=ifel(simple_landscape==630|simple_landscape==640,50,
                     ifel(simple_landscape==620,5,0))
freq(aizpildisanai)

trauclaiks1=terra::app(c(gfw2,t_MVRtrauclaiks),fun="min",na.rm=TRUE)
plot(trauclaiks1)
trauclaiks2=cover(trauclaiks1,aizpildisanai)
plot(trauclaiks2)

i2e_rez=input2egv(input=trauclaiks2,
                    evg_template = "./Templates/TemplateRasters/LV100m_10km.tif",
                    summary_function = "average",
                    missing_job = "FillOutput",
                    idw_weight = 2,
                    outlocation = "./RasterGrids_100m/2024/RAW/",
                    outfilename = "ForestsQuant_TimeSinceDisturbance-average_cell.tif",
                    layername = "egv_293",
                    plot_final=TRUE)

i2e_rez
rm(nogabali)
rm(mvr_trauclaiks)
rm(t_MVRtrauclaiks)
rm(gfw)
rm(gfw2)
rm(aizpildisanai)
rm(trauclaiks1)
rm(trauclaiks2)
rm(i2e_rez)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsQuant_TimeSinceDisturbance-average_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.294 ForestsQuant\_VolumeAspen-sum\_cell

**filename:** ForestsQuant\_VolumeAspen-sum\_cell.tif

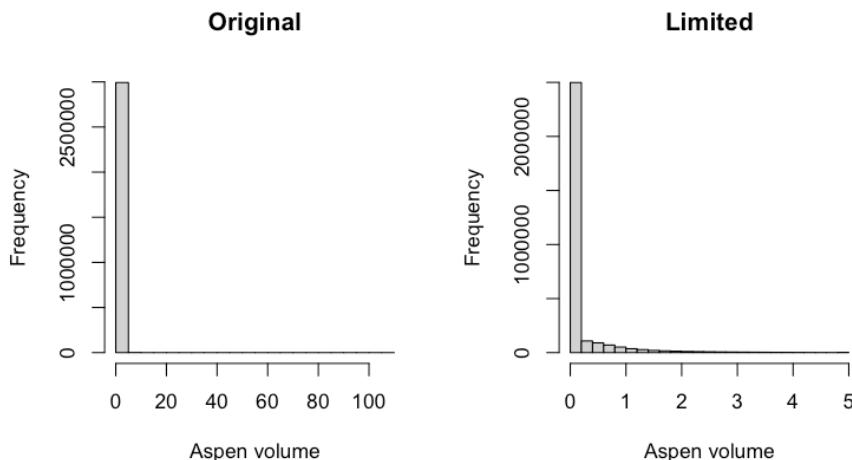
**layername:** egv\_294

**English name:** Timber volume of Aspens, Poplars within the analysis cell (1 ha)

**Latvian name:** Apšu, papeļu krāja analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the State Forest Service's State Forest Registry land category 12 and 14, and The Global Forest Watch pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

This EGV is prepared based on the information of timber volume of aspen (species codes: 8, 19, 68; see tree species codes in [Terminology and acronyms](#)) in the inventoried forest stands - [State Forest Service's State Forest Registry](#). This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised with the workflow `egvtools::polygon2input()`, restricting to pixels outside the clearcut mask. No background values are assigned during rasterisation. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating sum of pixel values. After the aggregation, cells with no forest information are filled with value 0. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
```

```

r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
    filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
    overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsQuant_VolumeAspen-sum_cell.tif evg_294 ----

apses=c("8","19","68")
nogabali=mvr %>%
  mutate(ApsuKraja=ifelse(s10 %in% apses, v10, 0)+ifelse(s11 %in% apses,v11,0)+
    ifelse(s12 %in% apses, v12,0)+ifelse(s13 %in% apses,v13,0)+
    ifelse(s14 %in% apses, v14,0)) %>%
  mutate(ApsuKraja2=ApsuKraja/10000*10*10) %>%
  mutate(ApsuKraja3=ifelse(ApsuKraja2>5,5,ApsuKraja2)) %>%
  filter(!is.na(ApsuKraja2))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$ApsuKraja2,main="Original",xlab="Aspen volume")
hist(nogabali$ApsuKraja3,main="Limited",xlab="Aspen volume")
par(mfrow=c(1,1))
options(scipen=0)

p2i_rez=polygon2input(vector_data=nogabali,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsQuant_VolumeAspen.tif",
  value_field = "ApsuKraja3",
  fun="max",
  prepare=FALSE,
  restrict_to = clearcut_mask,
  restrict_values = 0,
  plot_result=TRUE,
  overwrite=TRUE)

p2i_rez
i2e_rez=input2egv(input="./RasterGrids_10m/2024/ForestsQuant_VolumeAspen.tif",
  egv_template = "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "sum",
  missing_job = "CoverOutput",
  output_bg = "./Templates/TemplateRasters/nulls_LV100m_10km.tif",
  outlocation = "./RasterGrids_100m/2024/RAW/",
  outfilename = "ForestsQuant_VolumeAspen-sum_cell.tif",
  layername = "evg_294",
  plot_final=TRUE)

i2e_rez
rm(p2i_rez)
rm(nogabali)
rm(apses)
rm(i2e_rez)

```

```

unlink("./RasterGrids_10m/2024/ForestsQuant_VolumeAspen.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsQuant_VolumeAspen-sum_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.295 ForestsQuant\_VolumeBirch-sum\_cell

**filename:** ForestsQuant\_VolumeBirch-sum\_cell.tif

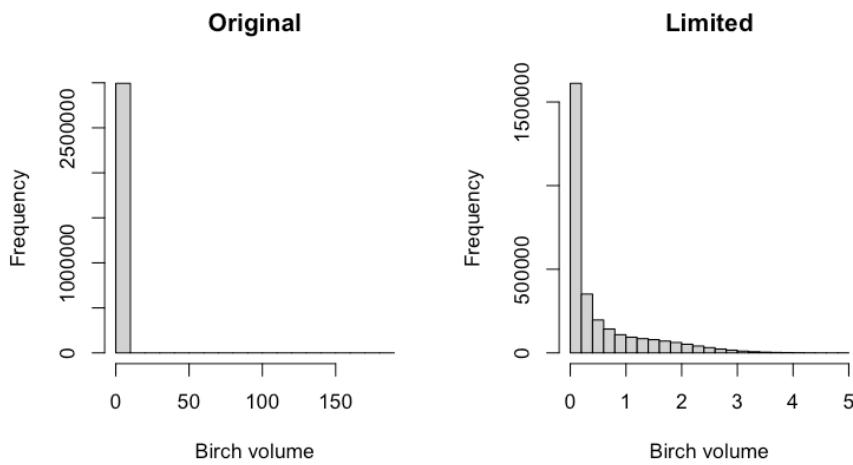
**layername:** egv\_295

**English name:** Timber volume of Birches within the analysis cell (1 ha)

**Latvian name:** Bērzu krāja analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

This EGV is prepared based on the information of timber volume of birch (species code: 4; see tree species codes in [Terminology and acronyms](#)) in the inventoried forest stands - [State Forest Service's State Forest Registry](#). This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised with the workflow `egvtools::polygon2input()`, restricting to pixels outside the clearcut mask. No background values are assigned during rasterisation. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating sum of pixel values. After the aggregation, cells with no forest information are filled with value 0. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
```

```

if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsQuant_VolumeBirch-sum_cell.tif egv_295 ----

berzi=c("4")
nogabali=mvr %>%
  mutate(BerzuKraja;ifelse(s10 %in% berzi, v10, 0)+ifelse(s11 %in% berzi,v11,0)+
    ifelse(s12 %in% berzi, v12,0)+ifelse(s13 %in% berzi,v13,0)+
    ifelse(s14 %in% berzi, v14,0)) %>%
  mutate(BerzuKraja2=BerzuKraja/10000*10*10) %>%
  mutate(BerzuKraja3=ifelse(BerzuKraja2>5,5,BerzuKraja2)) %>%
  filter(!is.na(BerzuKraja2))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$BerzuKraja2,main="Original",xlab="Birch volume")
hist(nogabali$BerzuKraja3,main="Limited",xlab="Birch volume")
par(mfrow=c(1,1))

```

```

options(scipen=0)

p2i_rez=polygon2input(vector_data=nogabali,
                       template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                       out_path = "./RasterGrids_10m/2024/",
                       file_name = "ForestsQuant_VolumeBirch.tif",
                       value_field = "BerzuKraja3",
                       fun="max",
                       prepare=FALSE,
                       restrict_to = clearcut_mask,
                       restrict_values = 0,
                       plot_result=TRUE,
                       overwrite=TRUE)

p2i_rez
i2e_rez=input2egv(input="./RasterGrids_10m/2024/ForestsQuant_VolumeBirch.tif",
                    egv_template = "./Templates/TemplateRasters/LV100m_10km.tif",
                    summary_function = "sum",
                    missing_job = "CoverOutput",
                    output_bg = "./Templates/TemplateRasters/nulls_LV100m_10km.tif",
                    outlocation = "./RasterGrids_100m/2024/Raw/",
                    outfilename = "ForestsQuant_VolumeBirch-sum_cell.tif",
                    layername = "egv_295",
                    plot_final=TRUE)

i2e_rez
rm(p2i_rez)
rm(nogabali)
rm(berzi)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsQuant_VolumeBirch.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsQuant_VolumeBirch-sum_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.296 ForestsQuant\_VolumeBlackAlder-sum\_cell

**filename:** ForestsQuant\_VolumeBlackAlder-sum\_cell.tif

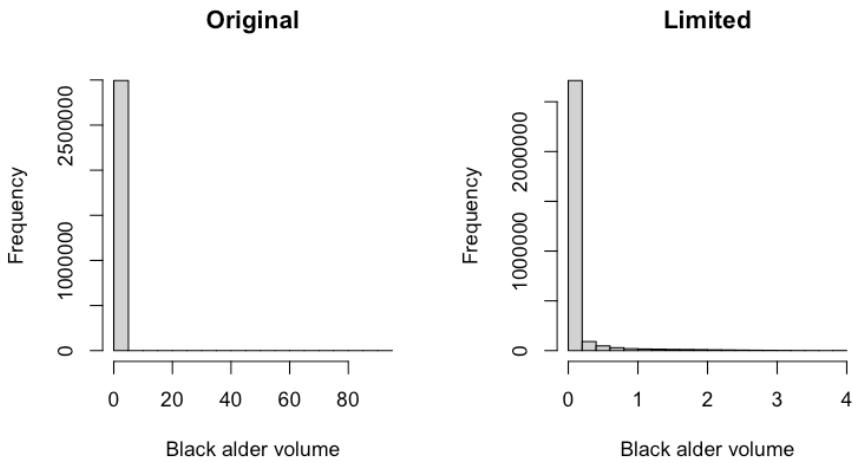
**layername:** egv\_296

**English name:** Timber volume of Black Alder within the analysis cell (1 ha)

**Latvian name:** Melnalkšņu krāja analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

This EGV is prepared based on the information of timber volume of black alder (species code: 6; see tree species codes in [Terminology and acronyms](#)) in the inventoried forest stands - [State Forest Service's State Forest Registry](#). This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised with the workflow `egvtools::polygon2input()`, restricting to pixels outside the clearcut mask. No background values are assigned during rasterisation. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating sum of pixel values. After the aggregation, cells with no forest information are filled with value 0. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
```

```

        filename=".~/RasterGrids_10m/2024/Mask_clearcuts.tif",
        overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsQuant_VolumeBlackAlder-sum_cell.tif      egv_296 ----

melnalksni=c("6")
nogabali=mvr %>%
  mutate(MeKraja=ifelse(s10 %in% melnalksni, v10, 0)+ifelse(s11 %in% melnalksni,v11,0)+
    ifelse(s12 %in% melnalksni, v12,0)+ifelse(s13 %in% melnalksni,v13,0)+
    ifelse(s14 %in% melnalksni, v14,0)) %>%
  mutate(MeKraja2=MeKraja/10000*10*10) %>%
  mutate(MeKraja3=ifelse(MeKraja2>4,4,MeKraja2)) %>%
  filter(!is.na(MeKraja2))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$MeKraja2,main="Original",xlab="Black alder volume")
hist(nogabali$MeKraja3,main="Limited",xlab="Black alder volume")
par(mfrow=c(1,1))
options(scipen=0)

p2i_rez=polygon2input(vector_data=nogabali,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsQuant_VolumeBlackAlder.tif",
  value_field = "MeKraja3",
  fun="max",
  prepare=FALSE,
  restrict_to = clearcut_mask,
  restrict_values = 0,
  plot_result=TRUE,
  overwrite=TRUE)

p2i_rez
i2e_rez=input2egv(input=".~/RasterGrids_10m/2024/ForestsQuant_VolumeBlackAlder.tif",
  egv_template = "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "sum",
  missing_job = "CoverOutput",
  output_bg = "./Templates/TemplateRasters/nulls_LV100m_10km.tif",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "ForestsQuant_VolumeBlackAlder-sum_cell.tif",
  layername = "egv_296",
  plot_final=TRUE)

i2e_rez
rm(p2i_rez)
rm(nogabali)
rm(melnalksni)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsQuant_VolumeBlackAlder.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsQuant_VolumeBlackAlder-sum_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)

```

```

centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.297 ForestsQuant\_VolumeBorealDeciduousOther-sum\_cell

**filename:** ForestsQuant\_VolumeBorealDeciduousOther-sum\_cell.tif

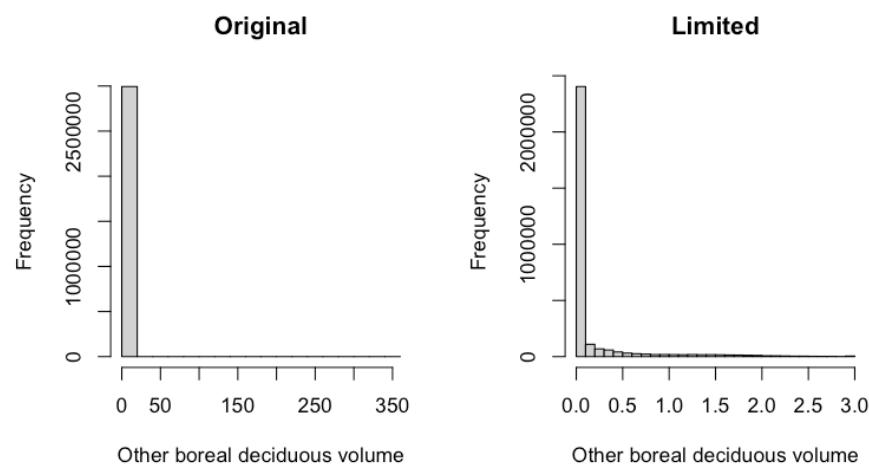
**layername:** egv\_297

**English name:** Timber volume of Other Boreal Deciduous trees within the analysis cell (1 ha)

**Latvian name:** Citu šaurlapju krāja analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

This EGV is prepared based on the information of timber volume of Boreal deciduous tree species not separately described with own EGVs (species codes: 9, 20, 21, 32, 35; see tree species codes in [Terminology and acronyms](#)) in the inventoried forest stands - [State Forest Service's State Forest Registry](#). This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised with the workflow `egvtools::polygon2input()`, restricting to pixels outside the clearcut mask. No background values are assigned during rasterisation. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating sum of pixel values. After the aggregation, cells with no forest information are filled with value 0. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

```

```

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsQuant_VolumeBorealDeciduousOther-sum_cell.tif egv_297 ----

sl_citi=c("9","20","21","32","35")
nogabali=mvr %>%
  mutate(SaurlapjuCKraja=ifelse(s10 %in% sl_citi, v10, 0)+ifelse(s11 %in% sl_citi,v11,0)+
    ifelse(s12 %in% sl_citi, v12,0)+ifelse(s13 %in% sl_citi,v13,0)+
    ifelse(s14 %in% sl_citi, v14,0)) %>%
  mutate(SaurlapjuCKraja2=SaurlapjuCKraja/10000*10*10) %>%
  mutate(SaurlapjuCKraja3=ifelse(SaurlapjuCKraja2>3,3,SaurlapjuCKraja2)) %>%
  filter(!is.na(SaurlapjuCKraja2))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$SaurlapjuCKraja2,main="Original",xlab="Other Boreal deciduous volume")
hist(nogabali$SaurlapjuCKraja3,main="Limited",xlab="Other Boreal deciduous volume")
par(mfrow=c(1,1))
options(scipen=0)

p2i_rez=polygon2input(vector_data=nogabali,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsQuant_VolumeBorealDeciduousOther.tif",
  value_field = "SaurlapjuCKraja3",
  fun="max",
  prepare=FALSE,

```

```

        restrict_to = clearcut_mask,
        restrict_values = 0,
        plot_result=TRUE,
        overwrite=TRUE)
p2i_rez
i2e_rez=input2egv(input="./RasterGrids_10m/2024/ForestsQuant_VolumeBorealDeciduousOther.tif",
                    evg_template = "./Templates/TemplateRasters/LV100m_10km.tif",
                    summary_function = "sum",
                    missing_job = "CoverOutput",
                    output_bg = "./Templates/TemplateRasters/nulls_LV100m_10km.tif",
                    outlocation = "./RasterGrids_100m/2024/Raw/",
                    outfilename = "ForestsQuant_VolumeBorealDeciduousOther-sum_cell.tif",
                    layername = "egv_297",
                    plot_final=TRUE)
i2e_rez
rm(p2i_rez)
rm(nogabali)
rm(sl_citi)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsQuant_VolumeBorealDeciduousOther.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsQuant_VolumeBorealDeciduousOther-sum_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.298 ForestsQuant\_VolumeBorealDeciduousTotal-sum\_cell

**filename:** ForestsQuant\_VolumeBorealDeciduousTotal-sum\_cell.tif

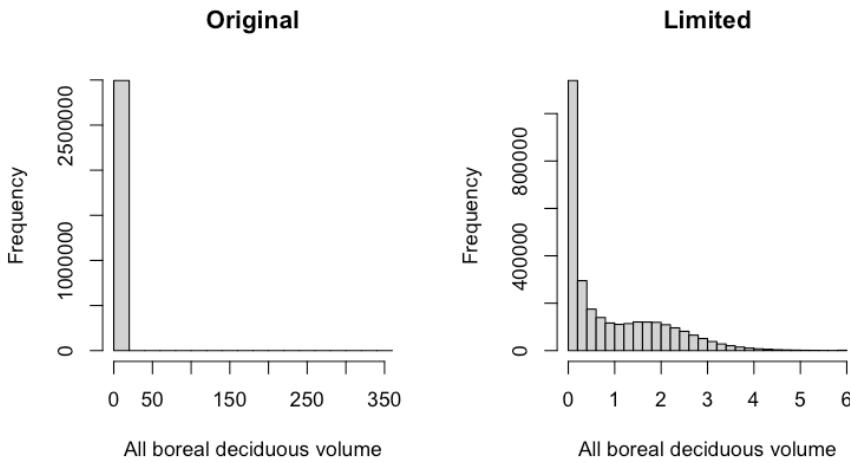
**layername:** egv\_298

**English name:** Timber volume of Boreal Deciduous trees within the analysis cell (1 ha)

**Latvian name:** Šaurlapju krāja analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

This EGV is prepared based on the information of timber volume of Boreal deciduous tree species (species codes: 4, 6, 8, 9, 19, 20, 21, 32, 35, 68; see tree species codes in [Terminology and acronyms](#)) in the inventoried forest stands - [State Forest Service's State Forest Registry](#). This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised with the workflow `egvtools::polygon2input()`, restricting to pixels outside the clearcut mask. No background values are assigned during rasterisation. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating sum of pixel values. After the aggregation, cells with no forest information are filled with value 0. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
```

```

        filename=".~/RasterGrids_10m/2024/Mask_clearcuts.tif",
        overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsQuant_VolumeBorealDeciduousTotal-sum_cell.tif  egv_298 ----

sl_visi=c("4","6","8","9","19","20","21","32","35","68")
nogabali=mvr %>%
  mutate(SaurlapjuVKraja;ifelse(s10 %in% sl_vis, v10, 0)+ifelse(s11 %in% sl_vis, v11, 0) +
    ifelse(s12 %in% sl_vis, v12, 0)+ifelse(s13 %in% sl_vis, v13, 0) +
    ifelse(s14 %in% sl_vis, v14, 0)) %>%
  mutate(SaurlapjuVKraja2=SaurlapjuVKraja/10000*10*10) %>%
  mutate(SaurlapjuVKraja3;ifelse(SaurlapjuVKraja2>6,6,SaurlapjuVKraja2)) %>%
  filter(!is.na(SaurlapjuVKraja2))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$SaurlapjuVKraja2,main="Original",xlab="All Boreal deciduous volume")
hist(nogabali$SaurlapjuVKraja3,main="Limited",xlab="All Boreal deciduous volume")
par(mfrow=c(1,1))
options(scipen=0)

p2i_rez=polygon2input(vector_data=nogabali,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsQuant_VolumeBorealDeciduousTotal.tif",
  value_field = "SaurlapjuVKraja3",
  fun="max",
  prepare=FALSE,
  restrict_to = clearcut_mask,
  restrict_values = 0,
  plot_result=TRUE,
  overwrite=TRUE)

p2i_rez
i2e_rez=input2egv(input="./RasterGrids_10m/2024/ForestsQuant_VolumeBorealDeciduousTotal.tif",
  egv_template = "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "sum",
  missing_job = "CoverOutput",
  output_bg = "./Templates/TemplateRasters/nulls_LV100m_10km.tif",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "ForestsQuant_VolumeBorealDeciduousTotal-sum_cell.tif",
  layername = "egv_298",
  plot_final=TRUE)

i2e_rez
rm(p2i_rez)
rm(nogabali)
rm(sl_vis)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsQuant_VolumeBorealDeciduousTotal.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsQuant_VolumeBorealDeciduousTotal-sum_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)

```

```

slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.299 ForestsQuant\_VolumeConiferous-sum\_cell

**filename:** ForestsQuant\_VolumeConiferous-sum\_cell.tif

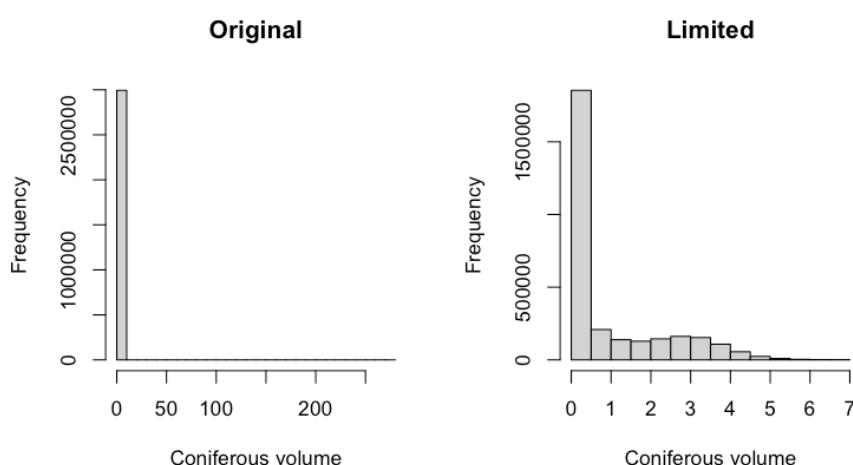
**layername:** egv\_299

**English name:** Timber volume of Coniferous trees within the analysis cell (1 ha)

**Latvian name:** Skujkoku krāja analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

This EGV is prepared based on the information of timber volume of coniferous tree species (species codes: 1, 14, 22, 3, 13, 15, 23, 28; see tree species codes in [Terminology and acronyms](#)) in the inventoried forest stands - [State Forest Service's State Forest Registry](#). This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised with the workflow `egvtools::polygon2input()`, restricting to pixels outside the clearcut mask. No background values are assigned during rasterisation. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating sum of pixel values. After the aggregation, cells with no forest information are filled with value 0. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

```

```

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsQuant_VolumeConiferous-sum_cell.tif      egv_299 ----

skujkoki=c("1","14","22","3","13","15","23","28")
nogabali=mvr %>%
  mutate(SkujkokuKraja=ifelse(s10 %in% skujkoki, v10, 0)+ifelse(s11 %in% skujkoki,v11,0) +
    ifelse(s12 %in% skujkoki, v12,0)+ifelse(s13 %in% skujkoki,v13,0) +
    ifelse(s14 %in% skujkoki, v14,0)) %>%
  mutate(SkujkokuKraja2=SkujkokuKraja/10000*10*10) %>%
  mutate(SkujkokuKraja3=ifelse(SkujkokuKraja2>7,7,SkujkokuKraja2)) %>%
  filter(!is.na(SkujkokuKraja2))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$SkujkokuKraja2,main="Original",xlab="Coniferous volume")
hist(nogabali$SkujkokuKraja3,main="Limited",xlab="Coniferous volume")
par(mfrow=c(1,1))
options(scipen=0)

p2i_rez=polygon2input(vector_data=nogabali,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsQuant_VolumeConiferous.tif",
  value_field = "SkujkokuKraja3",
  fun="max",

```

```

        prepare=FALSE,
        restrict_to = clearcut_mask,
        restrict_values = 0,
        plot_result=TRUE,
        overwrite=TRUE)
p2i_rez
i2e_rez=input2egv(input="./RasterGrids_10m/2024/ForestsQuant_VolumeConiferous.tif",
                    egv_template = "./Templates/TemplateRasters/LV100m_10km.tif",
                    summary_function = "sum",
                    missing_job = "CoverOutput",
                    output_bg = "./Templates/TemplateRasters/nulls_LV100m_10km.tif",
                    outlocation = "./RasterGrids_100m/2024/Raw/",
                    outfilename = "ForestsQuant_VolumeConiferous-sum_cell.tif",
                    layername = "egv_299",
                    plot_final=TRUE)
i2e_rez
rm(p2i_rez)
rm(nogabali)
rm(skujkoki)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsQuant_VolumeConiferous.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsQuant_VolumeConiferous-sum_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.300 ForestsQuant\_VolumeOak-sum\_cell

**filename:** ForestsQuant\_VolumeOak-sum\_cell.tif

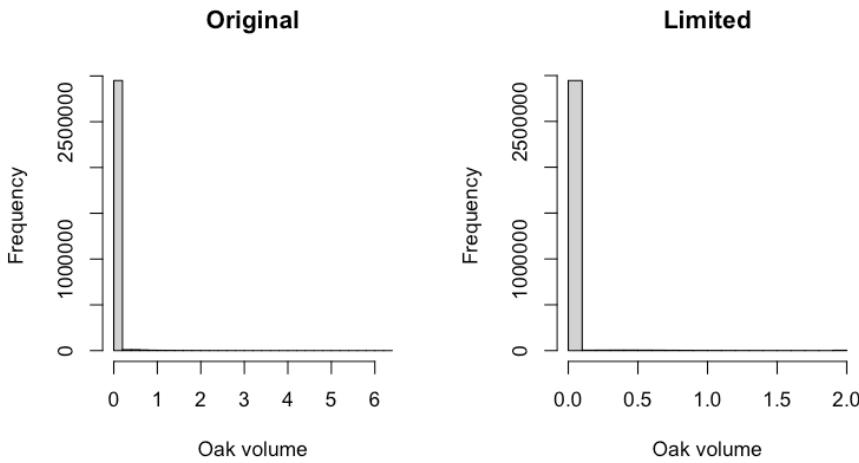
**layername:** egv\_300

**English name:** Timber volume of Oaks within the analysis cell (1 ha)

**Latvian name:** Ozolu krāja analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

This EGV is prepared based on the information of timber volume of oaks (species codes: 10, 61; see tree species codes in [Terminology and acronyms](#)) in the inventoried forest stands - [State Forest Service's State Forest Registry](#). This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised with the workflow `egvtools::polygon2input()`, restricting to pixels outside the clearcut mask. No background values are assigned during rasterisation. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating sum of pixel values. After the aggregation, cells with no forest information are filled with value 0. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
```

```

        filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
        overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsQuant_VolumeOak-sum_cell.tif    egv_300 ----
ozoli=c("10","61")
nogabali=mvr %>%
  mutate(OzoluKraja=ifelse(s10 %in% ozoli, v10, 0)+ifelse(s11 %in% ozoli,v11,0)+  

    ifelse(s12 %in% ozoli, v12,0)+ifelse(s13 %in% ozoli,v13,0)+  

    ifelse(s14 %in% ozoli, v14,0)) %>%
  mutate(OzoluKraja2=OzoluKraja/10000*10*10) %>%
  mutate(OzoluKraja3=ifelse(OzoluKraja2>2,2,OzoluKraja2)) %>%
  filter(!is.na(OzoluKraja2))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$OzoluKraja2,main="Original",xlab="Oak volume")
hist(nogabali$OzoluKraja3,main="Limited",xlab="Oak volume")
par(mfrow=c(1,1))
options(scipen=0)

p2i_rez=polygon2input(vector_data=nogabali,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsQuant_VolumeOak.tif",
  value_field = "OzoluKraja3",
  fun="max",
  prepare=FALSE,
  restrict_to = clearcut_mask,
  restrict_values = 0,
  plot_result=TRUE,
  overwrite=TRUE)

p2i_rez
i2e_rez=input2egv(input="./RasterGrids_10m/2024/ForestsQuant_VolumeOak.tif",
  egv_template = "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "sum",
  missing_job = "CoverOutput",
  output_bg = "./Templates/TemplateRasters/nulls_LV100m_10km.tif",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "ForestsQuant_VolumeOak-sum_cell.tif",
  layername = "egv_300",
  plot_final=TRUE)

i2e_rez
rm(p2i_rez)
rm(nogabali)
rm(ozoli)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsQuant_VolumeOak.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsQuant_VolumeOak-sum_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)

```

```

centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.301 ForestsQuant\_VolumeOakMaple-sum\_cell

**filename:** ForestsQuant\_VolumeOakMaple-sum\_cell.tif

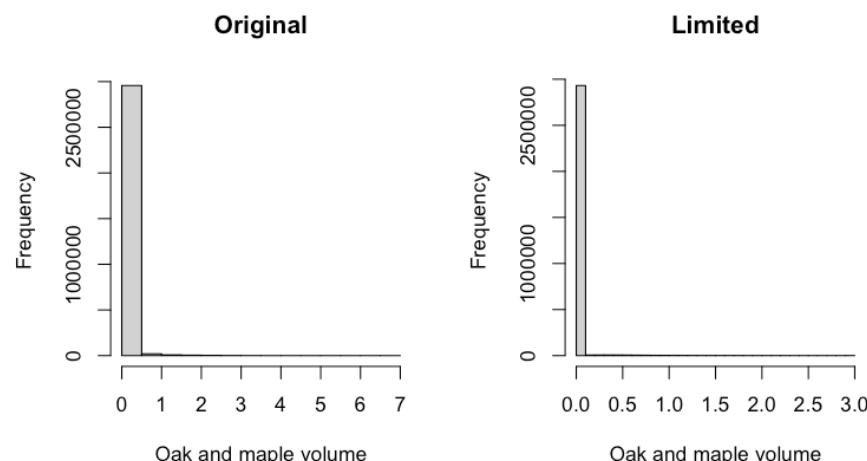
**layername:** egv\_301

**English name:** Timber volume of Oaks, Maples within the analysis cell (1 ha)

**Latvian name:** Ozolu, kļavu krāja analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

This EGV is prepared based on the information of timber volume of oaks and maples (species codes: 10, 61, 24, 63; see tree species codes in [Terminology and acronyms](#)) in the inventoried forest stands - [State Forest Service's State Forest Registry](#). This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised with the workflow `egvtools::polygon2input()`, restricting to pixels outside the clearcut mask. No background values are assigned during rasterisation. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating sum of pixel values. After the aggregation, cells with no forest information are filled with value 0. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----

```

```

template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsQuant_VolumeOakMaple-sum_cell.tif  egv_301 ----
ozolklavas=c("10","61","24","63")
nogabali=mvr %>%
  mutate(OzolKlavuKraja=ifelse(s10 %in% ozolklavas, v10, 0)+ifelse(s11 %in% ozolklavas,v11,0) +
    ifelse(s12 %in% ozolklavas, v12,0)+ifelse(s13 %in% ozolklavas,v13,0) +
    ifelse(s14 %in% ozolklavas, v14,0)) %>%
  mutate(OzolKlavuKraja2=OzolKlavuKraja/10000*10*10) %>%
  mutate(OzolKlavuKraja3=ifelse(OzolKlavuKraja2>3,3,OzolKlavuKraja2)) %>%
  filter(!is.na(OzolKlavuKraja2))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$OzolKlavuKraja2,main="Original",xlab="Oak and maple volume")
hist(nogabali$OzolKlavuKraja3,main="Limited",xlab="Oak and maple volume")
par(mfrow=c(1,1))
options(scipen=0)

p2i_rez=polygon2input(vector_data=nogabali,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsQuant_VolumeOakMaple.tif",
  value_field = "OzolKlavuKraja3",
  fun="max",
  prepare=FALSE,
  restrict_to = clearcut_mask,
  restrict_values = 0,

```

```

        plot_result=TRUE,
        overwrite=TRUE)
p2i_rez
i2e_rez=input2egv(input="./RasterGrids_10m/2024/ForestsQuant_VolumeOakMaple.tif",
    egv_template = "./Templates/TemplateRasters/LV100m_10km.tif",
    summary_function = "sum",
    missing_job = "CoverOutput",
    output_bg = "./Templates/TemplateRasters/nulls_LV100m_10km.tif",
    outlocation = "./RasterGrids_100m/2024/RAW/",
    outfilename = "ForestsQuant_VolumeOakMaple-sum_cell.tif",
    layername = "egv_301",
    plot_final=TRUE)

i2e_rez
rm(p2i_rez)
rm(nogabali)
rm(ozolklavas)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsQuant_VolumeOakMaple.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsQuant_VolumeOakMaple-sum_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.302 ForestsQuant\_VolumePine-sum\_cell

**filename:** ForestsQuant\_VolumePine-sum\_cell.tif

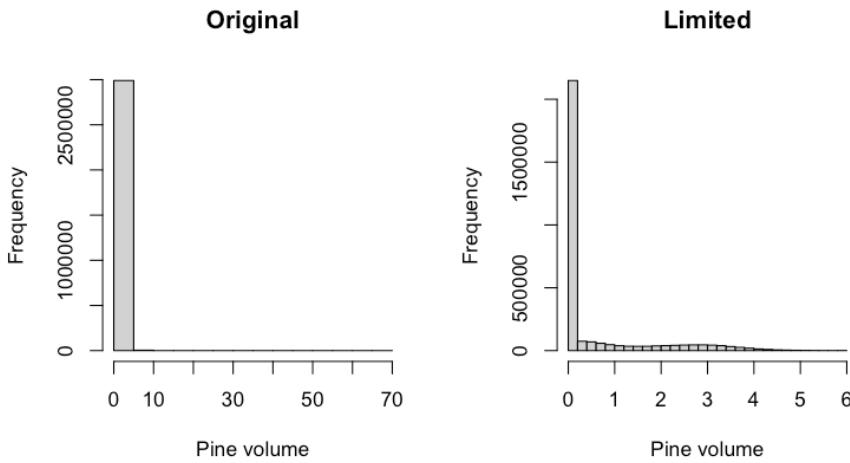
**layername:** egv\_302

**English name:** Timber volume of Pines within the analysis cell (1 ha)

**Latvian name:** Priežu krāja analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

This EGV is prepared based on the information of timber volume of pines (species codes: 1, 14, 22; see tree species codes in [Terminology and acronyms](#)) in the inventoried forest stands - [State Forest Service's State Forest Registry](#). This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised with the workflow `egvtools::polygon2input()`, restricting to pixels outside the clearcut mask. No background values are assigned during rasterisation. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating sum of pixel values. After the aggregation, cells with no forest information are filled with value 0. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,

```

```

        filename=".~/RasterGrids_10m/2024/Mask_clearcuts.tif",
        overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsQuant_VolumePine-sum_cell.tif  egv_302 ----

priedes=c("1","14","22")
nogabali=mvr %>%
  mutate(PriezuKraja=ifelse(s10 %in% priedes, v10, 0)+ifelse(s11 %in% priedes,v11,0) +
    ifelse(s12 %in% priedes, v12,0)+ifelse(s13 %in% priedes,v13,0) +
    ifelse(s14 %in% priedes, v14,0)) %>%
  mutate(PriezuKraja2=PriezuKraja/10000*10*10) %>%
  mutate(PriezuKraja3=ifelse(PriezuKraja2>6,6,PriezuKraja2)) %>%
  filter(!is.na(PriezuKraja2))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$PriezuKraja2,main="Original",xlab="Pine volume")
hist(nogabali$PriezuKraja3,main="Limited",xlab="Pine volume")
par(mfrow=c(1,1))
options(scipen=0)

p2i_rez=polygon2input(vector_data=nogabali,
  template_path = ".~/Templates/TemplateRasters/LV10m_10km.tif",
  out_path = ".~/RasterGrids_10m/2024/",
  file_name = "ForestsQuant_VolumePine.tif",
  value_field = "PriezuKraja3",
  fun="max",
  prepare=FALSE,
  restrict_to = clearcut_mask,
  restrict_values = 0,
  plot_result=TRUE,
  overwrite=TRUE)

p2i_rez
i2e_rez=input2egv(input=".~/RasterGrids_10m/2024/ForestsQuant_VolumePine.tif",
  egv_template = ".~/Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "sum",
  missing_job = "CoverOutput",
  output_bg = ".~/Templates/TemplateRasters/nulls_LV100m_10km.tif",
  outlocation = ".~/RasterGrids_100m/2024/Raw/",
  outfilename = "ForestsQuant_VolumePine-sum_cell.tif",
  layername = "egv_302",
  plot_final=TRUE)

i2e_rez
rm(p2i_rez)
rm(nogabali)
rm(priedes)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsQuant_VolumePine.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsQuant_VolumePine-sum_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)

```

```

videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

### 6.303 ForestsQuant\_VolumeSpruce-sum\_cell

**filename:** ForestsQuant\_VolumeSpruce-sum\_cell.tif

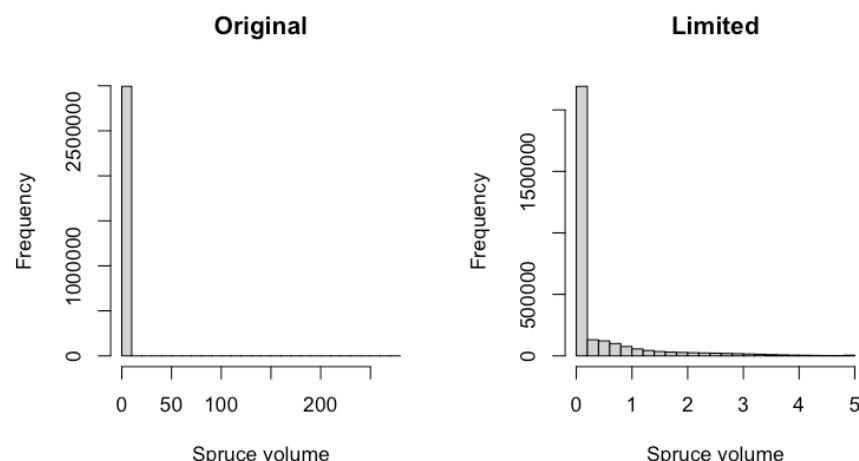
**layername:** egv\_303

**English name:** Timber volume of Spruces within the analysis cell (1 ha)

**Latvian name:** Egļu krāja analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

This EGV is prepared based on the information of timber volume of spruces (species codes: 3, 13, 15, 23, 28; see tree species codes in [Terminology and acronyms](#)) in the inventoried forest stands - [State Forest Service's State Forest Registry](#). This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised with the workflow `egvtools::polygon2input()`, restricting to pixels outside the clearcut mask. No background values are assigned during rasterisation. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating sum of pixel values. After the aggregation, cells with no forest information are filled with value 0. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

```

```

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsQuant_VolumeSpruce-sum_cell.tif      egv_303 ----

egles=c("3","13","15","23","28")
nogabali=mvr %>%
  mutate(EgluKraja=ifelse(s10 %in% egles, v10, 0)+ifelse(s11 %in% egles,v11,0)+
    ifelse(s12 %in% egles, v12,0)+ifelse(s13 %in% egles,v13,0)+
    ifelse(s14 %in% egles, v14,0)) %>%
  mutate(EgluKraja2=EgluKraja/10000*10*10) %>%
  mutate(EgluKraja3=ifelse(EgluKraja2>5,5,EgluKraja2)) %>%
  filter(!is.na(EgluKraja2))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$EgluKraja2,main="Original",xlab="Spruce volume")
hist(nogabali$EgluKraja3,main="Limited",xlab="Spruce volume")
par(mfrow=c(1,1))
options(scipen=0)

p2i_rez=polygon2input(vector_data=nogabali,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsQuant_VolumeSpruce.tif",
  value_field = "EgluKraja3",
  fun="max",
  prepare=FALSE,

```

```

        restrict_to = clearcut_mask,
        restrict_values = 0,
        plot_result=TRUE,
        overwrite=TRUE)
p2i_rez
i2e_rez=input2egv(input="./RasterGrids_10m/2024/ForestsQuant_VolumeSpruce.tif",
                    evg_template = "./Templates/TemplateRasters/LV100m_10km.tif",
                    summary_function = "sum",
                    missing_job = "CoverOutput",
                    output_bg = "./Templates/TemplateRasters/nulls_LV100m_10km.tif",
                    outlocation = "./RasterGrids_100m/2024/Raw/",
                    outfilename = "ForestsQuant_VolumeSpruce-sum_cell.tif",
                    layername = "egv_303",
                    plot_final=TRUE)
i2e_rez
rm(p2i_rez)
rm(nogabali)
rm(egles)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsQuant_VolumeSpruce.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsQuant_VolumeSpruce-sum_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.304 ForestsQuant\_VolumeTemperateDeciduousTotal-sum\_cell

**filename:** ForestsQuant\_VolumeTemperateDeciduousTotal-sum\_cell.tif

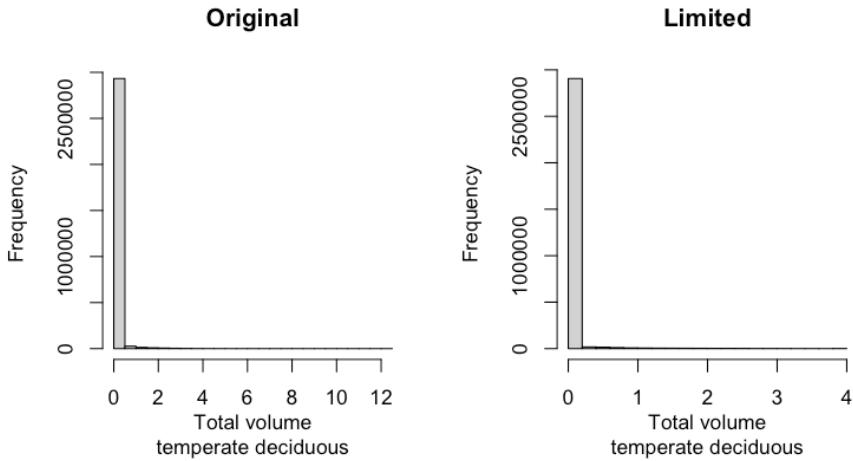
**layername:** egv\_304

**English name:** Timber volume of Temperate Deciduous trees within the analysis cell (1 ha)

**Latvian name:** Platlapju krāja analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

This EGV is prepared based on the information of timber volume of temperate deciduous tree species (species codes: 10, 11, 12, 16, 17, 18, 24, 25, 26, 27, 29, 50, 61, 62, 63, 64, 65, 66, 67, 69; see tree species codes in [Terminology and acronyms](#)) in the inventoried forest stands - [State Forest Service's State Forest Registry](#). This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised with the workflow `egvtools::polygon2input()`, restricting to pixels outside the clearcut mask. No background values are assigned during rasterisation. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating sum of pixel values. After the aggregation, cells with no forest information are filled with value 0. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
```

```

        filename=".~/RasterGrids_10m/2024/Mask_clearcuts.tif",
        overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsQuant_VolumeTemperateDeciduousTotal-sum_cell.tif    egv_304 ----

platlapji=c("10","11","12","16","17","18","24","25","26","27","29","50",
           "61","62","63","64","65","66","67","69")
nogabali=mvr %>%
  mutate(PlatKraja=ifelse(s10 %in% platlapji, v10, 0)+ifelse(s11 %in% platlapji,v11,0) +
    ifelse(s12 %in% platlapji, v12,0)+ifelse(s13 %in% platlapji,v13,0) +
    ifelse(s14 %in% platlapji, v14,0)) %>%
  mutate(PlatKraja2=PlatKraja/10000*10*10) %>%
  mutate(PlatKraja3=ifelse(PlatKraja2>4,4,PlatKraja2)) %>%
  filter(!is.na(PlatKraja2))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$PlatKraja2,main="Original",xlab="Total volume\ntemperate deciduous")
hist(nogabali$PlatKraja3,main="Limited",xlab="Total volume\ntemperate deciduous")
par(mfrow=c(1,1))
options(scipen=0)

p2i_rez=polygon2input(vector_data=nogabali,
                       template_path = ".~/Templates/TemplateRasters/LV10m_10km.tif",
                       out_path = ".~/RasterGrids_10m/2024/",
                       file_name = "ForestsQuant_VolumeTemperateDeciduousTotal.tif",
                       value_field = "PlatKraja3",
                       fun="max",
                       prepare=FALSE,
                       restrict_to = clearcut_mask,
                       restrict_values = 0,
                       plot_result=TRUE,
                       overwrite=TRUE)

p2i_rez
i2e_rez=input2egv(input=
  .~/RasterGrids_10m/2024/ForestsQuant_VolumeTemperateDeciduousTotal.tif",
  egv_template = ".~/Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "sum",
  missing_job = "CoverOutput",
  output_bg = ".~/Templates/TemplateRasters/nulls_LV100m_10km.tif",
  outlocation = ".~/RasterGrids_100m/2024/Raw/",
  outfilename = "ForestsQuant_VolumeTemperateDeciduousTotal-sum_cell.tif",
  layername = "egv_304",
  plot_final=TRUE)

i2e_rez
rm(p2i_rez)
rm(nogabali)
rm(platlapji)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsQuant_VolumeTemperateDeciduousTotal.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsQuant_VolumeTemperateDeciduousTotal-sum_cell.tif"

```

```

ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.305 ForestsQuant\_VolumeTemperateWithoutOak-sum\_cell

**filename:** ForestsQuant\_VolumeTemperateWithoutOak-sum\_cell.tif

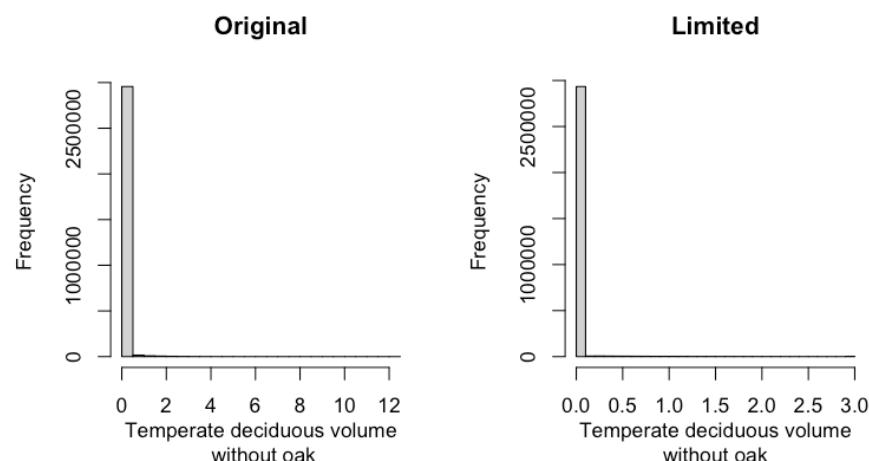
**layername:** evg\_305

**English name:** Timber volume of Temperate Deciduous trees (without oaks) within the analysis cell (1 ha)

**Latvian name:** Platlapju (bez ozoliem) krāja analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

This EGV is prepared based on the information of timber volume of temperate deciduous tree species except oaks (species codes: 11, 12, 16, 17, 18, 24, 25, 26, 27, 29, 50, 62, 63, 64, 65, 66, 67, 69; see tree species codes in [Terminology and acronyms](#)) in the inventoried forest stands - [State Forest Service's State Forest Registry](#). This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised with the workflow `egvtools::polygon2input()`, restricting to pixels outside the clearcut mask. No background values are assigned during rasterisation. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating sum of pixel values. After the aggregation, cells with no forest information are filled with value 0. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}

```

```

if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsQuant_VolumeTemperateWithoutOak-sum_cell.tif  egv_305 ----
neozoli=c("11","12","16","17","18","24","25","26","27","29","50",
         "62","63","64","65","66","67","69")
nogabali=mvr %>%
  mutate(BezOzoluKraja=ifelse(s10 %in% neozoli, v10, 0)+ifelse(s11 %in% neozoli,v11,0)+
    ifelse(s12 %in% neozoli, v12,0)+ifelse(s13 %in% neozoli,v13,0)+
    ifelse(s14 %in% neozoli, v14,0)) %>%
  mutate(BezOzoluKraja2=BezOzoluKraja/10000*10*10) %>%
  mutate(BezOzoluKraja3=ifelse(BezOzoluKraja2>3,3,BezOzoluKraja2)) %>%
  filter(!is.na(BezOzoluKraja2))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$BezOzoluKraja2,main="Original",xlab="Temperate deciduous volume\n without oak")
hist(nogabali$BezOzoluKraja3,main="Limited",xlab="Temperate deciduous volume\nwithout oak")
par(mfrow=c(1,1))
options(scipen=0)

p2i_rez=polygon2input(vector_data=nogabali,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",

```

```

    file_name = "ForestsQuant_VolumeTemperateWithoutOak.tif",
    value_field = "BezOzoluKraja3",
    fun="max",
    prepare=FALSE,
    restrict_to = clearcut_mask,
    restrict_values = 0,
    plot_result=TRUE,
    overwrite=TRUE)
p2i_rez
i2e_rez=input2egv(input="./RasterGrids_10m/2024/ForestsQuant_VolumeTemperateWithoutOak.tif",
                    evg_template = "./Templates/TemplateRasters/LV100m_10km.tif",
                    summary_function = "sum",
                    missing_job = "CoverOutput",
                    output_bg = "./Templates/TemplateRasters/nulls_LV100m_10km.tif",
                    outlocation = "./RasterGrids_100m/2024/Raw/",
                    outfilename = "ForestsQuant_VolumeTemperateWithoutOak-sum_cell.tif",
                    layername = "egv_305",
                    plot_final=TRUE)
i2e_rez
rm(p2i_rez)
rm(nogabali)
rm(neozoli)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsQuant_VolumeTemperateWithoutOak.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsQuant_VolumeTemperateWithoutOak-sum_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.306 ForestsQuant\_VolumeTemperateWithoutOakMaple-sum\_cell

**filename:** ForestsQuant\_VolumeTemperateWithoutOakMaple-sum\_cell.tif

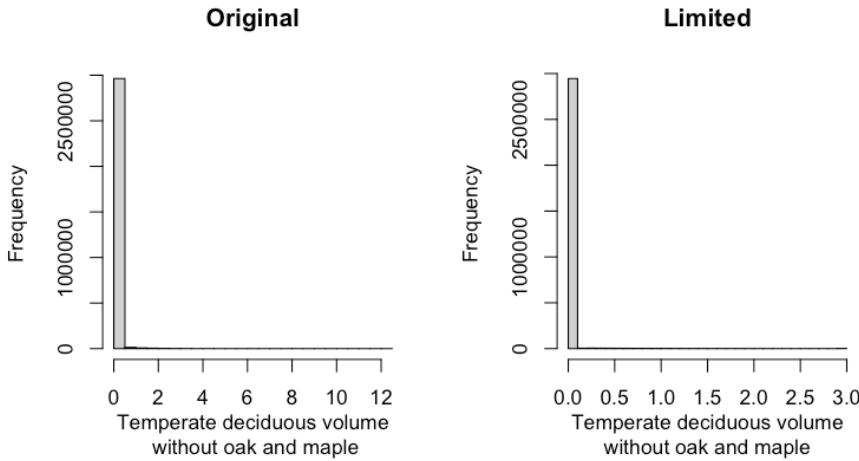
**layername:** egv\_306

**English name:** Timber volume of Temperate Deciduous trees (without oaks, maples) within the analysis cell (1 ha)

**Latvian name:** Platlapju (bez ozoliem, kļavām) krāja analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

This EGV is prepared based on the information of timber volume of teperate deciduous trees except oaks and maples (species codes: 11, 12, 16, 17, 18, 25, 26, 27, 29, 50, 62, 64, 65, 66, 67, 69; see tree species codes in [Terminology and acronyms](#)) in the inventoried forest stands - [State Forest Service's State Forest Registry](#). This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised with the workflow `egvtools::polygon2input()`, restricting to pixels outside the clearcut mask. No background values are assigned during rasterisation. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating sum of pixel values. After the aggregation, cells with no forest information are filled with value 0. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,

```

```

        filename=".~/RasterGrids_10m/2024/Mask_clearcuts.tif",
        overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsQuant_VolumeTemperateWithoutOakMaple-sum_cell.tif  egv_306 ----
neozolklavas=c("11","12","16","17","18","25","26","27","29","50",
               "62","64","65","66","67","69")
nogabali=mvr %>%
  mutate(BezOzolKlavuKraja=ifelse(s10 %in% neozolklavas, v10, 0)+ifelse(s11 %in%
    ↪ neozolklavas,v11,0) +
    ifelse(s12 %in% neozolklavas, v12,0)+ifelse(s13 %in% neozolklavas,v13,0) +
    ifelse(s14 %in% neozolklavas, v14,0)) %>%
  mutate(BezOzolKlavuKraja2=BezOzolKlavuKraja/1000*10*10) %>%
  mutate(BezOzolKlavuKraja3=ifelse(BezOzolKlavuKraja2>3,3,BezOzolKlavuKraja2)) %>%
  filter(!is.na(BezOzolKlavuKraja2))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$BezOzolKlavuKraja2,main="Original",xlab="Temperate deciduous volume\n without
  ↪ oak and maple")
hist(nogabali$BezOzolKlavuKraja3,main="Limited",xlab="Temperate deciduous volume\nwithout oak
  ↪ and maple")
par(mfrow=c(1,1))
options(scipen=0)

p2i_rez=polygon2input(vector_data=nogabali,
                       template_path = ".~/Templates/TemplateRasters/LV10m_10km.tif",
                       out_path = ".~/RasterGrids_10m/2024/",
                       file_name = "ForestsQuant_VolumeTemperateWithoutOakMaple.tif",
                       value_field = "BezOzolKlavuKraja3",
                       fun="max",
                       prepare=FALSE,
                       restrict_to = clearcut_mask,
                       restrict_values = 0,
                       plot_result=TRUE,
                       overwrite=TRUE)

p2i_rez
i2e_rez=input2egv(input=]
  ↪ ".~/RasterGrids_10m/2024/ForestsQuant_VolumeTemperateWithoutOakMaple.tif",
  egv_template = ".~/Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "sum",
  missing_job = "CoverOutput",
  output_bg = ".~/Templates/TemplateRasters/nulls_LV100m_10km.tif",
  outlocation = ".~/RasterGrids_100m/2024/RAW/",
  outfilename = "ForestsQuant_VolumeTemperateWithoutOakMaple-sum_cell.tif",
  layername = "egv_306",
  plot_final=TRUE)

i2e_rez
rm(p2i_rez)
rm(nogabali)
rm(neozolklavas)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsQuant_VolumeTemperateWithoutOakMaple.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

nosaukums="ForestsQuant_VolumeTemperateWithoutOakMaple-sum_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.307 ForestsQuant\_VolumeTotal-sum\_cell

**filename:** ForestsQuant\_VolumeTotal-sum\_cell.tif

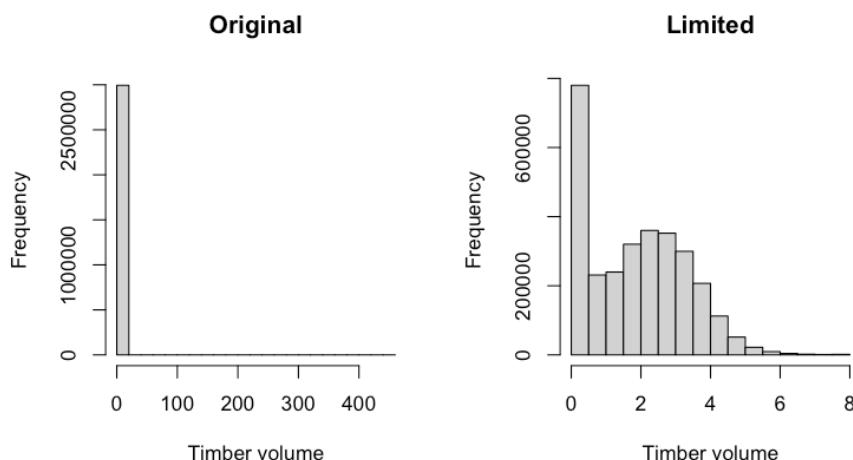
**layername:** egv\_307

**English name:** Timber volume within the analysis cell (1 ha)

**Latvian name:** Kopējā krāja analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

This EGV is prepared based on the information of timber volume in the inventoried forest stands - [State Forest Service's State Forest Registry](#). This attribute has some extreme values. We chose to limit them to the nearest integer showing only minimal accumulation in histogram.



Resulting values at polygon geometries are rasterised with the workflow `egvtools::polygon2input()`, restricting to pixels outside the clearcut mask. No background values are assigned during rasterisation. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()` by calculating sum of pixel values. After the aggregation, cells with no forest information are filled with value 0. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}

```

```

if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsQuant_VolumeTotal-sum_cell.tif egv_307 ----

nogabali=mvr %>%
  mutate(KopejaKraja=v10+v11+v12+v13+v14) %>%
  mutate(KopejaKraja2=KopejaKraja/10000*10*10) %>%
  mutate(KopejaKraja3=ifelse(KopejaKraja2>8,8,KopejaKraja2)) %>%
  filter(!is.na(KopejaKraja2))

par(mfrow=c(1,2))
options(scipen=999)
hist(nogabali$KopejaKraja2,main="Original",xlab="Timber volume")
hist(nogabali$KopejaKraja3,main="Limited",xlab="Timber volume")
par(mfrow=c(1,1))
options(scipen=0)

p2i_rez=polygon2input(vector_data=nogabali,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsQuant_VolumeTotal.tif",
  value_field = "KopejaKraja3",
  fun="max",
  prepare=FALSE,

```

```

        restrict_to = clearcut_mask,
        restrict_values = 0,
        plot_result=TRUE,
        overwrite=TRUE)
p2i_rez
i2e_rez=input2egv(input="./RasterGrids_10m/2024/ForestsQuant_VolumeTotal.tif",
                    evg_template = "./Templates/TemplateRasters/LV100m_10km.tif",
                    summary_function = "sum",
                    missing_job = "CoverOutput",
                    output_bg = "./Templates/TemplateRasters/nulls_LV100m_10km.tif",
                    outlocation = "./RasterGrids_100m/2024/RAW/",
                    outfilename = "ForestsQuant_VolumeTotal-sum_cell.tif",
                    layername = "egv_307",
                    plot_final=TRUE)
i2e_rez
rm(p2i_rez)
rm(nogabali)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsQuant_VolumeTotal.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsQuant_VolumeTotal-sum_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.308 ForestsSoil\_EutrophicDrained\_cell

**filename:** ForestsSoil\_EutrophicDrained\_cell.tif

**layername:** egv\_308

**English name:** Fractional cover of Drained Eutrophic Forests within the analysis cell (1 ha)

**Latvian name:** Susinātu eitrofu mežu platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** To prepare this EGV, forest stands with forest type equal to “19”, “21”, “24” or “25” are selected from the [State Forest Service’s State Forest Registry](#) and rasterised. Rasterisation is performed using the workflow `egvtools::polygon2input()` with background covering (value 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

```

```

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# ForestSoil_EutrophicDrained_cell.tif evg_308 ----
EutrophicDrained=mvr %>%
  filter(mt %in% c("19","21","24","25"))
p2i_rez=egvtools::polygon2input(vector_data = EutrophicDrained,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsSoil_EutrophicDrained_input.tif",
  value_field = "yes",
  prepare=FALSE,
  background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
  plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
  "ForestsSoil_EutrophicDrained_input.tif"),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "ForestsSoil_EutrophicDrained_cell.tif",
  layername = "evg_308",
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(EutrophicDrained)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsSoil_EutrophicDrained_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_EutrophicDrained_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.309 ForestsSoil\_EutrophicDrained\_r500

**filename:** ForestsSoil\_EutrophicDrained\_r500.tif

**layername:** egv\_309

**English name:** Fractional cover of Drained Eutrophic Forests within the 0.5 km landscape

**Latvian name:** Susinātu eitrofu mežu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicDrained_cell.tif"),
  layer_prefixes = c("ForestsSoil_EutrophicDrained"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_EutrophicDrained_r500.tif egv_309
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicDrained_r500.tif")
names(slanis)="egv_309"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicDrained_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_EutrophicDrained_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.310 ForestsSoil\_EutrophicDrained\_r1250

**filename:** ForestsSoil\_EutrophicDrained\_r1250.tif

**layername:** egv\_310

**English name:** Fractional cover of Drained Eutrophic Forests within the 1.25 km landscape

**Latvian name:** Susinātu eitrofu mežu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicDrained_cell.tif"),
  layer_prefixes = c("ForestsSoil_EutrophicDrained"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_EutrophicDrained_r1250.tif    egv_310
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicDrained_r1250.tif")
names(slanis)="egv_310"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicDrained_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_EutrophicDrained_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.311 ForestsSoil\_EutrophicDrained\_r3000

**filename:** ForestsSoil\_EutrophicDrained\_r3000.tif

**layername:** egv\_311

**English name:** Fractional cover of Drained Eutrophic Forests within the 3 km landscape

**Latvian name:** Susinātu eitrofu mežu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicDrained_cell.tif"),
  layer_prefixes = c("ForestsSoil_EutrophicDrained"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_EutrophicDrained_r3000.tif    egv_311
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicDrained_r3000.tif")
names(slanis)="egv_311"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicDrained_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_EutrophicDrained_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.312 ForestsSoil\_EutrophicDrained\_r10000

**filename:** ForestsSoil\_EutrophicDrained\_r10000.tif

**layername:** evg\_312

**English name:** Fractional cover of Drained Eutrophic Forests within the 10 km landscape

**Latvian name:** Susinātu eitrofu mežu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicDrained_cell.tif"),
  layer_prefixes = c("ForestsSoil_EutrophicDrained"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_EutrophicDrained_r10000.tif  evg_312
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicDrained_r10000.tif")
names(slanis)="evg_312"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicDrained_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_EutrophicDrained_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.313 ForestsSoil\_EutrophicMineral\_cell

**filename:** ForestsSoil\_EutrophicMineral\_cell.tif

**layername:** egv\_313

**English name:** Fractional cover of Eutrophic Forests on undrained Mineral Soils within the analysis cell (1 ha)

**Latvian name:** Eitrofu mežu nesusinātās minerālaugsnēs platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** To prepare this EGV, forest stands with forest type equal to “5”, “6”, “10” or “11” are selected from the [State Forest Service’s State Forest Registry](#) and rasterised. Rasterisation is performed using the workflow `egvtools::polygon2input()` with background covering (value 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# ForestsSoil_EutrophicMineral_cell.tif egv_313 ----
EutrophicMineral=mvr %>%
  filter(mt %in% c("5","6","10","11"))
p2i_rez=egvtools::polygon2input(vector_data = EutrophicMineral,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsSoil_EutrophicMineral_input.tif",
  value_field = "yes",
  prepare=FALSE,
  background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
  plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
  "ForestsSoil_EutrophicMineral_input.tif"),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/RAW/",
  outfilename = "ForestsSoil_EutrophicMineral_cell.tif",
  layername = "egv_313",
```

```

        idw_weight = 2,
        plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(EutrophicMineral)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsSoil_EutrophicMineral_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_EutrophicMineral_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.314 ForestsSoil\_EutrophicMineral\_r500

**filename:** ForestsSoil\_EutrophicMineral\_r500.tif

**layername:** egv\_314

**English name:** Fractional cover of Eutrophic Forests on undrained Mineral Soils within the 0.5 km landscape

**Latvian name:** Eitrofu mežu nesusinātās minerālaugsnēs platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/ForestsSoil_EutrophicMineral_cell.tif"),
  layer_prefixes = c("ForestsSoil_EutrophicMineral"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",

```

```

fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# ForestsSoil_EutrophicMineral_r500.tif egv_314
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicMineral_r500.tif")
names(slanis)="egv_314"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicMineral_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_EutrophicMineral_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.315 ForestsSoil\_EutrophicMineral\_r1250

**filename:** ForestsSoil\_EutrophicMineral\_r1250.tif

**layername:** egv\_315

**English name:** Fractional cover of Eutrophic Forests on undrained Mineral Soils within the 1.25 km landscape

**Latvian name:** Eitrofu mežu nesusinātās minerālaugsnēs platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadрати_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicMineral_cell.tif"),
  layer_prefixes = c("ForestsSoil_EutrophicMineral"),

```

```

output_dir  = "./RasterGrids_100m/2024/RAW/",
n_workers   = 6,
radii       = c("r1250"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight  = 2,
future_max_size = 40 * 1024^3)

# ForestsSoil_EutrophicMineral_r1250.tif    egv_315
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicMineral_r1250.tif")
names(slanis)="egv_315"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicMineral_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_EutrophicMineral_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.316 ForestsSoil\_EutrophicMineral\_r3000

**filename:** ForestsSoil\_EutrophicMineral\_r3000.tif

**layername:** egv\_316

**English name:** Fractional cover of Eutrophic Forests on undrained Mineral Soils within the 3 km landscape

**Latvian name:** Eitrofu mežu nesusinātās minerālaugsnēs platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",

```

```

template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers = c("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicMineral_cell.tif"),
layer_prefixes = c("ForestsSoil_EutrophicMineral"),
output_dir = "./RasterGrids_100m/2024/RAW/",
n_workers = 6,
radii = c("r3000"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# ForestsSoil_EutrophicMineral_r3000.tif    egv_316
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicMineral_r3000.tif")
names(slanis)="egv_316"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicMineral_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_EutrophicMineral_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

### 6.317 ForestsSoil\_EutrophicMineral\_r10000

**filename:** ForestsSoil\_EutrophicMineral\_r10000.tif

**layername:** egv\_317

**English name:** Fractional cover of Eutrophic Forests on undrained Mineral Soils within the 10 km landscape

**Latvian name:** Eitrofu mežu nesusinātās minerālaugsnēs platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function()

```

```

kvadrati_path = "./Templates/TemplateGrids/tiles/",
radii_path    = "./Templates/TemplateGridPoints/tiles/",
tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicMineral_cell.tif"),
layer_prefixes = c("ForestsSoil_EutrophicMineral"),
output_dir   = "./RasterGrids_100m/2024/RAW/",
n_workers    = 6,
radii        = c("r10000"),
radius_mode  = "sparse",
extract_fun  = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3)

# ForestsSoil_EutrophicMineral_r10000.tif  egv_317
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicMineral_r10000.tif")
names(slanis)="egv_317"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicMineral_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_EutrophicMineral_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.318 ForestsSoil\_EutrophicOrganic\_cell

**filename:** ForestsSoil\_EutrophicOrganic\_cell.tif

**layername:** egv\_318

**English name:** Fractional cover of Eutrophic Forests on undrained Organic Soils within the analysis cell (1 ha)

**Latvian name:** Eitrofu mežu nesusinātās organiskajās augsnēs platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** To prepare this EGV, forest stands with forest type equal to “15” or “16” are selected from the State Forest Service’s State Forest Registry and rasterised. Rasterisation is performed using the workflow `egvtools::polygon2input()` with background covering (value 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}

```

```

if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# ForestSoil_EutrophicOrganic_cell.tif evg_318 ----
EutrophicOrganic=mvr %>%
  filter(mt %in% c("15","16"))
p2i_rez=egvtools::polygon2input(vector_data = EutrophicOrganic,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsSoil_EutrophicOrganic_input.tif",
  value_field = "yes",
  prepare=FALSE,
  background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
  plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
  "ForestsSoil_EutrophicOrganic_input.tif"),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "ForestsSoil_EutrophicOrganic_cell.tif",
  layername = "evg_318",
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(EutrophicOrganic)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsSoil_EutrophicOrganic_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_EutrophicOrganic_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,

```

```
overwrite=TRUE)
```

## 6.319 ForestsSoil\_EutrophicOrganic\_r500

**filename:** ForestsSoil\_EutrophicOrganic\_r500.tif

**layername:** egv\_319

**English name:** Fractional cover of Eutrophic Forests on undrained Organic Soils within the 0.5 km landscape

**Latvian name:** Eitrofu mežu nesusinātās organiskajās augsnēs platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicOrganic_cell.tif"),
  layer_prefixes = c("ForestsSoil_EutrophicOrganic"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_EutrophicOrganic_r500.tif egv_319
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicOrganic_r500.tif")
names(slanis)="egv_319"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicOrganic_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_EutrophicOrganic_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
```

```

centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.320 ForestsSoil\_EutrophicOrganic\_r1250

**filename:** ForestsSoil\_EutrophicOrganic\_r1250.tif

**layername:** egv\_320

**English name:** Fractional cover of Eutrophic Forests on undrained Organic Soils within the 1.25 km landscape

**Latvian name:** Eitrofu mežu nesusinātās organiskajās augsnēs platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicOrganic_cell.tif"),
  layer_prefixes = c("ForestsSoil_EutrophicOrganic"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_EutrophicOrganic_r1250.tif    egv_320
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicOrganic_r1250.tif")
names(slanis)="egv_320"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicOrganic_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

nosaukums="ForestsSoil_EutrophicOrganic_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.321 ForestsSoil\_EutrophicOrganic\_r3000

**filename:** ForestsSoil\_EutrophicOrganic\_r3000.tif

**layername:** egv\_321

**English name:** Fractional cover of Eutrophic Forests on undrained Organic Soils within the 3 km landscape

**Latvian name:** Eitrofu mežu nesusinātās organiskajās augsnēs platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicOrganic_cell.tif"),
  layer_prefixes = c("ForestsSoil_EutrophicOrganic"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_EutrophicOrganic_r3000.tif    egv_321
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicOrganic_r3000.tif")
names(slanis)="egv_321"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicOrganic_r3000.tif",
  overwrite=TRUE)

# standardisation ----

```

```

if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_EutrophicOrganic_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.322 ForestsSoil\_EutrophicOrganic\_r10000

**filename:** ForestsSoil\_EutrophicOrganic\_r10000.tif

**layername:** egv\_322

**English name:** Fractional cover of Eutrophic Forests on undrained Organic Soils within the 10 km landscape

**Latvian name:** Eitrofu mežu nesusinātās organiskajās augsnēs platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadri_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicOrganic_cell.tif"),
  layer_prefixes = c("ForestsSoil_EutrophicOrganic"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_EutrophicOrganic_r10000.tif  egv_322
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicOrganic_r10000.tif")
names(slanis)="egv_322"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsSoil_EutrophicOrganic_r10000.tif",
            overwrite=TRUE)

```

```

    overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_EutrophicOrganic_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

### 6.323 ForestsSoil\_MesotrophicMineral\_cell

**filename:** ForestsSoil\_MesotrophicMineral\_cell.tif

**layername:** egv\_323

**English name:** Fractional cover of Mesotrophic Forests on undrained Mineral Soils within the analysis cell (1 ha)

**Latvian name:** Mezotrofu mežu nesusinātās minerālaugsnēs platības īpatsvars analīzē šūnā (1 ha)

**Procedure:** To prepare this EGV, forest stands with forest type equal to “4” or “9” are selected from the State Forest Service’s [State Forest Registry](#) and rasterised. Rasterisation is performed using the workflow `egvtools::polygon2input()` with background covering (value 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

```

```

# ForestsSoil_MesotrophicMineral_cell.tif    egv_323 ----
MesotrophicMineral=mvr %>%
  filter(mt %in% c("4","9"))
p2i_rez=egvtools::polygon2input(vector_data = MesotrophicMineral,
                                template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                out_path = "./RasterGrids_10m/2024/",
                                file_name = "ForestsSoil_MesotrophicMineral_input.tif",
                                value_field = "yes",
                                prepare=FALSE,
                                background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "ForestsSoil_MesotrophicMineral_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/Raw/",
                             outfilename = "ForestsSoil_MesotrophicMineral_cell.tif",
                             layername = "egv_323",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(MesotrophicMineral)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsSoil_MesotrophicMineral_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_MesotrophicMineral_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.324 ForestsSoil\_MesotrophicMineral\_r500

**filename:** ForestsSoil\_MesotrophicMineral\_r500.tif

**layername:** egv\_324

**English name:** Fractional cover of Mesotrophic Forests on undrained Mineral Soils within the 0.5 km landscape

**Latvian name:** Mezotrofu mežu nesusinātās minerālaugsnēs platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
```

```

if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/ForestsSoil_MesotrophicMineral_cell.tif"),
  layer_prefixes = c("ForestsSoil_MesotrophicMineral"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_MesotrophicMineral_r500.tif  egv_324
slanis=rast("./RasterGrids_100m/2024/Raw/ForestsSoil_MesotrophicMineral_r500.tif")
names(slanis)="egv_324"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/ForestsSoil_MesotrophicMineral_r500.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_MesotrophicMineral_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.325 ForestsSoil\_MesotrophicMineral\_r1250

**filename:** ForestsSoil\_MesotrophicMineral\_r1250.tif

**layername:** egv\_325

**English name:** Fractional cover of Mesotrophic Forests on undrained Mineral Soils within the 1.25 km landscape

**Latvian name:** Mezotrofu mežu nesusinātās minerālaugsnēs platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name.

Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadradii_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path  = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   = c("./RasterGrids_100m/2024/RAW/ForestsSoil_MesotrophicMineral_cell.tif"),
  layer_prefixes = c("ForestsSoil_MesotrophicMineral"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_MesotrophicMineral_r1250.tif  egv_325
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_MesotrophicMineral_r1250.tif")
names(slanis)="egv_325"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_MesotrophicMineral_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_MesotrophicMineral_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.326 ForestsSoil\_MesotrophicMineral\_r3000

**filename:** ForestsSoil\_MesotrophicMineral\_r3000.tif

**layername:** egv\_326

**English name:** Fractional cover of Mesotrophic Forests on undrained Mineral Soils within the 3 km landscape

**Latvian name:** Mezotrofu mežu nesusinātās minerālaugsnēs platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_MesotrophicMineral_cell.tif"),
  layer_prefixes = c("ForestsSoil_MesotrophicMineral"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_MesotrophicMineral_r3000.tif  egv_326
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_MesotrophicMineral_r3000.tif")
names(slanis)="egv_326"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_MesotrophicMineral_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_MesotrophicMineral_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra:::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.327 ForestsSoil\_MesotrophicMineral\_r10000

**filename:** ForestsSoil\_MesotrophicMineral\_r10000.tif

**layername:** egv\_327

**English name:** Fractional cover of Mesotrophic Forests on undrained Mineral Soils within the 10 km landscape

**Latvian name:** Mezotrofu mežu nesusinātās minerālaugsnēs platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_MesotrophicMineral_cell.tif"),
  layer_prefixes = c("ForestsSoil_MesotrophicMineral"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_MesotrophicMineral_r10000.tif egv_327
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_MesotrophicMineral_r10000.tif")
names(slanis)="egv_327"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_MesotrophicMineral_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_MesotrophicMineral_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.328 ForestsSoil\_OligotrophicDrained\_cell

**filename:** ForestsSoil\_OligotrophicDrained\_cell.tif

**layername:** egv\_328

**English name:** Fractional cover of Drained Oligotrophic Forests within the analysis cell (1 ha)

**Latvian name:** Susinātu oligotrofu mežu platības īpatsvars analīzēs šūnā (1 ha)

**Procedure:** To prepare this EGV, forest stands with forest type equal to “17”, “18”, “22” or “23” are selected from the [State Forest Service’s State Forest Registry](#) and rasterised. Rasterisation is performed using the workflow `egvtools::polygon2input()` with background covering (value 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# ForestsSoil_OligotrophicDrained_cell.tif  egv_328 ----
OligotrophicDrained=mvr %>%
  filter(mt %in% c("17","18","22","23"))
p2i_rez=egvtools::polygon2input(vector_data = OligotrophicDrained,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsSoil_OligotrophicDrained_input.tif",
  value_field = "yes",
  prepare=FALSE,
  background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
  plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
  "ForestsSoil_OligotrophicDrained_input.tif"),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/RAW/",
  outfilename = "ForestsSoil_OligotrophicDrained_cell.tif",
  layername = "egv_328",
  idw_weight = 2,
```

```

    plot_gaps = FALSE, plot_final = TRUE)
i2e_rez
rm(OligotrophicDrained)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsSoil_OligotrophicDrained_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_OligotrophicDrained_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.329 ForestsSoil\_OligotrophicDrained\_r500

**filename:** ForestsSoil\_OligotrophicDrained\_r500.tif

**layername:** egv\_329

**English name:** Fractional cover of Drained Oligotrophic Forests within the 0.5 km landscape

**Latvian name:** Susinātu oligotrofu mežu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/ForestsSoil_OligotrophicDrained_cell.tif"),
  layer_prefixes = c("ForestsSoil_OligotrophicDrained"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

```

```

# ForestsSoil_OligotrophicDrained_r500.tif  evg_329
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicDrained_r500.tif")
names(slanis)="evg_329"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicDrained_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_OligotrophicDrained_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.330 ForestsSoil\_OligotrophicDrained\_r1250

**filename:** ForestsSoil\_OligotrophicDrained\_r1250.tif

**layername:** evg\_330

**English name:** Fractional cover of Drained Oligotrophic Forests within the 1.25 km landscape

**Latvian name:** Susinātu oligotrofu mežu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadri_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicDrained_cell.tif"),
  layer_prefixes = c("ForestsSoil_OligotrophicDrained"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",

```

```

fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# ForestsSoil_OligotrophicDrained_r1250.tif egv_330
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicDrained_r1250.tif")
names(slanis)="egv_330"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicDrained_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_OligotrophicDrained_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.331 ForestsSoil\_OligotrophicDrained\_r3000

**filename:** ForestsSoil\_OligotrophicDrained\_r3000.tif

**layername:** egv\_331

**English name:** Fractional cover of Drained Oligotrophic Forests within the 3 km landscape

**Latvian name:** Susinātu oligotrofu mežu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicDrained_cell.tif"),
  layer_prefixes = c("ForestsSoil_OligotrophicDrained"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,

```

```

radii      = c("r3000"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# ForestsSoil_OligotrophicDrained_r3000.tif egv_331
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicDrained_r3000.tif")
names(slanis)="egv_331"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicDrained_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_OligotrophicDrained_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.332 ForestsSoil\_OligotrophicDrained\_r10000

**filename:** ForestsSoil\_OligotrophicDrained\_r10000.tif

**layername:** egv\_332

**English name:** Fractional cover of Drained Oligotrophic Forests within the 10 km landscape

**Latvian name:** Susinātu oligotrofu mežu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicDrained_cell.tif"),

```

```

layer_prefixes = c("ForestsSoil_OligotrophicDrained"),
output_dir    = "./RasterGrids_100m/2024/RAW/",
n_workers     = 6,
radii         = c("r10000"),
radius_mode   = "sparse",
extract_fun   = "mean",
fill_missing   = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3)

# ForestsSoil_OligotrophicDrained_r10000.tif    evg_332
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicDrained_r10000.tif")
names(slanis)="evg_332"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicDrained_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_OligotrophicDrained_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

### 6.333 ForestsSoil\_OligotrophicMineral\_cell

**filename:** ForestsSoil\_OligotrophicMineral\_cell.tif

**layername:** evg\_333

**English name:** Fractional cover of Oligotrophic Forests on undrained Mineral Soils within the analysis cell (1 ha)

**Latvian name:** Oligotrofu mežu nesusinātās minerālaugsnēs platības īpatsvars analīzē šūnā (1 ha)

**Procedure:** To prepare this EGV, forest stands with forest type equal to “1”, “2”, “3”, “7” or “8” are selected from the [State Forest Service’s State Forest Registry](#) and rasterised. Rasterisation is performed using the workflow `egvtools::polygon2input()` with background covering (value 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

```

```

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# ForestsSoil_OligotrophicMineral_cell.tif  evg_333 ----
OligotrophicMineral=mvr %>%
  filter(mt %in% c("1","2","3","7","8"))
p2i_rez=egvtools::polygon2input(vector_data = OligotrophicMineral,
                                 template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                 out_path = "./RasterGrids_10m/2024/",
                                 file_name = "ForestsSoil_OligotrophicMineral_input.tif",
                                 value_field = "yes",
                                 prepare=FALSE,
                                 background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                 plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "ForestsSoil_OligotrophicMineral_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/Raw/",
                             outfilename = "ForestsSoil_OligotrophicMineral_cell.tif",
                             layername = "evg_333",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(OligotrophicMineral)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsSoil_OligotrophicMineral_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_OligotrophicMineral_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.334 ForestsSoil\_OligotrophicMineral\_r500

**filename:** ForestsSoil\_OligotrophicMineral\_r500.tif

**layername:** egv\_334

**English name:** Fractional cover of Oligotrophic Forests on undrained Mineral Soils within the 0.5 km landscape

**Latvian name:** Oligotrofu mežu nesusinātās minerālaugsnēs platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicMineral_cell.tif"),
  layer_prefixes = c("ForestsSoil_OligotrophicMineral"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_OligotrophicMineral_r500.tif  egv_334
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicMineral_r500.tif")
names(slanis)="egv_334"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicMineral_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_OligotrophicMineral_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
```

```
overwrite=TRUE)
```

## 6.335 ForestsSoil\_OligotrophicMineral\_r1250

**filename:** ForestsSoil\_OligotrophicMineral\_r1250.tif

**layername:** egv\_335

**English name:** Fractional cover of Oligotrophic Forests on undrained Mineral Soils within the 1.25 km landscape

**Latvian name:** Oligotrofu mežu nesusinātās minerālaugsnēs platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicMineral_cell.tif"),
  layer_prefixes = c("ForestsSoil_OligotrophicMineral"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_OligotrophicMineral_r1250.tif egv_335
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicMineral_r1250.tif")
names(slanis)="egv_335"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicMineral_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_OligotrophicMineral_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
```

```

centrets=slanis$videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.336 ForestsSoil\_OligotrophicMineral\_r3000

**filename:** ForestsSoil\_OligotrophicMineral\_r3000.tif

**layername:** evg\_336

**English name:** Fractional cover of Oligotrophic Forests on undrained Mineral Soils within the 3 km landscape

**Latvian name:** Oligotrofu mežu nesusinātās minerālaugsnēs platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicMineral_cell.tif"),
  layer_prefixes = c("ForestsSoil_OligotrophicMineral"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_OligotrophicMineral_r3000.tif evg_336
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicMineral_r3000.tif")
names(slanis)="evg_336"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicMineral_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

nosaukums="ForestsSoil_OligotrophicMineral_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.337 ForestsSoil\_OligotrophicMineral\_r10000

**filename:** ForestsSoil\_OligotrophicMineral\_r10000.tif

**layername:** egv\_337

**English name:** Fractional cover of Oligotrophic Forests on undrained Mineral Soils within the 10 km landscape

**Latvian name:** Oligotrofu mežu nesusinātās minerālaugsnēs platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicMineral_cell.tif"),
  layer_prefixes = c("ForestsSoil_OligotrophicMineral"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_OligotrophicMineral_r10000.tif      egv_337
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicMineral_r10000.tif")
names(slanis)="egv_337"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicMineral_r10000.tif",
  overwrite=TRUE)

```

```

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_OligotrophicMineral_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.338 ForestsSoil\_OligotrophicOrganic\_cell

**filename:** ForestsSoil\_OligotrophicOrganic\_cell.tif

**layername:** egv\_338

**English name:** Fractional cover of Oligotrophic Forests on undrained Organic Soils within the analysis cell (1 ha)

**Latvian name:** Oligotrofu mežu nesusinātās organiskajās augsnēs platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** To prepare this EGV, forest stands with forest type equal to “12” or “14” are selected from the State Forest Service’s State Forest Registry and rasterised. Rasterisation is performed using the workflow `egvtools::polygon2input()` with background covering (value 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

```

```

# ForestsSoil_OligotrophicOrganic_cell.tif  egv_338 ----
OligotrophicOrganic=mvr %>%
  filter(mt %in% c("12","14"))
p2i_rez=egvtools::polygon2input(vector_data = OligotrophicOrganic,
                                template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                out_path = "./RasterGrids_10m/2024/",
                                file_name = "ForestsSoil_OligotrophicOrganic_input.tif",
                                value_field = "yes",
                                prepare=FALSE,
                                background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "ForestsSoil_OligotrophicOrganic_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/Raw/",
                             outfilename = "ForestsSoil_OligotrophicOrganic_cell.tif",
                             layername = "egv_338",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(OligotrophicOrganic)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsSoil_OligotrophicOrganic_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_OligotrophicMineral_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.339 ForestsSoil\_OligotrophicOrganic\_r500

**filename:** ForestsSoil\_OligotrophicOrganic\_r500.tif

**layername:** egv\_339

**English name:** Fractional cover of Oligotrophic Forests on undrained Organic Soils within the 0.5 km landscape

**Latvian name:** Oligotrofu mežu nesusinātās organiskajās augsnēs platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
```

```

if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicOrganic_cell.tif"),
  layer_prefixes = c("ForestsSoil_OligotrophicOrganic"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_OligotrophicOrganic_r500.tif egv_339
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicOrganic_r500.tif")
names(slanis)="egv_339"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicOrganic_r500.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_OligotrophicOrganic_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.340 ForestsSoil\_OligotrophicOrganic\_r1250

**filename:** ForestsSoil\_OligotrophicOrganic\_r1250.tif

**layername:** egv\_340

**English name:** Fractional cover of Oligotrophic Forests on undrained Organic Soils within the 1.25 km landscape

**Latvian name:** Oligotrofu mežu nesusinātās organiskajās augsnēs platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/ForestsSoil_OligotrophicOrganic_cell.tif"),
  layer_prefixes = c("ForestsSoil_OligotrophicOrganic"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_OligotrophicOrganic_r1250.tif egv_340
slanis=rast("./RasterGrids_100m/2024/Raw/ForestsSoil_OligotrophicOrganic_r1250.tif")
names(slanis)="egv_340"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/ForestsSoil_OligotrophicOrganic_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_OligotrophicOrganic_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.341 ForestsSoil\_OligotrophicOrganic\_r3000

**filename:** ForestsSoil\_OligotrophicOrganic\_r3000.tif

**layername:** egv\_341

**English name:** Fractional cover of Oligotrophic Forests on undrained Organic Soils within the 3 km landscape

**Latvian name:** Oligotrofu mežu nesusinātās organiskajās augsnēs platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`.

During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicOrganic_cell.tif"),
  layer_prefixes = c("ForestsSoil_OligotrophicOrganic"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_OligotrophicOrganic_r3000.tif egv_341
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicOrganic_r3000.tif")
names(slanis)="egv_341"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicOrganic_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_OligotrophicOrganic_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.342 ForestsSoil\_OligotrophicOrganic\_r10000

**filename:** ForestsSoil\_OligotrophicOrganic\_r10000.tif

**layername:** egv\_342

**English name:** Fractional cover of Oligotrophic Forests on undrained Organic Soils within the 10 km landscape

**Latvian name:** Oligotrofu mežu nesusinātās organiskajās augsnēs platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicOrganic_cell.tif"),
  layer_prefixes = c("ForestsSoil_OligotrophicOrganic"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsSoil_OligotrophicOrganic_r10000.tif      egv_342
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicOrganic_r10000.tif")
names(slanis)="egv_342"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsSoil_OligotrophicOrganic_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsSoil_OligotrophicOrganic_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.343 ForestsTreesAge\_BorealDeciduousOld\_cell

**filename:** ForestsTreesAge\_BorealDeciduousOld\_cell.tif

**layername:** evg\_343

**English name:** Fractional cover of Old (over rotation age) Boreal Deciduous Forests within the analysis cell (1 ha)

**Latvian name:** Vecu (kopš cirtmeta) šaurlapju mežu platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

To prepare this EGV, stands from the [State Forest Service's State Forest Registry](#) are classified into (in order):

- coniferous (see [Terminology and acronyms](#) for species codes) if timber volume of those species exceeded 75%;
- Boreal deciduous if timber volume of those species exceeded 75%;
- temperate deciduous if timber volume of those species exceeded 50%;
- mixed otherwise;

then Boreal deciduous stands exceeding the legal rotation age are selected and geometries are rasterised (presence = 1, NA otherwise). Rasterisation is performed using the workflow `egvtools::polygon2input()`, restricting to pixels outside clearcut mask and covering background with value 0. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)
```

```

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
    filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
    overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsTreesAge_BorealDeciduousOld_cell.tif  egv_343 ----
skujkoki=c("1","3","13","14","15","22","23","28") # 8
saurlapji=c("4","6","8","9","19","20","21","32","35","68") # 10
platlapji=c("10","11","12","16","17","18","24","25","26","27","28","29","50",
    "61","62","63","64","65","66","67","69") # 21
mvr=mvr %>%
  mutate(kraja_skujkoku=ifelse(s10 %in% skujkoki,v10,0) +
    ifelse(s11 %in% skujkoki,v11,0)+ifelse(s12 %in% skujkoki,v12,0) +
    ifelse(s13 %in% skujkoki,v13,0)+ifelse(s14 %in% skujkoki,v14,0),
    kraja_saurlapju=ifelse(s10 %in% saurlapji,v10,0) +
    ifelse(s11 %in% saurlapji,v11,0)+ifelse(s12 %in% saurlapji,v12,0) +
    ifelse(s13 %in% saurlapji,v13,0)+ifelse(s14 %in% saurlapji,v14,0),
    kraja_platlapju=ifelse(s10 %in% platlapji,v10,0) +
    ifelse(s11 %in% platlapji,v11,0)+ifelse(s12 %in% platlapji,v12,0) +
    ifelse(s13 %in% platlapji,v13,0)+ifelse(s14 %in% platlapji,v14,0)) %>%
  mutate(kopeja_kraja=kraja_skujkoku+kraja_platlapju+kraja_saurlapju) %>%
  mutate(tips=ifelse(kraja_skujkoku/kopeja_kraja>=0.75,"skujkoku",
    ifelse(kraja_saurlapju/kopeja_kraja>=0.75,"saurlapju",
      ifelse(kraja_platlapju/kopeja_kraja>0.5,"platlapju",
        "jauktu koku"))))

nogabali=mvr %>%
  filter(zkat=="10"&tips=="saurlapju"&(vgr=="4" | vgr=="5"))

p2i_rez=egvtools::polygon2input(vector_data = nogabali,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsTreesAge_BorealDeciduousOld_input.tif",
  value_field = "yes",
  restrict_to = clearcut_mask,
  restrict_values = 0,
  prepare=FALSE,
  background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
  plot_result = TRUE)

p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
  "ForestsTreesAge_BorealDeciduousOld_input.tif"),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "ForestsTreesAge_BorealDeciduousOld_cell.tif",
  layername = "egv_343",
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(nogabali)
rm(p2i_rez)
rm(i2e_rez)

```

```

unlink("./RasterGrids_10m/2024/ForestsTreesAge_BorealDeciduousOld_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_BorealDeciduousOld_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw//",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled//",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.344 ForestsTreesAge\_BorealDeciduousOld\_r500

**filename:** ForestsTreesAge\_BorealDeciduousOld\_r500.tif

**layername:** evg\_344

**English name:** Fractional cover of Old (over rotation age) Boreal Deciduous Forests within the 0.5 km landscape

**Latvian name:** Vecu (kopš cirtmeta) šaurlapju mežu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/Raw/ForestsTreesAge_BorealDeciduousOld_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_BorealDeciduousOld"),
  output_dir    = "./RasterGrids_100m/2024/Raw//",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

```

```

# ForestsTreesAge_BorealDeciduousOld_r500.tif    egv_344
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_BorealDeciduousOld_r500.tif")
names(slanis)="egv_344"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTreesAge_BorealDeciduousOld_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_BorealDeciduousOld_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.345 ForestsTreesAge\_BorealDeciduousOld\_r1250

**filename:** ForestsTreesAge\_BorealDeciduousOld\_r1250.tif

**layername:** egv\_345

**English name:** Fractional cover of Old (over rotation age) Boreal Deciduous Forests within the 1.25 km landscape

**Latvian name:** Vecu (kopš cirtmeta) šaurlapju mežu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_BorealDeciduousOld_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_BorealDeciduousOld"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r1250"),
  radius_mode   = "sparse",

```

```

extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# ForestsTreesAge_BorealDeciduousOld_r1250.tif egv_345
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_BorealDeciduousOld_r1250.tif")
names(slanis)="egv_345"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTreesAge_BorealDeciduousOld_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_BorealDeciduousOld_r1250.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.346 ForestsTreesAge\_BorealDeciduousOld\_r3000

**filename:** ForestsTreesAge\_BorealDeciduousOld\_r3000.tif

**layername:** egv\_346

**English name:** Fractional cover of Old (over rotation age) Boreal Deciduous Forests within the 3 km landscape

**Latvian name:** Vecu (kopš cirtmeta) šaurlapju mežu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path  = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   =
    c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_BorealDeciduousOld_cell.tif"),

```

```

layer_prefixes = c("ForestsTreesAge_BorealDeciduousOld"),
output_dir    = "./RasterGrids_100m/2024/RAW/",
n_workers     = 6,
radii         = c("r3000"),
radius_mode   = "sparse",
extract_fun   = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3)

# ForestsTreesAge_BorealDeciduousOld_r3000.tif egv_346
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_BorealDeciduousOld_r3000.tif")
names(slanis)="egv_346"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTreesAge_BorealDeciduousOld_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_BorealDeciduousOld_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.347 ForestsTreesAge\_BorealDeciduousOld\_r10000

**filename:** ForestsTreesAge\_BorealDeciduousOld\_r10000.tif

**layername:** egv\_347

**English name:** Fractional cover of Old (over rotation age) Boreal Deciduous Forests within the 10 km landscape

**Latvian name:** Vecu (kopš cirtmeta) šaurlapju mežu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(

```

```

kvadrati_path = "./Templates/TemplateGrids/tiles/",
radii_path    = "./Templates/TemplateGridPoints/tiles/",
tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers =
  ↳ c("./RasterGrids_100m/2024/Raw/ForestsTreesAge_BorealDeciduousOld_cell.tif"),
layer_prefixes = c("ForestsTreesAge_BorealDeciduousOld"),
output_dir    = "./RasterGrids_100m/2024/Raw/",
n_workers     = 6,
radii         = c("r10000"),
radius_mode   = "sparse",
extract_fun   = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3)

# ForestsTreesAge_BorealDeciduousOld_r10000.tif egv_347
slanis=rast("./RasterGrids_100m/2024/Raw/ForestsTreesAge_BorealDeciduousOld_r10000.tif")
names(slanis)="egv_347"
slanis2=project(slanis,template100)
writeRaster(slanis2,
           "./RasterGrids_100m/2024/Raw/ForestsTreesAge_BorealDeciduousOld_r10000.tif",
           overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_BorealDeciduousOld_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.348 ForestsTreesAge\_BorealDeciduousYoung\_cell

**filename:** ForestsTreesAge\_BorealDeciduousYoung\_cell.tif

**layername:** egv\_348

**English name:** Fractional cover of Young (pre-rotation age) Boreal Deciduous Forests within the analysis cell (1 ha)

**Latvian name:** Jaunu (pirms cirtmeta) šaurlapju mežu platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

To prepare this EGV, stands from the [State Forest Service's State Forest Registry](#) are classified into (in order):

- coniferous (see [Terminology and acronyms](#) for species codes) if timber volume of those species exceeded 75%;
- Boreal deciduous if timber volume of those species exceeded 75%;

- temperate deciduous if timber volume of those species exceeded 50%;
- mixed otherwise;

then Boreal deciduous stands younger than the legal rotation age are selected and geometries are rasterised (presence = 1, NA otherwise). Rasterisation is performed using the workflow `egvtools::polygon2input()`, restricting to pixels outside clearcut mask and covering background with value 0. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsTreesAge_BorealDeciduousYoung_cell.tif egv_348 ----
skujkoki=c("1","3","13","14","15","22","23","28") # 8
```

```

saurlapji=c("4","6","8","9","19","20","21","32","35","68") # 10
platlapji=c("10","11","12","16","17","18","24","25","26","27","28","29","50",
           "61","62","63","64","65","66","67","69") # 21
mvr=mvr %>%
  mutate(kraja_skujkoku=ifelse(s10 %in% skujkoki,v10,0)+
         ifelse(s11 %in% skujkoki,v11,0)+ifelse(s12 %in% skujkoki,v12,0)+
         ifelse(s13 %in% skujkoki,v13,0)+ifelse(s14 %in% skujkoki,v14,0),
         kraja_saurlapju=ifelse(s10 %in% saurlapji,v10,0)+
         ifelse(s11 %in% saurlapji,v11,0)+ifelse(s12 %in% saurlapji,v12,0)+
         ifelse(s13 %in% saurlapji,v13,0)+ifelse(s14 %in% saurlapji,v14,0),
         kraja_platlapju=ifelse(s10 %in% platlapji,v10,0)+
         ifelse(s11 %in% platlapji,v11,0)+ifelse(s12 %in% platlapji,v12,0)+
         ifelse(s13 %in% platlapji,v13,0)+ifelse(s14 %in% platlapji,v14,0)) %>%
  mutate(kopeja_kraja=kraja_skujkoku+kraja_platlapju+kraja_saurlapju) %>%
  mutate(tips=ifelse(kraja_skujkoku/kopeja_kraja>=0.75,"skujkoku",
                     ifelse(kraja_saurlapju/kopeja_kraja>=0.75,"saurlapju",
                            ifelse(kraja_platlapju/kopeja_kraja>0.5,"platlapju",
                                   "jauktu koku"))))

nogabali=mvr %>%
  filter(zkat=="10"&tips=="saurlapju"&(vgr=="1" | vgr=="2" | vgr=="3"))

p2i_rez=egvtools::polygon2input(vector_data = nogabali,
                                 template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                 out_path = "./RasterGrids_10m/2024/",
                                 file_name = "ForestsTreesAge_BorealDeciduousYoung_input.tif",
                                 value_field = "yes",
                                 restrict_to = clearcut_mask,
                                 restrict_values = 0,
                                 prepare=FALSE,
                                 background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                 plot_result = TRUE)

p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "ForestsTreesAge_BorealDeciduousYoung_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/Raw/",
                             outfilename = "ForestsTreesAge_BorealDeciduousYoung_cell.tif",
                             layername = "egv_348",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(nogabali)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsTreesAge_BorealDeciduousYoung_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_BorealDeciduousYoung_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.349 ForestsTreesAge\_BorealDeciduousYoung\_r500

**filename:** ForestsTreesAge\_BorealDeciduousYoung\_r500.tif

**layername:** egv\_349

**English name:** Fractional cover of Young (pre-rotation age) Boreal Deciduous Forests within the 0.5 km landscape

**Latvian name:** Jaunu (pirms cirtmeta) šaurlapju mežu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_BorealDeciduousYoung_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_BorealDeciduousYoung"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_BorealDeciduousYoung_r500.tif egv_349
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_BorealDeciduousYoung_r500.tif")
names(slanis)="egv_349"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsTreesAge_BorealDeciduousYoung_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_BorealDeciduousYoung_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
```

```

filename=saglabasanas_cels,
overwrite=TRUE)

```

## 6.350 ForestsTreesAge\_BorealDeciduousYoung\_r1250

**filename:** ForestsTreesAge\_BorealDeciduousYoung\_r1250.tif

**layername:** egv\_350

**English name:** Fractional cover of Young (pre-rotation age) Boreal Deciduous Forests within the 1.25 km landscape

**Latvian name:** Jaunu (pirms cirtmeta) šaurlapju mežu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path   = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers    =
    c("./RasterGrids_100m/2024/Raw/ForestsTreesAge_BorealDeciduousYoung_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_BorealDeciduousYoung"),
  output_dir     = "./RasterGrids_100m/2024/Raw/",
  n_workers      = 6,
  radii          = c("r1250"),
  radius_mode    = "sparse",
  extract_fun    = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_BorealDeciduousYoung_r1250.tif    egv_350
slanis=rast("./RasterGrids_100m/2024/Raw/ForestsTreesAge_BorealDeciduousYoung_r1250.tif")
names(slanis)="egv_350"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/ForestsTreesAge_BorealDeciduousYoung_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_BorealDeciduousYoung_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)

```

```

slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.351 ForestsTreesAge\_BorealDeciduousYoung\_r3000

**filename:** ForestsTreesAge\_BorealDeciduousYoung\_r3000.tif

**layername:** egv\_351

**English name:** Fractional cover of Young (pre-rotation age) Boreal Deciduous Forests within the 3 km landscape

**Latvian name:** Jaunu (pirms cirtmeta) šaurlapju mežu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    ↳ c("./RasterGrids_100m/2024/Raw/ForestsTreesAge_BorealDeciduousYoung_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_BorealDeciduousYoung"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii        = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing  = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_BorealDeciduousYoung_r3000.tif      egv_351
slanis=rast("./RasterGrids_100m/2024/Raw/ForestsTreesAge_BorealDeciduousYoung_r3000.tif")
names(slanis)="egv_351"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/ForestsTreesAge_BorealDeciduousYoung_r3000.tif",
  overwrite=TRUE)

# standardisation ----

```

```

if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_BorealDeciduousYoung_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.352 ForestsTreesAge\_BorealDeciduousYoung\_r10000

**filename:** ForestsTreesAge\_BorealDeciduousYoung\_r10000.tif

**layername:** evg\_352

**English name:** Fractional cover of Young (pre-rotation age) Boreal Deciduous Forests within the 10 km landscape

**Latvian name:** Jaunu (pirms cirtmeta) šaurlapju mežu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers =
    c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_BorealDeciduousYoung_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_BorealDeciduousYoung"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r10000"),
  radius_mode  = "sparse",
  extract_fun  = "mean",
  fill_missing  = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_BorealDeciduousYoung_r10000.tif  evg_352
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_BorealDeciduousYoung_r10000.tif")
names(slanis)="evg_352"

```

```

slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTreesAge_BorealDeciduousYoung_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_BorealDeciduousYoung_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.353 ForestsTreesAge\_ConiferousOld\_cell

**filename:** ForestsTreesAge\_ConiferousOld\_cell.tif

**layername:** egv\_353

**English name:** Fractional cover of Old (over rotation age) Coniferous Forests within the analysis cell (1 ha)

**Latvian name:** Vecu (kopš cirtmeta) skujkoku mežu platības īpatsvars analīzēs šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

To prepare this EGV, stands from the [State Forest Service's State Forest Registry](#) are classified into (in order):

- coniferous (see [Terminology and acronyms](#) for species codes) if timber volume of those species exceeded 75%;
- Boreal deciduous if timber volume of those species exceeded 75%;
- temperate deciduous if timber volume of those species exceeded 50%;
- mixed otherwise;

then coniferous stands exceeding the legal rotation age are selected and geometries are rasterised (presence = 1, NA otherwise). Rasterisation is performed using the workflow `egvtools::polygon2input()`, restricting to pixels outside clearcut mask and covering background with value 0. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}

```

```

if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsTreesAge_ConiferousOld_cell.tif    evg_353 ----
skujkoki=c("1","3","13","14","15","22","23","28") # 8
saurlapji=c("4","6","8","9","19","20","21","32","35","68") # 10
platlapji=c("10","11","12","16","17","18","24","25","26","27","28","29","50",
  "61","62","63","64","65","66","67","69") # 21
mvr=mvr %>%
  mutate(kraja_skujkoku=ifelse(s10 %in% skujkoki,v10,0) +
    ifelse(s11 %in% skujkoki,v11,0)+ifelse(s12 %in% skujkoki,v12,0) +
    ifelse(s13 %in% skujkoki,v13,0)+ifelse(s14 %in% skujkoki,v14,0),
    kraja_saurlapju=ifelse(s10 %in% saurlapji,v10,0) +
    ifelse(s11 %in% saurlapji,v11,0)+ifelse(s12 %in% saurlapji,v12,0) +
    ifelse(s13 %in% saurlapji,v13,0)+ifelse(s14 %in% saurlapji,v14,0),
    kraja_platlapju=ifelse(s10 %in% platlapji,v10,0) +
    ifelse(s11 %in% platlapji,v11,0)+ifelse(s12 %in% platlapji,v12,0) +
    ifelse(s13 %in% platlapji,v13,0)+ifelse(s14 %in% platlapji,v14,0)) %>%
  mutate(kopeja_kraja=kraja_skujkoku+kraja_platlapju+kraja_saurlapju) %>%
  mutate(tips=ifelse(kraja_skujkoku/kopeja_kraja>=0.75,"skujkoku",
    ifelse(kraja_saurlapju/kopeja_kraja>=0.75,"saurlapju",
      ifelse(kraja_platlapju/kopeja_kraja>0.5,"platlapju",
        "jauktu koku"))))

nogabali=mvr %>%

```

```

filter(zkat=="10"&tips=="skujkoku"&(vgr=="4" | vgr=="5"))

p2i_rez=egvtools::polygon2input(vector_data = nogabali,
                                template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                out_path = "./RasterGrids_10m/2024/",
                                file_name = "ForestsTreesAge_ConiferousOld_input.tif",
                                value_field = "yes",
                                restrict_to = clearcut_mask,
                                restrict_values = 0,
                                prepare=FALSE,
                                background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                plot_result = TRUE)

p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "ForestsTreesAge_ConiferousOld_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/Raw/",
                             outfilename = "ForestsTreesAge_ConiferousOld_cell.tif",
                             layername = "egv_353",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(nogabali)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsTreesAge_ConiferousOld_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_ConiferousOld_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.354 ForestsTreesAge\_ConiferousOld\_r500

**filename:** ForestsTreesAge\_ConiferousOld\_r500.tif

**layername:** egv\_354

**English name:** Fractional cover of Old (over rotation age) Coniferous Forests within the 0.5 km landscape

**Latvian name:** Vecu (kopš cirtmeta) skujkoku mežu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
```

```

if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousOld_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_ConiferousOld"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_ConiferousOld_r500.tif      egv_354
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousOld_r500.tif")
names(slanis)="egv_354"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousOld_r500.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_ConiferousOld_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.355 ForestsTreesAge\_ConiferousOld\_r1250

**filename:** ForestsTreesAge\_ConiferousOld\_r1250.tif

**layername:** egv\_355

**English name:** Fractional cover of Old (over rotation age) Coniferous Forests within the 1.25 km landscape

**Latvian name:** Vecu (kopš cirtmeta) skujkoku mežu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/ForestsTreesAge_ConiferousOld_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_ConiferousOld"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_ConiferousOld_r1250.tif  egv_355
slanis=rast("./RasterGrids_100m/2024/Raw/ForestsTreesAge_ConiferousOld_r1250.tif")
names(slanis)="egv_355"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/ForestsTreesAge_ConiferousOld_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_ConiferousOld_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.356 ForestsTreesAge\_ConiferousOld\_r3000

**filename:** ForestsTreesAge\_ConiferousOld\_r3000.tif

**layername:** egv\_356

**English name:** Fractional cover of Old (over rotation age) Coniferous Forests within the 3 km landscape

**Latvian name:** Vecu (kopš cirtmeta) skujkoku mežu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the

output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/ForestsTreesAge_ConiferousOld_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_ConiferousOld"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii        = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_ConiferousOld_r3000.tif  egv_356
slanis=rast("./RasterGrids_100m/2024/Raw/ForestsTreesAge_ConiferousOld_r3000.tif")
names(slanis)="egv_356"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/ForestsTreesAge_ConiferousOld_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_ConiferousOld_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.357 ForestsTreesAge\_ConiferousOld\_r10000

**filename:** ForestsTreesAge\_ConiferousOld\_r10000.tif

**layername:** egv\_357

**English name:** Fractional cover of Old (over rotation age) Coniferous Forests within the 10 km landscape

**Latvian name:** Vecu (kopš cirtmeta) skujkoku mežu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the `analysis cells` inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousOld_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_ConiferousOld"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_ConiferousOld_r10000.tif  egv_357
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousOld_r10000.tif")
names(slanis)="egv_357"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousOld_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_ConiferousOld_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.358 ForestsTreesAge\_ConiferousYoung\_cell

**filename:** ForestsTreesAge\_ConiferousYoung\_cell.tif

**layername:** egv\_358

**English name:** Fractional cover of Young (pre-rotation age) Coniferous Forests within the analysis cell (1 ha)

**Latvian name:** Jaunu (pirms cirtmeta) skujkoku mežu platības īpatsvars analīzēs šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

To prepare this EGV, stands from the [State Forest Service's State Forest Registry](#) are classified into (in order):

- coniferous (see [Terminology and acronyms](#) for species codes) if timber volume of those species exceeded 75%;
- Boreal deciduous if timber volume of those species exceeded 75%;
- temperate deciduous if timber volume of those species exceeded 50%;
- mixed otherwise;

then coniferous stands younger than the legal rotation age are selected and geometries are rasterised (presence = 1, NA otherwise). Rasterisation is performed using the workflow `egvtools::polygon2input()`, restricting to pixels outside clearcut mask and covering background with value 0. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
```

```

tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
    filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
    overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsTreesAge_ConiferousYoung_cell.tif  egv_358 ----
skujkoki=c("1","3","13","14","15","22","23","28") # 8
saurlapji=c("4","6","8","9","19","20","21","32","35","68") # 10
platlapji=c("10","11","12","16","17","18","24","25","26","27","28","29","50",
    "61","62","63","64","65","66","67","69") # 21
mvr=mvr %>%
  mutate(kraja_skujkoku=ifelse(s10 %in% skujkoki,v10,0) +
    ifelse(s11 %in% skujkoki,v11,0)+ifelse(s12 %in% skujkoki,v12,0) +
    ifelse(s13 %in% skujkoki,v13,0)+ifelse(s14 %in% skujkoki,v14,0),
    kraja_saurlapju=ifelse(s10 %in% saurlapji,v10,0) +
    ifelse(s11 %in% saurlapji,v11,0)+ifelse(s12 %in% saurlapji,v12,0) +
    ifelse(s13 %in% saurlapji,v13,0)+ifelse(s14 %in% saurlapji,v14,0),
    kraja_platlapju=ifelse(s10 %in% platlapji,v10,0) +
    ifelse(s11 %in% platlapji,v11,0)+ifelse(s12 %in% platlapji,v12,0) +
    ifelse(s13 %in% platlapji,v13,0)+ifelse(s14 %in% platlapji,v14,0)) %>%
  mutate(kopeja_kraja=kraja_skujkoku+kraja_platlapju+kraja_saurlapju) %>%
  mutate(tips=ifelse(kraja_skujkoku/kopeja_kraja>=0.75,"skujkoku",
    ifelse(kraja_saurlapju/kopeja_kraja>=0.75,"saurlapju",
    ifelse(kraja_platlapju/kopeja_kraja>0.5,"platlapju",
    "jauktu koku"))))

nogabali=mvr %>%
  filter(zkat=="10"&tips=="skujkoku"&(vgr=="1"|vgr=="2"|vgr=="3"))

p2i_rez=egvtools::polygon2input(vector_data = nogabali,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsTreesAge_ConiferousYoung_input.tif",
  value_field = "yes",
  restrict_to = clearcut_mask,
  restrict_values = 0,
  prepare=FALSE,
  background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
  plot_result = TRUE)

p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
  "ForestsTreesAge_ConiferousYoung_input.tif"),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "ForestsTreesAge_ConiferousYoung_cell.tif",
  layername = "egv_358",
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(nogabali)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsTreesAge_ConiferousYoung_input.tif")

```

```

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_ConiferousYoung_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.359 ForestsTreesAge\_ConiferousYoung\_r500

**filename:** ForestsTreesAge\_ConiferousYoung\_r500.tif

**layername:** egv\_359

**English name:** Fractional cover of Young (pre-rotation age) Coniferous Forests within the 0.5 km landscape

**Latvian name:** Jaunu (pirms cirtmeta) skujkoku mežu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousYoung_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_ConiferousYoung"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_ConiferousYoung_r500.tif egv_359
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousYoung_r500.tif")
names(slanis)="egv_359"
slanis2=project(slanis,template100)
writeRaster(slanis2,

```

```

"./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousYoung_r500.tif",
overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_ConiferousYoung_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.360 ForestsTreesAge\_ConiferousYoung\_r1250

**filename:** ForestsTreesAge\_ConiferousYoung\_r1250.tif

**layername:** egv\_360

**English name:** Fractional cover of Young (pre-rotation age) Coniferous Forests within the 1.25 km landscape

**Latvian name:** Jaunu (pirms cirtmeta) skujkoku mežu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousYoung_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_ConiferousYoung"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

```

```

# ForestsTreesAge_ConiferousYoung_r1250.tif evg_360
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousYoung_r1250.tif")
names(slanis)="evg_360"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousYoung_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_ConiferousYoung_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.361 ForestsTreesAge\_ConiferousYoung\_r3000

**filename:** ForestsTreesAge\_ConiferousYoung\_r3000.tif

**layername:** evg\_361

**English name:** Fractional cover of Young (pre-rotation age) Coniferous Forests within the 3 km landscape

**Latvian name:** Jaunu (pirms cirtmeta) skujkoku mežu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadратi_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousYoung_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_ConiferousYoung"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,

```

```

future_max_size = 40 * 1024^3

# ForestsTreesAge_ConiferousYoung_r3000.tif egv_361
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousYoung_r3000.tif")
names(slanis)="egv_361"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousYoung_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_ConiferousYoung_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.362 ForestsTreesAge\_ConiferousYoung\_r10000

**filename:** ForestsTreesAge\_ConiferousYoung\_r10000.tif

**layername:** egv\_362

**English name:** Fractional cover of Young (pre-rotation age) Coniferous Forests within the 10 km landscape

**Latvian name:** Jaunu (pirms cirtmeta) skujkoku mežu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousYoung_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_ConiferousYoung"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",

```

```

extract_fun  = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3)

# ForestsTreesAge_ConiferousYoung_r10000.tif    egv_362
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousYoung_r10000.tif")
names(slanis)="egv_362"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTreesAge_ConiferousYoung_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_ConiferousYoung_r10000.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.363 ForestsTreesAge\_MixedOld\_cell

**filename:** ForestsTreesAge\_MixedOld\_cell.tif

**layername:** egv\_363

**English name:** Fractional cover of Old (over rotation age) Mixed Forests within the analysis cell (1 ha)

**Latvian name:** Vecu (kopš cirtmeta) jauktu koku mežu plātības īpatsvars analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

To prepare this EGV, stands from the [State Forest Service's State Forest Registry](#) are classified into (in order):

- coniferous (see [Terminology and acronyms](#) for species codes) if timber volume of those species exceeded 75%;
- Boreal deciduous if timber volume of those species exceeded 75%;
- temperate deciduous if timber volume of those species exceeded 50%;
- mixed otherwise;

then mixed stands exceeding the legal rotation age are selected and geometries are rasterised (presence = 1, NA otherwise). Rasterisation is performed using the workflow `egvtools::polygon2input()`, restricting to pixels outside clearcut mask and covering background with value 0. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsTreesAge_MixedOld_cell.tif egv_363 ----
skujkoki=c("1","3","13","14","15","22","23","28") # 8
saurlapji=c("4","6","8","9","19","20","21","32","35","68") # 10
platlapji=c("10","11","12","16","17","18","24","25","26","27","28","29","50",
  "61","62","63","64","65","66","67","69") # 21
mvr=mvr %>%
  mutate(kraja_skujkoku=ifelse(s10 %in% skujkoki,v10,0) +
    ifelse(s11 %in% skujkoki,v11,0)+ifelse(s12 %in% skujkoki,v12,0) +
    ifelse(s13 %in% skujkoki,v13,0)+ifelse(s14 %in% skujkoki,v14,0),
  kraja_saurlapju=ifelse(s10 %in% saurlapji,v10,0) +
    ifelse(s11 %in% saurlapji,v11,0)+ifelse(s12 %in% saurlapji,v12,0) +
    ifelse(s13 %in% saurlapji,v13,0)+ifelse(s14 %in% saurlapji,v14,0),
  kraja_platlapju=ifelse(s10 %in% platlapji,v10,0) +
    ifelse(s11 %in% platlapji,v11,0)+ifelse(s12 %in% platlapji,v12,0) +
    ifelse(s13 %in% platlapji,v13,0)+ifelse(s14 %in% platlapji,v14,0))

```

```

  ifelse(s13 %in% platlapji,v13,0)+ifelse(s14 %in% platlapji,v14,0)) %>%
  mutate(kopeja_kraja=kraja_skujkoku+kraja_platlapju+kraja_saurlapju) %>%
  mutate(tips=ifelse(kraja_skujkoku/kopeja_kraja>=0.75,"skujkoku",
                     ifelse(kraja_saurlapju/kopeja_kraja>=0.75,"saurlapju",
                     ifelse(kraja_platlapju/kopeja_kraja>0.5,"platlapju",
                     "jauktu koku"))))

nogabali=mvr %>%
  filter(zkat=="10"&tips=="jauktu koku"&(vgr=="4" | vgr=="5"))

p2i_rez=egvtools::polygon2input(vector_data = nogabali,
                                 template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                 out_path = "./RasterGrids_10m/2024/",
                                 file_name = "ForestsTreesAge_MixedOld_input.tif",
                                 value_field = "yes",
                                 restrict_to = clearcut_mask,
                                 restrict_values = 0,
                                 prepare=FALSE,
                                 background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                 plot_result = TRUE)

p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "ForestsTreesAge_MixedOld_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/Raw/",
                             outfilename = "ForestsTreesAge_MixedOld_cell.tif",
                             layername = "egv_363",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(nogabali)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsTreesAge_MixedOld_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_MixedOld_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.364 ForestsTreesAge\_MixedOld\_r500

**filename:** ForestsTreesAge\_MixedOld\_r500.tif

**layername:** egv\_364

**English name:** Fractional cover of Old (over rotation age) Mixed Forests within the 0.5 km landscape

**Latvian name:** Vecu (kopš cirtmeta) jauktu koku mežu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the

output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/ForestsTreesAge_MixedOld_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_MixedOld"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii        = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_MixedOld_r500.tif egv_364
slanis=rast("./RasterGrids_100m/2024/Raw/ForestsTreesAge_MixedOld_r500.tif")
names(slanis)="egv_364"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/ForestsTreesAge_MixedOld_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_MixedOld_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.365 ForestsTreesAge\_MixedOld\_r1250

**filename:** ForestsTreesAge\_MixedOld\_r1250.tif

**layername:** egv\_365

**English name:** Fractional cover of Old (over rotation age) Mixed Forests within the 1.25 km landscape

**Latvian name:** Vecu (kopš cirtmeta) jauktu koku mežu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_MixedOld_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_MixedOld"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_MixedOld_r1250.tif      egv_365
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_MixedOld_r1250.tif")
names(slanis)="egv_365"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsTreesAge_MixedOld_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_MixedOld_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.366 ForestsTreesAge\_MixedOld\_r3000

**filename:** ForestsTreesAge\_MixedOld\_r3000.tif

**layername:** egv\_366

**English name:** Fractional cover of Old (over rotation age) Mixed Forests within the 3 km landscape

**Latvian name:** Vecu (kopš cirtmeta) jauktu koku mežu plātības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes:::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/ForestsTreesAge_MixedOld_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_MixedOld"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_MixedOld_r3000.tif    egv_366
slanis=rast("./RasterGrids_100m/2024/Raw/ForestsTreesAge_MixedOld_r3000.tif")
names(slanis)="egv_366"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/ForestsTreesAge_MixedOld_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_MixedOld_r3000.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.367 ForestsTreesAge\_MixedOld\_r10000

**filename:** ForestsTreesAge\_MixedOld\_r10000.tif

**layername:** egv\_367

**English name:** Fractional cover of Old (over rotation age) Mixed Forests within the 10 km landscape

**Latvian name:** Vecu (kopš cirtmeta) jauktu koku mežu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/ForestsTreesAge_MixedOld_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_MixedOld"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_MixedOld_r10000.tif  egv_367
slanis=rast("./RasterGrids_100m/2024/Raw/ForestsTreesAge_MixedOld_r10000.tif")
names(slanis)="egv_367"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/ForestsTreesAge_MixedOld_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_MixedOld_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.368 ForestsTreesAge\_MixedYoung\_cell

**filename:** ForestsTreesAge\_MixedYoung\_cell.tif

**layername:** egv\_368

**English name:** Fractional cover of Young (pre-rotation age) Mixed Forests within the analysis cell (1 ha)

**Latvian name:** Jaunu (pirms cirtmeta) jauktu koku mežu platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

To prepare this EGV, stands from the [State Forest Service's State Forest Registry](#) are classified into (in order):

- coniferous (see [Terminology and acronyms](#) for species codes) if timber volume of those species exceeded 75%;
- Boreal deciduous if timber volume of those species exceeded 75%;
- temperate deciduous if timber volume of those species exceeded 50%;
- mixed otherwise;

then mixed stands younger than the legal rotation age are selected and geometries are rasterised (presence = 1, NA otherwise). Rasterisation is performed using the workflow `egvtools::polygon2input()`, restricting to pixels outside clearcut mask and covering background with value 0. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
```

```

r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
    filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
    overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsTreesAge_MixedYoung_cell.tif   evg_368 ----
skujkoki=c("1","3","13","14","15","22","23","28") # 8
saurlapji=c("4","6","8","9","19","20","21","32","35","68") # 10
platlapji=c("10","11","12","16","17","18","24","25","26","27","28","29","50",
           "61","62","63","64","65","66","67","69") # 21
mvr=mvr %>%
  mutate(kraja_skujkoku=ifelse(s10 %in% skujkoki,v10,0)+
         ifelse(s11 %in% skujkoki,v11,0)+ifelse(s12 %in% skujkoki,v12,0)+
         ifelse(s13 %in% skujkoki,v13,0)+ifelse(s14 %in% skujkoki,v14,0),
         kraja_saurlapju=ifelse(s10 %in% saurlapji,v10,0)+
         ifelse(s11 %in% saurlapji,v11,0)+ifelse(s12 %in% saurlapji,v12,0)+
         ifelse(s13 %in% saurlapji,v13,0)+ifelse(s14 %in% saurlapji,v14,0),
         kraja_platlapju=ifelse(s10 %in% platlapji,v10,0)+
         ifelse(s11 %in% platlapji,v11,0)+ifelse(s12 %in% platlapji,v12,0)+
         ifelse(s13 %in% platlapji,v13,0)+ifelse(s14 %in% platlapji,v14,0)) %>%
  mutate(kopeja_kraja=kraja_skujkoku+kraja_platlapju+kraja_saurlapju) %>%
  mutate(tips=ifelse(kraja_skujkoku/kopeja_kraja>=0.75,"skujkoku",
                     ifelse(kraja_saurlapju/kopeja_kraja>=0.75,"saurlapju",
                           ifelse(kraja_platlapju/kopeja_kraja>0.5,"platlapju",
                                 "jauktu koku"))))

nogabali=mvr %>%
  filter(zkat=="10"&tips=="jauktu koku"&(vgr=="1"|vgr=="2"|vgr=="3"))

p2i_rez=egvtools::polygon2input(vector_data = nogabali,
                                template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                out_path = "./RasterGrids_10m/2024/",
                                file_name = "ForestsTreesAge_MixedYoung_input.tif",
                                value_field = "yes",
                                restrict_to = clearcut_mask,
                                restrict_values = 0,
                                prepare=FALSE,
                                background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                plot_result = TRUE)

p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                           "ForestsTreesAge_MixedYoung_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/Raw/",
                             outfilename = "ForestsTreesAge_MixedYoung_cell.tif",
                             layername = "egv_368",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)

i2e_rez

```

```

rm(nogabali)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsTreesAge_MixedYoung_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_MixedYoung_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.369 ForestsTreesAge\_MixedYoung\_r500

**filename:** ForestsTreesAge\_MixedYoung\_r500.tif

**layername:** egv\_369

**English name:** Fractional cover of Young (pre-rotation age) Mixed Forests within the 0.5 km landscape

**Latvian name:** Jaunu (pirms cirtmeta) jauktu koku mežu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_10m/2024/RAW/ForestsTreesAge_MixedYoung_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_MixedYoung"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

```

```

# ForestsTreesAge_MixedYoung_r500.tif    evg_369
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_MixedYoung_r500.tif")
names(slanis)="evg_369"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTreesAge_MixedYoung_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_MixedYoung_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.370 ForestsTreesAge\_MixedYoung\_r1250

**filename:** ForestsTreesAge\_MixedYoung\_r1250.tif

**layername:** evg\_370

**English name:** Fractional cover of Young (pre-rotation age) Mixed Forests within the 1.25 km landscape

**Latvian name:** Jaunu (pirms cirtmeta) jauktu koku mežu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_MixedYoung_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_MixedYoung"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,

```

```

future_max_size = 40 * 1024^3)

# ForestsTreesAge_MixedYoung_r1250.tif  egv_370
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_MixedYoung_r1250.tif")
names(slanis)="egv_370"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTreesAge_MixedYoung_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_MixedYoung_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.371 ForestsTreesAge\_MixedYoung\_r3000

**filename:** ForestsTreesAge\_MixedYoung\_r3000.tif

**layername:** egv\_371

**English name:** Fractional cover of Young (pre-rotation age) Mixed Forests within the 3 km landscape

**Latvian name:** Jaunu (pirms cirtmeta) jauktu koku mežu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_MixedYoung_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_MixedYoung"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",

```

```

extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# ForestsTreesAge_MixedYoung_r3000.tif egv_371
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_MixedYoung_r3000.tif")
names(slanis)="egv_371"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTreesAge_MixedYoung_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_MixedYoung_r3000.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.372 ForestsTreesAge\_MixedYoung\_r10000

**filename:** ForestsTreesAge\_MixedYoung\_r10000.tif

**layername:** egv\_372

**English name:** Fractional cover of Young (pre-rotation age) Mixed Forests within the 10 km landscape

**Latvian name:** Jaunu (pirms cirtmeta) jauktu koku mežu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_MixedYoung_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_MixedYoung"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",

```

```

n_workers    = 6,
radii        = c("r10000"),
radius_mode  = "sparse",
extract_fun   = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3)

# ForestsTreesAge_MixedYoung_r10000.tif egv_372
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_MixedYoung_r10000.tif")
names(slanis)="egv_372"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTreesAge_MixedYoung_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_MixedYoung_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

### 6.373 ForestsTreesAge\_TemperateDeciduousOld\_cell

**filename:** ForestsTreesAge\_TemperateDeciduousOld\_cell.tif

**layername:** egv\_373

**English name:** Fractional cover of Old (over rotation age) Temperate Deciduous Forests within the analysis cell (1 ha)

**Latvian name:** Vecu (kopš cirtmeta) platlapju mežu platības īpatsvars analīzē šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

To prepare this EGV, stands from the [State Forest Service's State Forest Registry](#) are classified into (in order):

- coniferous (see [Terminology and acronyms](#) for species codes) if timber volume of those species exceeded 75%;
- Boreal deciduous if timber volume of those species exceeded 75%;
- temperate deciduous if timber volume of those species exceeded 50%;
- mixed otherwise;

then temperate deciduous stands exceeding the legal rotation age are selected and geometries are rasterised (presence = 1, NA otherwise). Rasterisation is performed using the workflow `egvtools::polygon2input()`, restricting to pixels outside clearcut mask and covering background with value 0. The resulting layer is then

aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsTreesAge_TemperateDeciduousOld_cell.tif    egv_373 ----
skujkoki=c("1","3","13","14","15","22","23","28") # 8
saurlapji=c("4","6","8","9","19","20","21","32","35","68") # 10
platlapji=c("10","11","12","16","17","18","24","25","26","27","28","29","50",
  "61","62","63","64","65","66","67","69") # 21
mvr=mvr %>%
  mutate(kraja_skujkoku=ifelse(s10 %in% skujkoki,v10,0)+
    ifelse(s11 %in% skujkoki,v11,0)+ifelse(s12 %in% skujkoki,v12,0)+
    ifelse(s13 %in% skujkoki,v13,0)+ifelse(s14 %in% skujkoki,v14,0),
```

```

kraja_saurlapju=ifelse(s10 %in% saurlapji,v10,0)+  

  ifelse(s11 %in% saurlapji,v11,0)+ifelse(s12 %in% saurlapji,v12,0)+  

  ifelse(s13 %in% saurlapji,v13,0)+ifelse(s14 %in% saurlapji,v14,0),  

  kraja_platlapju=ifelse(s10 %in% platlapji,v10,0)+  

  ifelse(s11 %in% platlapji,v11,0)+ifelse(s12 %in% platlapji,v12,0)+  

  ifelse(s13 %in% platlapji,v13,0)+ifelse(s14 %in% platlapji,v14,0)) %>%  

  mutate(kopeja_kraja=kraja_skujkoku+kraja_platlapju+kraja_saurlapju) %>%  

  mutate(tips=ifelse(kraja_skujkoku/kopeja_kraja>=0.75,"skujkoku",  

    ifelse(kraja_saurlapju/kopeja_kraja>=0.75,"saurlapju",  

      ifelse(kraja_platlapju/kopeja_kraja>0.5,"platlapju",  

        "jauktu koku")))  

nogabali=mvr %>%  

  filter(zkat=="10"&tips=="platlapju"&(vgr=="4" | vgr=="5"))

p2i_rez=egvtools::polygon2input(vector_data = nogabali,  

  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",  

  out_path = "./RasterGrids_10m/2024/",  

  file_name = "ForestsTreesAge_TemperateDeciduousOld_input.tif",  

  value_field = "yes",  

  restrict_to = clearcut_mask,  

  restrict_values = 0,  

  prepare=FALSE,  

  background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",  

  plot_result = TRUE)

p2i_rez  

i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",  

  "ForestsTreesAge_TemperateDeciduousOld_input.tif"),  

  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",  

  summary_function = "average",  

  missing_job = "FillOutput",  

  outlocation = "./RasterGrids_100m/2024/Raw/",  

  outfilename = "ForestsTreesAge_TemperateDeciduousOld_cell.tif",  

  layername = "egv_373",  

  idw_weight = 2,  

  plot_gaps = FALSE,plot_final = TRUE)

i2e_rez  

rm(nogabali)  

rm(p2i_rez)  

rm(i2e_rez)  

unlink("./RasterGrids_10m/2024/ForestsTreesAge_TemperateDeciduousOld_input.tif")

# standardisation ----  

if(!require(terra)) {install.packages("terra"); require(terra)}  

if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_TemperateDeciduousOld_cell.tif"  

ielasianas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)  

saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)  

slanis=rast(ielasianas_cels)  

videjais=global(slanis,fun="mean",na.rm=TRUE)  

centrets=slanis-videjais[,1]  

standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)  

merogots=centrets/standartnovirze[,1]  

writeRaster(merogots,  

  filename=saglabasanas_cels,  

  overwrite=TRUE)

```

## 6.374 ForestsTreesAge\_TemperateDeciduousOld\_r500

**filename:** ForestsTreesAge\_TemperateDeciduousOld\_r500.tif

**layername:** egv\_374

**English name:** Fractional cover of Old (over rotation age) Temperate Deciduous Forests within the 0.5 km

landscape

**Latvian name:** Vecu (kopš cirtmeta) platlapju mežu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   =
    ↳ c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousOld_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_TemperateDeciduousOld"),
  output_dir     = "./RasterGrids_100m/2024/RAW/",
  n_workers      = 6,
  radii          = c("r500"),
  radius_mode    = "sparse",
  extract_fun    = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_TemperateDeciduousOld_r500.tif    egv_374
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousOld_r500.tif")
names(slanis)="egv_374"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousOld_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_TemperateDeciduousOld_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)
```

## 6.375 ForestsTreesAge\_TemperateDeciduousOld\_r1250

**filename:** ForestsTreesAge\_TemperateDeciduousOld\_r1250.tif

**layername:** egv\_375

**English name:** Fractional cover of Old (over rotation age) Temperate Deciduous Forests within the 1.25 km landscape

**Latvian name:** Vecu (kopš cirtmeta) platlapju mežu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousOld_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_TemperateDeciduousOld"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_TemperateDeciduousOld_r1250.tif  egv_375
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousOld_r1250.tif")
names(slanis)="egv_375"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousOld_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_TemperateDeciduousOld_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
```

```

filename=saglabasanas_cels,
overwrite=TRUE)

```

## 6.376 ForestsTreesAge\_TemperateDeciduousOld\_r3000

**filename:** ForestsTreesAge\_TemperateDeciduousOld\_r3000.tif

**layername:** egv\_376

**English name:** Fractional cover of Old (over rotation age) Temperate Deciduous Forests within the 3 km landscape

**Latvian name:** Vecu (kopš cirtmeta) platlapju mežu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path   = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   =
    c("./RasterGrids_100m/2024/Raw/ForestsTreesAge_TemperateDeciduousOld_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_TemperateDeciduousOld"),
  output_dir     = "./RasterGrids_100m/2024/Raw/",
  n_workers      = 6,
  radii          = c("r3000"),
  radius_mode    = "sparse",
  extract_fun    = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_TemperateDeciduousOld_r3000.tif  egv_376
slanis=rast("./RasterGrids_100m/2024/Raw/ForestsTreesAge_TemperateDeciduousOld_r3000.tif")
names(slanis)="egv_376"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/ForestsTreesAge_TemperateDeciduousOld_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_TemperateDeciduousOld_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)

```

```

slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.377 ForestsTreesAge\_TemperateDeciduousOld\_r10000

**filename:** ForestsTreesAge\_TemperateDeciduousOld\_r10000.tif

**layername:** egv\_377

**English name:** Fractional cover of Old (over rotation age) Temperate Deciduous Forests within the 10 km landscape

**Latvian name:** Vecu (kopš cirtmeta) platlapju mežu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/Raw/ForestsTreesAge_TemperateDeciduousOld_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_TemperateDeciduousOld"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing  = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_TemperateDeciduousOld_r10000.tif egv_377
slanis=rast("./RasterGrids_100m/2024/Raw/ForestsTreesAge_TemperateDeciduousOld_r10000.tif")
names(slanis)="egv_377"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/ForestsTreesAge_TemperateDeciduousOld_r10000.tif",
  overwrite=TRUE)

# standardisation ----

```

```

if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_TemperateDeciduousOld_r1000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.378 ForestsTreesAge\_TemperateDeciduousYoung\_cell

**filename:** ForestsTreesAge\_TemperateDeciduousYoung\_cell.tif

**layername:** egv\_378

**English name:** Fractional cover of Young (pre-rotation age) Temperate Deciduous Forests within the analysis cell (1 ha)

**Latvian name:** Jaunu (pirms cirtmeta) platlapju mežu platības īpatsvars analīzēs šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

To prepare this EGV, stands from the [State Forest Service's State Forest Registry](#) are classified into (in order):

- coniferous (see [Terminology and acronyms](#) for species codes) if timber volume of those species exceeded 75%;
- Boreal deciduous if timber volume of those species exceeded 75%;
- temperate deciduous if timber volume of those species exceeded 50%;
- mixed otherwise;

then temperate deciduous stands younger than the legal rotation age are selected and geometries are rasterised (presence = 1, NA otherwise). Rasterisation is performed using the workflow `egvtools::polygon2input()`, restricting to pixels outside clearcut mask and covering background with value 0. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

```

```

template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsTreesAge_TemperateDeciduousYoung_cell.tif  evg_378 ----
skujkoki=c("1","3","13","14","15","22","23","28") # 8
saurlapji=c("4","6","8","9","19","20","21","32","35","68") # 10
platlapji=c("10","11","12","16","17","18","24","25","26","27","28","29","50",
  "61","62","63","64","65","66","67","69") # 21
mvr=mvr %>%
  mutate(kraja_skujkoku=ifelse(s10 %in% skujkoki,v10,0) +
    ifelse(s11 %in% skujkoki,v11,0)+ifelse(s12 %in% skujkoki,v12,0) +
    ifelse(s13 %in% skujkoki,v13,0)+ifelse(s14 %in% skujkoki,v14,0),
    kraja_saurlapju=ifelse(s10 %in% saurlapji,v10,0) +
    ifelse(s11 %in% saurlapji,v11,0)+ifelse(s12 %in% saurlapji,v12,0) +
    ifelse(s13 %in% saurlapji,v13,0)+ifelse(s14 %in% saurlapji,v14,0),
    kraja_platlapju=ifelse(s10 %in% platlapji,v10,0) +
    ifelse(s11 %in% platlapji,v11,0)+ifelse(s12 %in% platlapji,v12,0) +
    ifelse(s13 %in% platlapji,v13,0)+ifelse(s14 %in% platlapji,v14,0)) %>%
  mutate(kopeja_kraja=kraja_skujkoku+kraja_platlapju+kraja_saurlapju) %>%
  mutate(tips=ifelse(kraja_skujkoku/kopeja_kraja>=0.75,"skujkoku",
    ifelse(kraja_saurlapju/kopeja_kraja>=0.75,"saurlapju",
      ifelse(kraja_platlapju/kopeja_kraja>0.5,"platlapju",
        "jauktu koku"))))

nogabali=mvr %>%
  filter(zkat=="10"&tips=="platlapju"&(vgr=="1"|vgr=="2"|vgr=="3"))

p2i_rez=egvtools::polygon2input(vector_data = nogabali,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsTreesAge_TemperateDeciduousYoung_input.tif",

```

```

        value_field = "yes",
        restrict_to = clearcut_mask,
        restrict_values = 0,
        prepare=FALSE,
        background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
        plot_result = TRUE)
p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "ForestsTreesAge_TemperateDeciduousYoung_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/RAW/",
                             outfilename = "ForestsTreesAge_TemperateDeciduousYoung_cell.tif",
                             layername = "egv_378",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(nogabali)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsTreesAge_TemperateDeciduousYoung_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_TemperateDeciduousYoung_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.379 ForestsTreesAge\_TemperateDeciduousYoung\_r500

**filename:** ForestsTreesAge\_TemperateDeciduousYoung\_r500.tif

**layername:** egv\_379

**English name:** Fractional cover of Young (pre-rotation age) Temperate Deciduous Forests within the 0.5 km landscape

**Latvian name:** Jaunu (pirms cirtmeta) platlapju mežu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----

```

```

template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   =
    ↳ c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousYoung_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_TemperateDeciduousYoung"),
  output_dir     = "./RasterGrids_100m/2024/RAW/",
  n_workers      = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_TemperateDeciduousYoung_r500.tif  egv_379
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousYoung_r500.tif")
names(slanis)="egv_379"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousYoung_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_TemperateDeciduousYoung_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.380 ForestsTreesAge\_TemperateDeciduousYoung\_r1250

**filename:** ForestsTreesAge\_TemperateDeciduousYoung\_r1250.tif

**layername:** egv\_380

**English name:** Fractional cover of Young (pre-rotation age) Temperate Deciduous Forests within the 1.25 km landscape

**Latvian name:** Jaunu (pirms cirtmeta) platlapju mežu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
```

```

if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    ↵ c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousYoung_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_TemperateDeciduousYoung"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_TemperateDeciduousYoung_r1250.tif egv_380
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousYoung_r1250.tif")
names(slanis)="egv_380"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousYoung_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_TemperateDeciduousYoung_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.381 ForestsTreesAge\_TemperateDeciduousYoung\_r3000

**filename:** ForestsTreesAge\_TemperateDeciduousYoung\_r3000.tif

**layername:** egv\_381

**English name:** Fractional cover of Young (pre-rotation age) Temperate Deciduous Forests within the 3 km landscape

**Latvian name:** Jaunu (pirms cirtmeta) platlapju mežu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the

output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousYoung_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_TemperateDeciduousYoung"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_TemperateDeciduousYoung_r3000.tif egv_381
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousYoung_r3000.tif")
names(slanis)="egv_381"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousYoung_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_TemperateDeciduousYoung_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.382 ForestsTreesAge\_TemperateDeciduousYoung\_r10000

**filename:** ForestsTreesAge\_TemperateDeciduousYoung\_r10000.tif

**layername:** egv\_382

**English name:** Fractional cover of Young (pre-rotation age) Temperate Deciduous Forests within the 10 km landscape

**Latvian name:** Jaunu (pirms cirtmeta) platlapju mežu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  =
    ↳ c("./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousYoung_cell.tif"),
  layer_prefixes = c("ForestsTreesAge_TemperateDeciduousYoung"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing  = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTreesAge_TemperateDeciduousYoung_r10000.tif      egv_382
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousYoung_r10000.tif")
names(slanis)="egv_382"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsTreesAge_TemperateDeciduousYoung_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTreesAge_TemperateDeciduousYoung_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.383 ForestsTrees\_BorealDeciduous\_cell

**filename:** ForestsTrees\_BorealDeciduous\_cell.tif

**layername:** evg\_383

**English name:** Fractional cover of Boreal Deciduous Forests within the analysis cell (1 ha)

**Latvian name:** Šaurlapju mežu platības īpatsvars analīzēs šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the State Forest Service's State Forest Registry land category 12 and 14, and The Global Forest Watch pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

To prepare this EGV, stands from the State Forest Service's State Forest Registry are classified into (in order):

- coniferous (see [Terminology and acronyms](#) for species codes) if timber volume of those species exceeded 75%;
- Boreal deciduous if timber volume of those species exceeded 75%;
- temperate deciduous if timber volume of those species exceeded 50%;
- mixed otherwise;

then Boreal deciduous stands are selected and geometries are rasterised (presence = 1, NA otherwise). Rasterisation is performed using the workflow `egvtools::polygon2input()`, restricting to pixels outside clearcut mask and covering background with value 0. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
```

```

tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
    filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
    overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsTrees_BorealDeciduous_cell.tif egv_383 ----
skujkoki=c("1","3","13","14","15","22","23","28") # 8
saurlapji=c("4","6","8","9","19","20","21","32","35","68") # 10
platlapji=c("10","11","12","16","17","18","24","25","26","27","28","29","50",
    "61","62","63","64","65","66","67","69") # 21
mvr=mvr %>%
  mutate(kraja_skujkoku=ifelse(s10 %in% skujkoki,v10,0) +
    ifelse(s11 %in% skujkoki,v11,0)+ifelse(s12 %in% skujkoki,v12,0) +
    ifelse(s13 %in% skujkoki,v13,0)+ifelse(s14 %in% skujkoki,v14,0),
    kraja_saurlapju=ifelse(s10 %in% saurlapji,v10,0) +
    ifelse(s11 %in% saurlapji,v11,0)+ifelse(s12 %in% saurlapji,v12,0) +
    ifelse(s13 %in% saurlapji,v13,0)+ifelse(s14 %in% saurlapji,v14,0),
    kraja_platlapju=ifelse(s10 %in% platlapji,v10,0) +
    ifelse(s11 %in% platlapji,v11,0)+ifelse(s12 %in% platlapji,v12,0) +
    ifelse(s13 %in% platlapji,v13,0)+ifelse(s14 %in% platlapji,v14,0)) %>%
  mutate(kopeja_kraja=kraja_skujkoku+kraja_platlapju+kraja_saurlapju) %>%
  mutate(tips=ifelse(kraja_skujkoku/kopeja_kraja>=0.75,"skujkoku",
    ifelse(kraja_saurlapju/kopeja_kraja>=0.75,"saurlapju",
    ifelse(kraja_platlapju/kopeja_kraja>0.5,"platlapju",
    "jauktu koku"))))

nogabali=mvr %>%
  filter(zkat=="10"&tips=="saurlapju")

p2i_rez=egvtools::polygon2input(vector_data = nogabali,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsTrees_BorealDeciduous_input.tif",
  value_field = "yes",
  restrict_to = clearcut_mask,
  restrict_values = 0,
  prepare=FALSE,
  background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
  plot_result = TRUE)

p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
  "ForestsTrees_BorealDeciduous_input.tif"),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "ForestsTrees_BorealDeciduous_cell.tif",
  layername = "egv_383",
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(nogabali)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsTrees_BorealDeciduous_input.tif")

```

```

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_BorealDeciduous_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.384 ForestsTrees\_BorealDeciduous\_r500

**filename:** ForestsTrees\_BorealDeciduous\_r500.tif

**layername:** egv\_384

**English name:** Fractional cover of Boreal Deciduous Forests within the 0.5 km landscape

**Latvian name:** Šaurlapju mežu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTrees_BorealDeciduous_cell.tif"),
  layer_prefixes = c("ForestsTrees_BorealDeciduous"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsTrees_BorealDeciduous_r500.tif egv_384
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTrees_BorealDeciduous_r500.tif")
names(slanis)="egv_384"
slanis2=project(slanis,template100)
writeRaster(slanis2,

```

```

"./RasterGrids_100m/2024/RAW/ForestsTrees_BorealDeciduous_r500.tif",
overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_BorealDeciduous_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.385 ForestsTrees\_BorealDeciduous\_r1250

**filename:** ForestsTrees\_BorealDeciduous\_r1250.tif

**layername:** evg\_385

**English name:** Fractional cover of Boreal Deciduous Forests within the 1.25 km landscape

**Latvian name:** Šaurlapju mežu platības īpatsvars 1,25 km aina vā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path  = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   = c("./RasterGrids_100m/2024/RAW/ForestsTrees_BorealDeciduous_cell.tif"),
  layer_prefixes = c("ForestsTrees_BorealDeciduous"),
  output_dir     = "./RasterGrids_100m/2024/RAW/",
  n_workers      = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsTrees_BorealDeciduous_r1250.tif      evg_385
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTrees_BorealDeciduous_r1250.tif")

```

```

names(slanis)="egv_385"
slanis2=project(slanis,template100)
writeRaster(slanis2,
    "./RasterGrids_100m/2024/RAW/ForestsTrees_BorealDeciduous_r1250.tif",
    overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_BorealDeciduous_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.386 ForestsTrees\_BorealDeciduous\_r3000

**filename:** ForestsTrees\_BorealDeciduous\_r3000.tif

**layername:** egv\_386

**English name:** Fractional cover of Boreal Deciduous Forests within the 3 km landscape

**Latvian name:** Šaurlapju mežu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
    kvadратi_path = "./Templates/TemplateGrids/tiles/",
    radii_path   = "./Templates/TemplateGridPoints/tiles/",
    tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
    template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
    input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTrees_BorealDeciduous_cell.tif"),
    layer_prefixes = c("ForestsTrees_BorealDeciduous"),
    output_dir    = "./RasterGrids_100m/2024/RAW/",
    n_workers     = 6,
    radii         = c("r3000"),
    radius_mode   = "sparse",
    extract_fun   = "mean",
    fill_missing   = TRUE,
    IDW_weight    = 2,
    future_max_size = 40 * 1024^3)

```

```

# ForestsTrees_BorealDeciduous_r3000.tif      egv_386
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTrees_BorealDeciduous_r3000.tif")
names(slanis)="egv_386"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTrees_BorealDeciduous_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_BorealDeciduous_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.387 ForestsTrees\_BorealDeciduous\_r10000

**filename:** ForestsTrees\_BorealDeciduous\_r10000.tif

**layername:** egv\_387

**English name:** Fractional cover of Boreal Deciduous Forests within the 10 km landscape

**Latvian name:** Šaurlapju mežu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTrees_BorealDeciduous_cell.tif"),
  layer_prefixes = c("ForestsTrees_BorealDeciduous"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,

```

```

IDW_weight = 2,
future_max_size = 40 * 1024^3)

# ForestsTrees_BorealDeciduous_r10000.tif    egv_387
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTrees_BorealDeciduous_r10000.tif")
names(slanis)="egv_387"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTrees_BorealDeciduous_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_BorealDeciduous_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.388 ForestsTrees\_Coniferous\_cell

**filename:** ForestsTrees\_Coniferous\_cell.tif

**layername:** egv\_388

**English name:** Fractional cover of Coniferous Forests within the analysis cell (1 ha)

**Latvian name:** Skujkoku mežu platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

To prepare this EGV, stands from the [State Forest Service's State Forest Registry](#) are classified into (in order):

- coniferous (see [Terminology and acronyms](#) for species codes) if timber volume of those species exceeded 75%;
- Boreal deciduous if timber volume of those species exceeded 75%;
- temperate deciduous if timber volume of those species exceeded 50%;
- mixed otherwise;

then coniferous stands are selected and geometries are rasterised (presence = 1, NA otherwise). Rasterisation is performed using the workflow `evgttools::polygon2input()`, restricting to pixels outside clearcut mask and covering background with value 0. The resulting layer is then aggregated to EGV resolution using the workflow `evgttools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsTrees_Coniferous_cell.tif egv_388 ----
skujkoki=c("1","3","13","14","15","22","23","28") # 8
saurlapji=c("4","6","8","9","19","20","21","32","35","68") # 10
platlapji=c("10","11","12","16","17","18","24","25","26","27","28","29","50",
  "61","62","63","64","65","66","67","69") # 21
mvr=mvr %>%
  mutate(kraja_skujkoku=ifelse(s10 %in% skujkoki,v10,0) +
    ifelse(s11 %in% skujkoki,v11,0)+ifelse(s12 %in% skujkoki,v12,0) +
    ifelse(s13 %in% skujkoki,v13,0)+ifelse(s14 %in% skujkoki,v14,0),
  kraja_saurlapju=ifelse(s10 %in% saurlapji,v10,0) +
    ifelse(s11 %in% saurlapji,v11,0)+ifelse(s12 %in% saurlapji,v12,0) +
    ifelse(s13 %in% saurlapji,v13,0)+ifelse(s14 %in% saurlapji,v14,0),
  kraja_platlapju=ifelse(s10 %in% platlapji,v10,0) +
    ifelse(s11 %in% platlapji,v11,0)+ifelse(s12 %in% platlapji,v12,0) +
    ifelse(s13 %in% platlapji,v13,0)+ifelse(s14 %in% platlapji,v14,0))

```

```

  ifelse(s13 %in% platlapji,v13,0)+ifelse(s14 %in% platlapji,v14,0)) %>%
  mutate(kopeja_kraja=kraja_skujkoku+kraja_platlapju+kraja_saurlapju) %>%
  mutate(tips=ifelse(kraja_skujkoku/kopeja_kraja>=0.75,"skujkoku",
                     ifelse(kraja_saurlapju/kopeja_kraja>=0.75,"saurlapju",
                           ifelse(kraja_platlapju/kopeja_kraja>0.5,"platlapju",
                                 "jauktu koku"))))

nogabali=mvr %>%
  filter(zkat=="10"&tips=="skujkoku")

p2i_rez=egvtools::polygon2input(vector_data = nogabali,
                                template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                out_path = "./RasterGrids_10m/2024/",
                                file_name = "ForestsTrees_Coniferous_input.tif",
                                value_field = "yes",
                                restrict_to = clearcut_mask,
                                restrict_values = 0,
                                prepare=FALSE,
                                background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                plot_result = TRUE)

p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                         "ForestsTrees_Coniferous_input.tif"),
                             egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                             summary_function = "average",
                             missing_job = "FillOutput",
                             outlocation = "./RasterGrids_100m/2024/Raw/",
                             outfilename = "ForestsTrees_Coniferous_cell.tif",
                             layername = "egv_388",
                             idw_weight = 2,
                             plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(nogabali)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsTrees_Coniferous_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_Coniferous_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.389 ForestsTrees\_Coniferous\_r500

**filename:** ForestsTrees\_Coniferous\_r500.tif

**layername:** egv\_389

**English name:** Fractional cover of Coniferous Forests within the 0.5 km landscape

**Latvian name:** Skujkoku mežu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the

output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/ForestsTrees_Coniferous_cell.tif"),
  layer_prefixes = c("ForestsTrees_Coniferous"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTrees_Coniferous_r500.tif egv_389
slanis=rast("./RasterGrids_100m/2024/Raw/ForestsTrees_Coniferous_r500.tif")
names(slanis)="egv_389"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/ForestsTrees_Coniferous_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_Coniferous_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.390 ForestsTrees\_Coniferous\_r1250

**filename:** ForestsTrees\_Coniferous\_r1250.tif

**layername:** egv\_390

**English name:** Fractional cover of Coniferous Forests within the 1.25 km landscape

**Latvian name:** Skujkoku mežu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTrees_Coniferous_cell.tif"),
  layer_prefixes = c("ForestsTrees_Coniferous"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# ForestsTrees_Coniferous_r1250.tif egv_390
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTrees_Coniferous_r1250.tif")
names(slanis)="egv_390"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsTrees_Coniferous_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_Coniferous_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.391 ForestsTrees\_Coniferous\_r3000

**filename:** ForestsTrees\_Coniferous\_r3000.tif

**layername:** egv\_391

**English name:** Fractional cover of Coniferous Forests within the 3 km landscape

**Latvian name:** Skujkoku mežu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes:::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTrees_Coniferous_cell.tif"),
  layer_prefixes = c("ForestsTrees_Coniferous"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTrees_Coniferous_r3000.tif egv_391
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTrees_Coniferous_r3000.tif")
names(slanis)="egv_391"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsTrees_Coniferous_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_Coniferous_r3000.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.392 ForestsTrees\_Coniferous\_r10000

**filename:** ForestsTrees\_Coniferous\_r10000.tif

**layername:** egv\_392

**English name:** Fractional cover of Coniferous Forests within the 10 km landscape

**Latvian name:** Skujkoku mežu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTrees_Coniferous_cell.tif"),
  layer_prefixes = c("ForestsTrees_Coniferous"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTrees_Coniferous_r10000.tif    egv_392
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTrees_Coniferous_r10000.tif")
names(slanis)="egv_392"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsTrees_Coniferous_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_Coniferous_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.393 ForestsTrees\_Mixed\_cell

**filename:** ForestsTrees\_Mixed\_cell.tif

**layername:** egv\_393

**English name:** Fractional cover of Mixed Forests within the analysis cell (1 ha)

**Latvian name:** Jauktu koku mežu platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

To prepare this EGV, stands from the [State Forest Service's State Forest Registry](#) are classified into (in order):

- coniferous (see [Terminology and acronyms](#) for species codes) if timber volume of those species exceeded 75%;
- Boreal deciduous if timber volume of those species exceeded 75%;
- temperate deciduous if timber volume of those species exceeded 50%;
- mixed otherwise;

then mixed stands are selected and geometries are rasterised (presence = 1, NA otherwise). Rasterisation is performed using the workflow `egvtools::polygon2input()`, restricting to pixels outside clearcut mask and covering background with value 0. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
```

```

r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
    filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
    overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsTrees_Mixed_cell.tif   evg_393 ----
skujkoki=c("1","3","13","14","15","22","23","28") # 8
saurlapji=c("4","6","8","9","19","20","21","32","35","68") # 10
platlapji=c("10","11","12","16","17","18","24","25","26","27","28","29","50",
           "61","62","63","64","65","66","67","69") # 21
mvr=mvr %>%
  mutate(kraja_skujkoku=ifelse(s10 %in% skujkoki,v10,0)+
         ifelse(s11 %in% skujkoki,v11,0)+ifelse(s12 %in% skujkoki,v12,0)+
         ifelse(s13 %in% skujkoki,v13,0)+ifelse(s14 %in% skujkoki,v14,0),
        kraja_saurlapju=ifelse(s10 %in% saurlapji,v10,0)+
         ifelse(s11 %in% saurlapji,v11,0)+ifelse(s12 %in% saurlapji,v12,0)+
         ifelse(s13 %in% saurlapji,v13,0)+ifelse(s14 %in% saurlapji,v14,0),
        kraja_platlapju=ifelse(s10 %in% platlapji,v10,0)+
         ifelse(s11 %in% platlapji,v11,0)+ifelse(s12 %in% platlapji,v12,0)+
         ifelse(s13 %in% platlapji,v13,0)+ifelse(s14 %in% platlapji,v14,0)) %>%
  mutate(kopeja_kraja=kraja_skujkoku+kraja_platlapju+kraja_saurlapju) %>%
  mutate(tips=ifelse(kraja_skujkoku/kopeja_kraja>=0.75,"skujkoku",
                     ifelse(kraja_saurlapju/kopeja_kraja>=0.75,"saurlapju",
                           ifelse(kraja_platlapju/kopeja_kraja>0.5,"platlapju",
                                 "jauktu koku"))))

nogabali=mvr %>%
  filter(zkat=="10"&tips=="jauktu koku")

p2i_rez=egvtools::polygon2input(vector_data = nogabali,
                                template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
                                out_path = "./RasterGrids_10m/2024/",
                                file_name = "ForestsTrees_Mixed_input.tif",
                                value_field = "yes",
                                restrict_to = clearcut_mask,
                                restrict_values = 0,
                                prepare=FALSE,
                                background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
                                plot_result = TRUE)

p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",
                                           "ForestsTrees_Mixed_input.tif"),
                            egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                            summary_function = "average",
                            missing_job = "FillOutput",
                            outlocation = "./RasterGrids_100m/2024/Raw/",
                            outfilename = "ForestsTrees_Mixed_cell.tif",
                            layername = "egv_393",
                            idw_weight = 2,
                            plot_gaps = FALSE,plot_final = TRUE)

i2e_rez

```

```

rm(nogabali)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsTrees_Mixed_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_Mixed_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.394 ForestsTrees\_Mixed\_r500

**filename:** ForestsTrees\_Mixed\_r500.tif

**layername:** egv\_394

**English name:** Fractional cover of Mixed Forests within the 0.5 km landscape

**Latvian name:** Jauktu koku mežu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/ForestsTrees_Mixed_cell.tif"),
  layer_prefixes = c("ForestsTrees_Mixed"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing  = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

```

```

# ForestsTrees_Mixed_r500.tif    evg_394
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTrees_Mixed_r500.tif")
names(slanis)="evg_394"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTrees_Mixed_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_Mixed_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.395 ForestsTrees\_Mixed\_r1250

**filename:** ForestsTrees\_Mixed\_r1250.tif

**layername:** evg\_395

**English name:** Fractional cover of Mixed Forests within the 1.25 km landscape

**Latvian name:** Jauktu koku mežu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadратi_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTrees_Mixed_cell.tif"),
  layer_prefixes = c("ForestsTrees_Mixed"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,

```

```

future_max_size = 40 * 1024^3)

# ForestsTrees_Mixed_r1250.tif  egv_395
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTrees_Mixed_r1250.tif")
names(slanis)="egv_395"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTrees_Mixed_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_Mixed_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.396 ForestsTrees\_Mixed\_r3000

**filename:** ForestsTrees\_Mixed\_r3000.tif

**layername:** egv\_396

**English name:** Fractional cover of Mixed Forests within the 3 km landscape

**Latvian name:** Jauktu koku mežu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTrees_Mixed_cell.tif"),
  layer_prefixes = c("ForestsTrees_Mixed"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",

```

```

extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# ForestsTrees_Mixed_r3000.tif egv_396
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTrees_Mixed_r3000.tif")
names(slanis)="egv_396"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTrees_Mixed_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_Mixed_r3000.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.397 ForestsTrees\_Mixed\_r10000

**filename:** ForestsTrees\_Mixed\_r10000.tif

**layername:** egv\_397

**English name:** Fractional cover of Mixed Forests within the 10 km landscape

**Latvian name:** Jauktu koku mežu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTrees_Mixed_cell.tif"),
  layer_prefixes = c("ForestsTrees_Mixed"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",

```

```

n_workers    = 6,
radii        = c("r10000"),
radius_mode  = "sparse",
extract_fun   = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3)

# ForestsTrees_Mixed_r10000.tif egv_397
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTrees_Mixed_r10000.tif")
names(slanis)="egv_397"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/ForestsTrees_Mixed_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_Mixed_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.398 ForestsTrees\_TemperateDeciduous\_cell

**filename:** ForestsTrees\_TemperateDeciduous\_cell.tif

**layername:** egv\_398

**English name:** Fractional cover of Temperate Deciduous Forests within the analysis cell (1 ha)

**Latvian name:** Platlapju mežu platības īpatsvars analīzē šūnā (1 ha)

**Procedure:** Most EGVs describing forests are spatially restricted to areas outside of clearcuts and dead stands. This mask is created using a combination of the [State Forest Service's State Forest Registry](#) land category 12 and 14, and [The Global Forest Watch](#) pixels classified as lost tree canopy cover since 2020 (raster layer matching input, presence = 1, absence = 0).

To prepare this EGV, stands from the [State Forest Service's State Forest Registry](#) are classified into (in order):

- coniferous (see [Terminology and acronyms](#) for species codes) if timber volume of those species exceeded 75%;
- Boreal deciduous if timber volume of those species exceeded 75%;
- temperate deciduous if timber volume of those species exceeded 50%;
- mixed otherwise;

then temperate deciduous stands are selected and geometries are rasterised (presence = 1, NA otherwise). Rasterisation is performed using the workflow `egvtools::polygon2input()`, restricting to pixels outside clearcut mask and covering background with value 0. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the

cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# mvr ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
mvr$yes=1

# clear cut mask ----
izcirtumi=mvr %>%
  filter(zkat %in% c("12","14")) %>%
  dplyr::select(yes)
r_izcirtumi_mvr=fasterize(izcirtumi,rastrs10,field="yes")
t_izcirtumi_mvr=rast(r_izcirtumi_mvr)
plot(t_izcirtumi_mvr)

tcl=rast("./Geodata/2024/Trees/GFW/TreeCoverLoss_v1_12.tif")
tcl2=ifel(tcl<20,0,1)
tclX=cover(tcl2,nulls10)
plot(tclX)

clearcut_mask=cover(t_izcirtumi_mvr,tclX,
  filename="./RasterGrids_10m/2024/Mask_clearcuts.tif",
  overwrite=TRUE)
plot(clearcut_mask)

rm(izcirtumi)
rm(r_izcirtumi_mvr)
rm(t_izcirtumi_mvr)
rm(tcl)
rm(tcl2)
rm(tclX)

# ForestsTrees_TemperateDeciduous_cell.tif  egv_398 ----
skujkoki=c("1","3","13","14","15","22","23","28") # 8
saurlapji=c("4","6","8","9","19","20","21","32","35","68") # 10
platlapji=c("10","11","12","16","17","18","24","25","26","27","28","29","50",
  "61","62","63","64","65","66","67","69") # 21
mvr=mvr %>%
  mutate(kraja_skujkoku=ifelse(s10 %in% skujkoki,v10,0)+
    ifelse(s11 %in% skujkoki,v11,0)+ifelse(s12 %in% skujkoki,v12,0)+
    ifelse(s13 %in% skujkoki,v13,0)+ifelse(s14 %in% skujkoki,v14,0),
  kraja_saurlapju=ifelse(s10 %in% saurlapji,v10,0)+
```

```

    ifelse(s11 %in% saurlapji,v11,0)+ifelse(s12 %in% saurlapji,v12,0)+  

    ifelse(s13 %in% saurlapji,v13,0)+ifelse(s14 %in% saurlapji,v14,0),  

    kraja_platlapju=ifelse(s10 %in% platlapji,v10,0)+  

    ifelse(s11 %in% platlapji,v11,0)+ifelse(s12 %in% platlapji,v12,0)+  

    ifelse(s13 %in% platlapji,v13,0)+ifelse(s14 %in% platlapji,v14,0)) %>%  

  mutate(kopeja_kraja=kraja_skujkoku+kraja_platlapju+kraja_saurlapju) %>%  

  mutate(tips=ifelse(kraja_skujkoku/kopeja_kraja>=0.75,"skujkoku",  

    ifelse(kraja_saurlapju/kopeja_kraja>=0.75,"saurlapju",  

    ifelse(kraja_platlapju/kopeja_kraja>0.5,"platlapju",  

      "jauktu koku"))))  

nogabali=mvr %>%
  filter(zkat=="10"&tips=="platlapju")

p2i_rez=egvtools::polygon2input(vector_data = nogabali,
  template_path = "./Templates/TemplateRasters/LV10m_10km.tif",
  out_path = "./RasterGrids_10m/2024/",
  file_name = "ForestsTrees_TemperateDeciduous_input.tif",
  value_field = "yes",
  restrict_to = clearcut_mask,
  restrict_values = 0,
  prepare=FALSE,
  background_raster = "./Templates/TemplateRasters/nulls_LV10m_10km.tif",
  plot_result = TRUE)

p2i_rez
i2e_rez=egvtools::input2egv(input=paste0("./RasterGrids_10m/2024/",  

  "ForestsTrees_TemperateDeciduous_input.tif"),
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "ForestsTrees_TemperateDeciduous_cell.tif",
  layername = "egv_398",
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(nogabali)
rm(p2i_rez)
rm(i2e_rez)
unlink("./RasterGrids_10m/2024/ForestsTrees_TemperateDeciduous_input.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_TemperateDeciduous_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.399 ForestsTrees\_TemperateDeciduous\_r500

**filename:** ForestsTrees\_TemperateDeciduous\_r500.tif

**layername:** egv\_399

**English name:** Fractional cover of Temperate Deciduous Forests within the 0.5 km landscape

**Latvian name:** Platlapju mežu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTrees_TemperateDeciduous_cell.tif"),
  layer_prefixes = c("ForestsTrees_TemperateDeciduous"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTrees_TemperateDeciduous_r500.tif  egv_399
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTrees_TemperateDeciduous_r500.tif")
names(slanis)="egv_399"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsTrees_TemperateDeciduous_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_TemperateDeciduous_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.400 ForestsTrees\_TemperateDeciduous\_r1250

**filename:** ForestsTrees\_TemperateDeciduous\_r1250.tif

**layername:** egv\_400

**English name:** Fractional cover of Temperate Deciduous Forests within the 1.25 km landscape

**Latvian name:** Platlapju mežu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes:::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTrees_TemperateDeciduous_cell.tif"),
  layer_prefixes = c("ForestsTrees_TemperateDeciduous"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTrees_TemperateDeciduous_r1250.tif egv_400
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTrees_TemperateDeciduous_r1250.tif")
names(slanis)="egv_400"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsTrees_TemperateDeciduous_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_TemperateDeciduous_r1250.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.401 ForestsTrees\_TemperateDeciduous\_r3000

**filename:** ForestsTrees\_TemperateDeciduous\_r3000.tif

**layername:** egv\_401

**English name:** Fractional cover of Temperate Deciduous Forests within the 3 km landscape

**Latvian name:** Platlapju mežu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTrees_TemperateDeciduous_cell.tif"),
  layer_prefixes = c("ForestsTrees_TemperateDeciduous"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTrees_TemperateDeciduous_r3000.tif egv_401
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTrees_TemperateDeciduous_r3000.tif")
names(slanis)="egv_401"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsTrees_TemperateDeciduous_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_TemperateDeciduous_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.402 ForestsTrees\_TemperateDeciduous\_r10000

**filename:** ForestsTrees\_TemperateDeciduous\_r10000.tif

**layername:** egv\_402

**English name:** Fractional cover of Temperate Deciduous Forests within the 10 km landscape

**Latvian name:** Platlapju mežu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/ForestsTrees_TemperateDeciduous_cell.tif"),
  layer_prefixes = c("ForestsTrees_TemperateDeciduous"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# ForestsTrees_TemperateDeciduous_r10000.tif      egv_402
slanis=rast("./RasterGrids_100m/2024/RAW/ForestsTrees_TemperateDeciduous_r10000.tif")
names(slanis)="egv_402"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/ForestsTrees_TemperateDeciduous_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="ForestsTrees_TemperateDeciduous_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.403 General\_AllotmentGardens\_cell

**filename:** General\_AllotmentGardens\_cell.tif

**layername:** evg\_403

**English name:** Fractional cover of Allotment gardens within the analysis cell (1 ha)

**Latvian name:** Vasarnīcu kompleksu platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, the allotment gardens and farmsteads from the [Landscape classification](#) are selected (value 410 is reclassified to value 1; all others are set to 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# General_AllotmentGardens_cell.tif evg_403 ----
allotmentgardens;ifel(simple_landscape==410,1,0)
i2e_rez=egvtools::input2egv(input=allotmentgardens,
                           egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                           summary_function = "average",
                           missing_job = "FillOutput",
                           outlocation = "./RasterGrids_100m/2024/RAW/",
                           outfilename = "General_AllotmentGardens_cell.tif",
                           layername = "evg_403",
                           idw_weight = 2,
                           plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(allotmentgardens)
rm(i2e_rez)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_AllotmentGardens_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
```

```

writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.404 General\_AllotmentGardens\_r500

**filename:** General\_AllotmentGardens\_r500.tif

**layername:** evg\_404

**English name:** Fractional cover of Allotment gardens within the 0.5 km landscape

**Latvian name:** Vasarnīcu kompleksu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_AllotmentGardens_cell.tif"),
  layer_prefixes = c("General_AllotmentGardens"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing  = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_AllotmentGardens_r500.tif evg_404
slanis=rast("./RasterGrids_100m/2024/RAW/General_AllotmentGardens_r500.tif")
names(slanis)="evg_404"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_AllotmentGardens_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_AllotmentGardens_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)

```

```

centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.405 General\_AllotmentGardens\_r1250

**filename:** General\_AllotmentGardens\_r1250.tif

**layername:** egv\_405

**English name:** Fractional cover of Allotment gardens within the 1.25 km landscape

**Latvian name:** Vasarnīcu kompleksu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path   = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers    = c("./RasterGrids_100m/2024/RAW/General_AllotmentGardens_cell.tif"),
  layer_prefixes = c("General_AllotmentGardens"),
  output_dir     = "./RasterGrids_100m/2024/RAW/",
  n_workers      = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# General_AllotmentGardens_r1250.tif    egv_405
slanis=rast("./RasterGrids_100m/2024/RAW/General_AllotmentGardens_r1250.tif")
names(slanis)="egv_405"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_AllotmentGardens_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_AllotmentGardens_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)

```

```

saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.406 General\_AllotmentGardens\_r3000

**filename:** General\_AllotmentGardens\_r3000.tif

**layername:** evg\_406

**English name:** Fractional cover of Allotment gardens within the 3 km landscape

**Latvian name:** Vasarnīcu kompleksu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/General_AllotmentGardens_cell.tif"),
  layer_prefixes = c("General_AllotmentGardens"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_AllotmentGardens_r3000.tif      evg_406
slanis=rast("./RasterGrids_100m/2024/Raw/General_AllotmentGardens_r3000.tif")
names(slanis)="evg_406"
slanis2=project(slanis,template100)
writeRaster(slanis2,
    "./RasterGrids_100m/2024/Raw/General_AllotmentGardens_r3000.tif",
    overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

nosaukums="General_AllotmentGardens_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.407 General\_AllotmentGardens\_r10000

**filename:** General\_AllotmentGardens\_r10000.tif

**layername:** evg\_407

**English name:** Fractional cover of Allotment gardens within the 10 km landscape

**Latvian name:** Vasarnīcu kompleksu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadri_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_AllotmentGardens_cell.tif"),
  layer_prefixes = c("General_AllotmentGardens"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# General_AllotmentGardens_r10000.tif  evg_407
slanis=rast("./RasterGrids_100m/2024/RAW/General_AllotmentGardens_r10000.tif")
names(slanis)="evg_407"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_AllotmentGardens_r10000.tif",
            overwrite=TRUE)

```

```

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_AllotmentGardens_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.408 General\_BareSoilQuarry\_cell

**filename:** General\_BareSoilQuarry\_cell.tif

**layername:** egv\_408

**English name:** Fractional cover of areas with Bare Soil, Quarries within the analysis cell (1 ha)

**Latvian name:** Atklātas augsnes un karjeru platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, the bare soil and quarry areas from the [Landscape classification](#) are selected (value 800 is reclassified to value 1; all others are set to 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# General_BareSoilQuarry_cell.tif  egv_408 ----
baresoilquerry=ifel(simple_landscape==800,1,0)
i2e_rez=egvtools::input2egv(input=baresoilquerry,
                           egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                           summary_function = "average",
                           missing_job = "FillOutput",
                           outlocation = "./RasterGrids_100m/2024/RAW/",
                           outfilename = "General_BareSoilQuarry_cell.tif",
                           
```

```

        layername = "egv_408",
        idw_weight = 2,
        plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(baresoilquerry)
rm(i2e_rez)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_BareSoilQuarry_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.409 General\_BareSoilQuarry\_r500

**filename:** General\_BareSoilQuarry\_r500.tif

**layername:** egv\_409

**English name:** Fractional cover of areas with Bare Soil, Quarries within the 0.5 km landscape

**Latvian name:** Atklātas augsnes un karjeru platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_BareSoilQuarry_cell.tif"),
  layer_prefixes = c("General_BareSoilQuarry"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

```

```

# General_BareSoilQuarry_r500.tif    egv_409
slanis=rast("./RasterGrids_100m/2024/RAW/General_BareSoilQuarry_r500.tif")
names(slanis)="egv_409"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_BareSoilQuarry_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_BareSoilQuarry_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.410 General\_BareSoilQuarry\_r1250

**filename:** General\_BareSoilQuarry\_r1250.tif

**layername:** egv\_410

**English name:** Fractional cover of areas with Bare Soil, Quarries within the 1.25 km landscape

**Latvian name:** Atklātas augsnes un karjeru platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadri_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_BareSoilQuarry_cell.tif"),
  layer_prefixes = c("General_BareSoilQuarry"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",

```

```

fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# General_BareSoilQuarry_r1250.tif evg_410
slanis=rast("./RasterGrids_100m/2024/RAW/General_BareSoilQuarry_r1250.tif")
names(slanis)="evg_410"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_BareSoilQuarry_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_BareSoilQuarry_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.411 General\_BareSoilQuarry\_r3000

**filename:** General\_BareSoilQuarry\_r3000.tif

**layername:** evg\_411

**English name:** Fractional cover of areas with Bare Soil, Quarries within the 3 km landscape

**Latvian name:** Atklātas augsnes un karjeru platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_BareSoilQuarry_cell.tif"),
  layer_prefixes = c("General_BareSoilQuarry"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  
```

```

radii      = c("r3000"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# General_BareSoilQuarry_r3000.tif  egv_411
slanis=rast("./RasterGrids_100m/2024/RAW/General_BareSoilQuarry_r3000.tif")
names(slanis)="egv_411"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_BareSoilQuarry_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_BareSoilQuarry_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.4.12 General\_BareSoilQuarry\_r10000

**filename:** General\_BareSoilQuarry\_r10000.tif

**layername:** egv\_412

**English name:** Fractional cover of areas with Bare Soil, Quarries within the 10 km landscape

**Latvian name:** Atklātas augsnes un karjeru platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   = c("./RasterGrids_100m/2024/RAW/General_BareSoilQuarry_cell.tif"),

```

```

layer_prefixes = c("General_BareSoilQuarry"),
output_dir    = "./RasterGrids_100m/2024/RAW/",
n_workers     = 6,
radii         = c("r10000"),
radius_mode   = "sparse",
extract_fun   = "mean",
fill_missing   = TRUE,
IDW_weight    = 2,
future_max_size = 40 * 1024^3)

# General_BareSoilQuarry_r10000.tif egv_412
slanis=rast("./RasterGrids_100m/2024/RAW/General_BareSoilQuarry_r10000.tif")
names(slanis)="egv_412"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_BareSoilQuarry_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_BareSoilQuarry_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.413 General\_Builtup\_cell

**filename:** General\_Builtup\_cell.tif

**layername:** egv\_413

**English name:** Fractional cover of Built-Up areas within the analysis cell (1 ha)

**Latvian name:** Apbūves platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, the built-up areas from the [Landscape classification](#) are selected (value 500 is reclassified to value 1; all others are set to 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----

```

```

template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# General_Builtup_cell.tif evg_413 ----
builtup=ifel(simple_landscape==500,1,0)
i2e_rez=egvtools::input2egv(input=builtup,
                           egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                           summary_function = "average",
                           missing_job = "FillOutput",
                           outlocation = "./RasterGrids_100m/2024/RAW/",
                           outfilename = "General_Builtup_cell.tif",
                           layername = "evg_413",
                           idw_weight = 2,
                           plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(builtup)
rm(i2e_rez)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Builtup_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.414 General\_Builtup\_r500

**filename:** General\_Builtup\_r500.tif

**layername:** evg\_414

**English name:** Fractional cover of Built-Up areas within the 0.5 km landscape

**Latvian name:** Apbūves platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

```

```

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_Builtup_cell.tif"),
  layer_prefixes = c("General_Builtup"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_Builtup_r500.tif  egv_414
slanis=rast("./RasterGrids_100m/2024/RAW/General_Builtup_r500.tif")
names(slanis)="egv_414"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_Builtup_r500.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Builtup_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.415 General\_Builtup\_r1250

**filename:** General\_Builtup\_r1250.tif

**layername:** egv\_415

**English name:** Fractional cover of Built-Up areas within the 1.25 km landscape

**Latvian name:** Apbūves platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs -----
if(!require(terra)) {install.packages("terra"); require(terra)}

```

```

if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_Builtup_cell.tif"),
  layer_prefixes = c("General_Builtup"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# General_Builtup_r1250.tif egv_415
slanis=rast("./RasterGrids_100m/2024/RAW/General_Builtup_r1250.tif")
names(slanis)="egv_415"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_Builtup_r1250.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Builtup_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.416 General\_Builtup\_r3000

**filename:** General\_Builtup\_r3000.tif

**layername:** egv\_416

**English name:** Fractional cover of Built-Up areas within the 3 km landscape

**Latvian name:** Apbūves platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/General_Builtup_cell.tif"),
  layer_prefixes = c("General_Builtup"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_Builtup_r3000.tif egv_416
slanis=rast("./RasterGrids_100m/2024/Raw/General_Builtup_r3000.tif")
names(slanis)="egv_416"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/General_Builtup_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Builtup_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.417 General\_Builtup\_r10000

**filename:** General\_Builtup\_r10000.tif

**layername:** egv\_417

**English name:** Fractional cover of Built-Up areas within the 10 km landscape

**Latvian name:** Apbūves platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the

output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_Builtup_cell.tif"),
  layer_prefixes = c("General_Builtup"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_Builtup_r10000.tif      egv_417
slanis=rast("./RasterGrids_100m/2024/RAW/General_Builtup_r10000.tif")
names(slanis)="egv_417"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_Builtup_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Builtup_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.418 General\_Farmland\_cell

**filename:** General\_Farmland\_cell.tif

**layername:** egv\_418

**English name:** Fractional cover of Farmland within the analysis cell (1 ha)

**Latvian name:** Lauksaimniecībā izmantojamo zemju platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, the farmlands from the [Landscape classification](#) are selected (values between 300 and 400 are reclassified to value 1; all others are set to 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# General_Farmland_cell.tif egv_418 ----
farmland=ifel(simple_landscape>=300&simple_landscape<400,1,0)
i2e_rez=egvtools::input2egv(input=farmland,
                           egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                           summary_function = "average",
                           missing_job = "FillOutput",
                           outlocation = "./RasterGrids_100m/2024/RAW/",
                           outfilename = "General_Farmland_cell.tif",
                           layername = "egv_418",
                           idw_weight = 2,
                           plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(farmland)
rm(i2e_rez)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Farmland_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)
```

## 6.419 General\_Farmland\_r500

**filename:** General\_Farmland\_r500.tif

**layername:** evg\_419

**English name:** Fractional cover of Farmland within the 0.5 km landscape

**Latvian name:** Lauksaimniecībā izmantojamo zemju platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_Farmland_cell.tif"),
  layer_prefixes = c("General_Farmland"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_Farmland_r500.tif evg_419
slanis=rast("./RasterGrids_100m/2024/RAW/General_Farmland_r500.tif")
names(slanis)="evg_419"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_Farmland_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Farmland_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.420 General\_Farmland\_r1250

**filename:** General\_Farmland\_r1250.tif

**layername:** evg\_420

**English name:** Fractional cover of Farmland within the 1.25 km landscape

**Latvian name:** Lauksaimniecībā izmantojamo zemju platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_Farmland_cell.tif"),
  layer_prefixes = c("General_Farmland"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_Farmland_r1250.tif      evg_420
slanis=rast("./RasterGrids_100m/2024/RAW/General_Farmland_r1250.tif")
names(slanis)="evg_420"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_Farmland_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Farmland_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.421 General\_Farmland\_r3000

**filename:** General\_Farmland\_r3000.tif

**layername:** evg\_421

**English name:** Fractional cover of Farmland within the 3 km landscape

**Latvian name:** Lauksaimniecībā izmantojamo zemju platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_Farmland_cell.tif"),
  layer_prefixes = c("General_Farmland"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_Farmland_r3000.tif      evg_421
slanis=rast("./RasterGrids_100m/2024/RAW/General_Farmland_r3000.tif")
names(slanis)="evg_421"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_Farmland_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Farmland_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.422 General\_Farmland\_r10000

**filename:** General\_Farmland\_r10000.tif

**layername:** egv\_422

**English name:** Fractional cover of Farmland within the 10 km landscape

**Latvian name:** Lauksaimniecībā izmantojamo zemju platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_Farmland_cell.tif"),
  layer_prefixes = c("General_Farmland"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_Farmland_r10000.tif  egv_422
slanis=rast("./RasterGrids_100m/2024/RAW/General_Farmland_r10000.tif")
names(slanis)="egv_422"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_Farmland_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Farmland_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.423 General\_ForestsWithoutInventory\_cell

**filename:** General\_ForestsWithoutInventory\_cell.tif

**layername:** egv\_423

**English name:** Fractional cover of Forests Without Inventory within the analysis cell (1 ha)

**Latvian name:** Netaksēto mežu platības īpatsvars analīzē ūnā (1 ha)

**Procedure:** First, clearcuts and forest stands from the State Forest Service's State Forest Registry are rasterised to match inputs (presence as value 1; NA elsewhere). Then, from the Landscape classification class 630 is reclassified to value 1, others to 0). These layers are then combined so that values 1 from the second layer, where spatially matching NA values in the first layer and classified as 1; otherwise 0. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# General_ForestsWithoutInventory_cell.tif  egv_423 ----
mvr=st_read_parquet("./Geodata/2024/MVR/nogabali_2024janv.parquet")
tksetie=mvr %>%
  mutate(yes=1) %>%
  filter(zkat %in% c("10","12","14","16"))
taksetie_r=fasterize(tksetie,rastrs10,field="yes",fun="first")
taksetie_t=rast(taksetie_r)
visi_mezi=ifel(simple_landscape==630,1,0)
netaksetie=ifel(is.na(taksetie_t)&visi_mezi==1,1,0)
plot(netaksetie)

i2e_rez=egvtools::input2egv(input=netaksetie,
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/RAW/",
  outfilename = "General_ForestsWithoutInventory_cell.tif",
  layername = "egv_423",
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(netaksetie)
rm(i2e_rez)
rm(mvr)
rm(tksetie)
```

```

rm(taksetie_r)
rm(taksetie_t)
rm(visi_mezi)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_ForestsWithoutInventory_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.424 General\_ForestsWithoutInventory\_r500

**filename:** General\_ForestsWithoutInventory\_r500.tif

**layername:** egv\_424

**English name:** Fractional cover of Forests Without Inventory within the 0.5 km landscape

**Latvian name:** Netaksēto mežu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_ForestsWithoutInventory_cell.tif"),
  layer_prefixes = c("General_ForestsWithoutInventory"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_ForestsWithoutInventory_r500.tif egv_424

```

```

slanis=rast("./RasterGrids_100m/2024/RAW/General_ForestsWithoutInventory_r500.tif")
names(slanis)="egv_424"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_ForestsWithoutInventory_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_ForestsWithoutInventory_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.425 General\_ForestsWithoutInventory\_r1250

**filename:** General\_ForestsWithoutInventory\_r1250.tif

**layername:** egv\_425

**English name:** Fractional cover of Forests Without Inventory within the 1.25 km landscape

**Latvian name:** Netaksēto mežu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_ForestsWithoutInventory_cell.tif"),
  layer_prefixes = c("General_ForestsWithoutInventory"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

```

```

# General_ForestsWithoutInventory_r1250.tif evg_425
slanis=rast("./RasterGrids_100m/2024/RAW/General_ForestsWithoutInventory_r1250.tif")
names(slanis)="evg_425"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_ForestsWithoutInventory_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_ForestsWithoutInventory_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.426 General\_ForestsWithoutInventory\_r3000

**filename:** General\_ForestsWithoutInventory\_r3000.tif

**layername:** evg\_426

**English name:** Fractional cover of Forests Without Inventory within the 3 km landscape

**Latvian name:** Netaksēto mežu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadri_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_ForestsWithoutInventory_cell.tif"),
  layer_prefixes = c("General_ForestsWithoutInventory"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",

```

```

fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# General_ForestsWithoutInventory_r3000.tif egv_426
slanis=rast("./RasterGrids_100m/2024/RAW/General_ForestsWithoutInventory_r3000.tif")
names(slanis)="egv_426"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_ForestsWithoutInventory_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_ForestsWithoutInventory_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.427 General\_ForestsWithoutInventory\_r10000

**filename:** General\_ForestsWithoutInventory\_r10000.tif

**layername:** egv\_427

**English name:** Fractional cover of Forests Without Inventory within the 10 km landscape

**Latvian name:** Netaksēto mežu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_ForestsWithoutInventory_cell.tif"),
  layer_prefixes = c("General_ForestsWithoutInventory"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  
```

```

radii      = c("r10000",
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# General_ForestsWithoutInventory_r10000.tif    egv_427
slanis=rast("./RasterGrids_100m/2024/RAW/General_ForestsWithoutInventory_r10000.tif")
names(slanis)="egv_427"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_ForestsWithoutInventory_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_ForestsWithoutInventory_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.428 General\_GardensOrchards\_cell

**filename:** General\_GardensOrchards\_cell.tif

**layername:** egv\_428

**English name:** Fractional cover of Allotment gardens, Orchards within the analysis cell (1 ha)

**Latvian name:** Vasarnīcu kompleksu un augļudārzu platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, the allotment gardens and orchards from the [Landscape classification](#) are selected (values between 400 and 500 are reclassified to value 1; all others are set to 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

```

```

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# General_GardensOrchards_cell.tif evg_428 ----
parejie=ifel(simple_landscape>=400&simple_landscape<500,1,0)
i2e_rez=egvtools::input2egv(input=parejie,
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/RAW/",
  outfilename = "General_GardensOrchards_cell.tif",
  layername = "evg_428",
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(parejie)
rm(i2e_rez)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_GardensOrchards_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.429 General\_GardensOrchards\_r500

**filename:** General\_GardensOrchards\_r500.tif

**layername:** evg\_429

**English name:** Fractional cover of Allotment gardens, Orchards within the 0.5 km landscape

**Latvian name:** Vasarnīcu kompleksu un augļudārzu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

```

```

# radii ----
radius_function(
  kvadri_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/General_GardensOrchards_cell.tif"),
  layer_prefixes = c("General_GardensOrchards"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii        = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# General_GardensOrchards_r500.tif  egv_429
slanis=rast("./RasterGrids_100m/2024/Raw/General_GardensOrchards_r500.tif")
names(slanis)="egv_429"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/General_GardensOrchards_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_GardensOrchards_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.430 General\_GardensOrchards\_r1250

**filename:** General\_GardensOrchards\_r1250.tif

**layername:** egv\_430

**English name:** Fractional cover of Allotment gardens, Orchards within the 1.25 km landscape

**Latvian name:** Vasarnīcu kompleksu un augļudārzu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

```

```

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_GardensOrchards_cell.tif"),
  layer_prefixes = c("General_GardensOrchards"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_GardensOrchards_r1250.tif egv_430
slanis=rast("./RasterGrids_100m/2024/RAW/General_GardensOrchards_r1250.tif")
names(slanis)="egv_430"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_GardensOrchards_r1250.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_GardensOrchards_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.431 General\_GardensOrchards\_r3000

**filename:** General\_GardensOrchards\_r3000.tif

**layername:** egv\_431

**English name:** Fractional cover of Allotment gardens, Orchards within the 3 km landscape

**Latvian name:** Vasarnīcu kompleksu un augļudārzu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs -----
if(!require(terra)) {install.packages("terra"); require(terra)}

```

```

if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_GardensOrchards_cell.tif"),
  layer_prefixes = c("General_GardensOrchards"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# General_GardensOrchards_r3000.tif egv_431
slanis=rast("./RasterGrids_100m/2024/RAW/General_GardensOrchards_r3000.tif")
names(slanis)="egv_431"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_GardensOrchards_r3000.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_GardensOrchards_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.432 General\_GardensOrchards\_r10000

**filename:** General\_GardensOrchards\_r10000.tif

**layername:** egv\_432

**English name:** Fractional cover of Allotment gardens, Orchards within the 10 km landscape

**Latvian name:** Vasarnīcu kompleksu un augļudārzu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/General_GardensOrchards_cell.tif"),
  layer_prefixes = c("General_GardensOrchards"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_GardensOrchards_r10000.tif    egv_432
slanis=rast("./RasterGrids_100m/2024/Raw/General_GardensOrchards_r10000.tif")
names(slanis)="egv_432"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/General_GardensOrchards_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_GardensOrchards_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.433 General\_Roads\_cell

**filename:** General\_Roads\_cell.tif

**layername:** egv\_433

**English name:** Fractional cover of Roads within the analysis cell (1 ha)

**Latvian name:** Ceļu platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, the roads from the [Landscape classification](#) are selected (value 100 is reclassified to value 1; all others are set to 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During

aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# General_Roads_cell.tif    egv_433 ----
celi=ifel(simple_landscape==100,1,0)
i2e_rez=egvtools::input2egv(input=celi,
                           egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                           summary_function = "average",
                           missing_job = "FillOutput",
                           outlocation = "./RasterGrids_100m/2024/RAW/",
                           outfilename = "General_Roads_cell.tif",
                           layername = "egv_433",
                           idw_weight = 2,
                           plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(celi)
rm(i2e_rez)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Roads_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.434 General\_ShrubsOrchards\_cell

**filename:** General\_ShrubsOrchards\_cell.tif

**layername:** egv\_434

**English name:** Fractional cover of Shrubs, Young stands, Orchards within the analysis cell (1 ha)

**Latvian name:** Krūmāju, jaunaudžu un augļudārzu platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, agricultural parcels declared as short term coppice are selected from the [Rural Support Service's information on declared fields](#) and rasterised to match inputs. Then orchards and shrubs-low forest stands from [Landscape classification](#) are selected (values equal to 420 or 620 are reclassified to value 1, others as 0). The first layer is then covered over the second. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# General_ShrubsOrchards_cell.tif  egv_434 ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
table(kodi$SDM_grupa_sakums,useNA="always")
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))
ilggadigiekrumveida=lad %>%
  filter(SDM_grupa_sakums == "krūmveida ilggadīgie stādījumi")
krumveida_r=fasterize(ilggadigiekrumveida,rastrs10,field="yes",fun="first")
krumveida_t=rast(krumveida_r)
augludarzi=ifel(simple_landscape==420|simple_landscape==620,1,0)
apvienoti=cover(krumveida_t,augludarzi)
plot(apvienoti)

i2e_rez=egvtools::input2egv(input=apvienoti,
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/RAW/",
  outfilename = "General_ShrubsOrchards_cell.tif",
  layername = "egv_434",
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(apvienoti)
rm(i2e_rez)
rm(ilggadigiekrumveida)
rm(krumveida_r)
rm(krumveida_t)
```

```

rm(augludarzi)

rm(kodi)
rm(lad)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_ShrubsOrchards_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.435 General\_ShrubsOrchards\_r500

**filename:** General\_ShrubsOrchards\_r500.tif

**layername:** egv\_435

**English name:** Fractional cover of Shrubs, Young stands, Orchards within the 0.5 km landscape

**Latvian name:** Krūmāju, jaunaudžu un augļudārzu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_ShrubsOrchards_cell.tif"),
  layer_prefixes = c("General_ShrubsOrchards"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

```

```

# General_ShrubsOrchards_r500.tif    evg_435
slanis=rast("./RasterGrids_100m/2024/RAW/General_ShrubsOrchards_r500.tif")
names(slanis)="evg_435"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_ShrubsOrchards_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_ShrubsOrchards_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.436 General\_ShrubsOrchards\_r1250

**filename:** General\_ShrubsOrchards\_r1250.tif

**layername:** evg\_436

**English name:** Fractional cover of Shrubs, Young stands, Orchards within the 1.25 km landscape

**Latvian name:** Krūmāju, jaunaudžu un augļudārzu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_ShrubsOrchards_cell.tif"),
  layer_prefixes = c("General_ShrubsOrchards"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,

```

```

future_max_size = 40 * 1024^3)

# General_ShrubsOrchards_r1250.tif  egv_436
slanis=rast("./RasterGrids_100m/2024/RAW/General_ShrubsOrchards_r1250.tif")
names(slanis)="egv_436"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_ShrubsOrchards_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_ShrubsOrchards_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.437 General\_ShrubsOrchards\_r3000

**filename:** General\_ShrubsOrchards\_r3000.tif

**layername:** egv\_437

**English name:** Fractional cover of Shrubs, Young stands, Orchards within the 3 km landscape

**Latvian name:** Krūmāju, jaunaudžu un augļudārzu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_ShrubsOrchards_cell.tif"),
  layer_prefixes = c("General_ShrubsOrchards"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",

```

```

extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# General_ShrubsOrchards_r3000.tif egv_437
slanis=rast("./RasterGrids_100m/2024/RAW/General_ShrubsOrchards_r3000.tif")
names(slanis)="egv_437"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_ShrubsOrchards_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_ShrubsOrchards_r3000.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.438 General\_ShrubsOrchards\_r10000

**filename:** General\_ShrubsOrchards\_r10000.tif

**layername:** egv\_438

**English name:** Fractional cover of Shrubs, Young stands, Orchards within the 10 km landscape

**Latvian name:** Krūmāju, jaunaudžu un augļudārzu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_ShrubsOrchards_cell.tif"),
  layer_prefixes = c("General_ShrubsOrchards"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",

```

```

n_workers    = 6,
radii        = c("r10000"),
radius_mode  = "sparse",
extract_fun   = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3)

# General_ShrubsOrchards_r10000.tif egv_438
slanis=rast("./RasterGrids_100m/2024/RAW/General_ShrubsOrchards_r10000.tif")
names(slanis)="egv_438"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_ShrubsOrchards_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_ShrubsOrchards_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.439 General\_ShrubsOrchardsGardens\_cell

**filename:** General\_ShrubsOrchardsGardens\_cell.tif

**layername:** egv\_439

**English name:** Fractional cover of Shrubs, Young stands, Orchards, Allotment gardens within the analysis cell (1 ha)

**Latvian name:** Krūmāju, jaunaudžu, augļudārzu un vasarnīcu kompleksu platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, agricultural parcels declared as short term coppice are selected from the [Rural Support Service's information on declared fields](#) and rasterised to match inputs. Then orchards, allotment gardens and shrubs-low forest stands from the [Landscape classification](#) are selected (values between 400 and 500 or equal to 620 are reclassified to value 1, others as 0). The first layer is then covered over the second. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

```

```

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# General_ShrubsOrchardsGardens_cell.tif      evg_439 ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
table(kodi$SDM_grupa_sakums,useNA="always")
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))
ilggadigiekrumveida=lad %>%
  filter(SDM_grupa_sakums == "krūmveida ilggadīgie stādijumi")
krumveida_r=fasterize(ilggadigiekrumveida,rastrs10,field="yes",fun="first")
krumveida_t=rast(krumveida_r)
parejie=ifel((simple_landscape>=400&simple_landscape<500) |
  simple_landscape==620,1,0)
apvienoti=cover(krumveida_t,parejie)
plot(apvienoti)

i2e_rez=egvtools::input2egv(input=apvienoti,
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/RAW/",
  outfilename = "General_ShrubsOrchardsGardens_cell.tif",
  layername = "evg_439",
  idw_weight = 2,
  plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(apvienoti)
rm(i2e_rez)
rm(ilggadigiekrumveida)
rm(krumveida_r)
rm(krumveida_t)
rm(parejie)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_ShrubsOrchardsGardens_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.440 General\_ShrubsOrchardsGardens\_r500

**filename:** General\_ShrubsOrchardsGardens\_r500.tif

**layername:** egv\_440

**English name:** Fractional cover of Shrubs, Young stands, Orchards, Allotment gardens within the 0.5 km landscape

**Latvian name:** Krūmāju, jaunaudžu, augļudārzu un vasarnīcu kompleksu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_ShrubsOrchardsGardens_cell.tif"),
  layer_prefixes = c("General_ShrubsOrchardsGardens"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_ShrubsOrchardsGardens_r500.tif    egv_440
slanis=rast("./RasterGrids_100m/2024/RAW/General_ShrubsOrchardsGardens_r500.tif")
names(slanis)="egv_440"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_ShrubsOrchardsGardens_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_ShrubsOrchardsGardens_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
```

```
overwrite=TRUE)
```

## 6.441 General\_ShrubsOrchardsGardens\_r1250

**filename:** General\_ShrubsOrchardsGardens\_r1250.tif

**layername:** egv\_441

**English name:** Fractional cover of Shrubs, Young stands, Orchards, Allotment gardens within the 1.25 km landscape

**Latvian name:** Krūmāju, jaunaudžu, augļudārzu un vasarnīcu kompleksu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_Shru
  bsOrchardsGardens_cell.tif"),
  layer_prefixes = c("General_ShrubsOrchardsGardens"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# General_ShrubsOrchardsGardens_r1250.tif  egv_441
slanis=rast("./RasterGrids_100m/2024/RAW/General_ShrubsOrchardsGardens_r1250.tif")
names(slanis)="egv_441"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_ShrubsOrchardsGardens_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_ShrubsOrchardsGardens_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
```

```

centrets=slanis$videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.442 General\_ShrubsOrchardsGardens\_r3000

**filename:** General\_ShubsOrchardsGardens\_r3000.tif

**layername:** egv\_442

**English name:** Fractional cover of Shrubs, Young stands, Orchards, Allotment gardens within the 3 km landscape

**Latvian name:** Krūmāju, jaunaudžu, augļudārzu un vasarnīcu kompleksu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/General_ShubsOrchardsGardens_cell.tif"),
  layer_prefixes = c("General_ShubsOrchardsGardens"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# General_ShubsOrchardsGardens_r3000.tif  egv_442
slanis=rast("./RasterGrids_100m/2024/Raw/General_ShubsOrchardsGardens_r3000.tif")
names(slanis)="egv_442"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/General_ShubsOrchardsGardens_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

nosaukums="General_ShrubsOrchardsGardens_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.443 General\_ShrubsOrchardsGardens\_r10000

**filename:** General\_ShrubsOrchardsGardens\_r10000.tif

**layername:** egv\_443

**English name:** Fractional cover of Shrubs, Young stands, Orchards, Allotment gardens within the 10 km landscape

**Latvian name:** Krūmāju, jaunaudžu, augļudārzu un vasarnīcu kompleksu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_ShrubsOrchardsGardens_cell.tif"),
  layer_prefixes = c("General_ShrubsOrchardsGardens"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# General_ShrubsOrchardsGardens_r10000.tif egv_443
slanis=rast("./RasterGrids_100m/2024/RAW/General_ShrubsOrchardsGardens_r10000.tif")
names(slanis)="egv_443"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_ShrubsOrchardsGardens_r10000.tif",
  overwrite=TRUE)

```

```

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_ShrubsOrchardsGardens_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovidze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovidze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.444 General\_SwampsMiresBogsHelophytes\_cell

**filename:** General\_SwampsMiresBogsHelophytes\_cell.tif

**layername:** egv\_444

**English name:** Fractional cover of Swamps, Mires, Bogs, Reed-, Sedge-, Rush- Beds within the analysis cell (1 ha)

**Latvian name:** Purvu, niedrāju, grīslāju, meldrāju platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, the swamps, mires, bogs and reed, sedge, rush beds from the [Landscape classification](#) are selected (values between 700 and 800 are reclassified to value 1; all others are set to 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# General_SwampsMiresBogsHelophytes_cell.tif    egv_444 ----
purvi=ifel(simple_landscape>=700&simple_landscape<800,1,0)
i2e_rez=egvtools::input2egv(input=purvi,
    egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
    summary_function = "average",
    missing_job = "FillOutput",
    
```

```

        outlocation = "./RasterGrids_100m/2024/RAW/",
        outfilename = "General_SwampsMiresBogsHelophytes_cell.tif",
        layername = "egv_444",
        idw_weight = 2,
        plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(purvi)
rm(i2e_rez)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_SwampsMiresBogsHelophytes_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.445 General\_SwampsMiresBogsHelophytes\_r500

**filename:** General\_SwampsMiresBogsHelophytes\_r500.tif

**layername:** egv\_445

**English name:** Fractional cover of Swamps, Mires, Bogs, Reed-, Sedge-, Rush- Beds within the 0.5 km landscape

**Latvian name:** Purvu, niedrāju, grīslāju, meldrāju platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_SwampsMiresBogsHelophytes_cell.tif"),
  layer_prefixes = c("General_SwampsMiresBogsHelophytes"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",

```

```

extract_fun  = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3)

# General_SwampsMiresBogsHelophytes_r500.tif    evg_445
slanis=rast("./RasterGrids_100m/2024/RAW/General_SwampsMiresBogsHelophytes_r500.tif")
names(slanis)="evg_445"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_SwampsMiresBogsHelophytes_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_SwampsMiresBogsHelophytes_r500.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.446 General\_SwampsMiresBogsHelophytes\_r1250

**filename:** General\_SwampsMiresBogsHelophytes\_r1250.tif

**layername:** evg\_446

**English name:** Fractional cover of Swamps, Mires, Bogs, Reed-, Sedge-, Rush- Beds within the 1.25 km landscape

**Latvian name:** Purvu, niedrāju, grīslāju, meldrāju platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadрати_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_SwampsMiresBogsHelophytes_cell.tif"),

```

```

layer_prefixes = c("General_SwampsMiresBogsHelophytes"),
output_dir    = "./RasterGrids_100m/2024/RAW/",
n_workers     = 6,
radii         = c("r1250"),
radius_mode   = "sparse",
extract_fun   = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3)

# General_SwampsMiresBogsHelophytes_r1250.tif  egv_446
slanis=rast("./RasterGrids_100m/2024/RAW/General_SwampsMiresBogsHelophytes_r1250.tif")
names(slanis)="egv_446"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_SwampsMiresBogsHelophytes_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_SwampsMiresBogsHelophytes_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.447 General\_SwampsMiresBogsHelophytes\_r3000

**filename:** General\_SwampsMiresBogsHelophytes\_r3000.tif

**layername:** egv\_447

**English name:** Fractional cover of Swamps, Mires, Bogs, Reed-, Sedge-, Rush- Beds within the 3 km landscape

**Latvian name:** Purvu, niedrāju, grīslāju, meldrāju platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(

```

```

kvadrati_path = "./Templates/TemplateGrids/tiles/",
radii_path    = "./Templates/TemplateGridPoints/tiles/",
tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers  = c("./RasterGrids_100m/2024/RAW/General_SwampsMiresBogsHelophytes_cell.tif"),
layer_prefixes = c("General_SwampsMiresBogsHelophytes"),
output_dir   = "./RasterGrids_100m/2024/RAW/",
n_workers    = 6,
radii        = c("r3000"),
radius_mode  = "sparse",
extract_fun  = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 40 * 1024^3

# General_SwampsMiresBogsHelophytes_r3000.tif  egv_447
slanis=rast("./RasterGrids_100m/2024/RAW/General_SwampsMiresBogsHelophytes_r3000.tif")
names(slanis)="egv_447"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_SwampsMiresBogsHelophytes_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_SwampsMiresBogsHelophytes_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.448 General\_SwampsMiresBogsHelophytes\_r10000

**filename:** General\_SwampsMiresBogsHelophytes\_r10000.tif

**layername:** egv\_448

**English name:** Fractional cover of Swamps, Mires, Bogs, Reed-, Sedge-, Rush- Beds within the 10 km landscape

**Latvian name:** Purvu, niedrāju, grīslāju, meldrāju platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

```

```

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii -----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_SwampsMiresBogsHelophytes_cell.tif"),
  layer_prefixes = c("General_SwampsMiresBogsHelophytes"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_SwampsMiresBogsHelophytes_r10000.tif egv_448
slanis=rast("./RasterGrids_100m/2024/RAW/General_SwampsMiresBogsHelophytes_r10000.tif")
names(slanis)="egv_448"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_SwampsMiresBogsHelophytes_r10000.tif",
  overwrite=TRUE)

# standardisation -----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_SwampsMiresBogsHelophytes_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.449 General\_Trees\_cell

**filename:** General\_Trees\_cell.tif

**layername:** egv\_449

**English name:** Fractional cover of Trees, Shrubs, Clear-cuts within the analysis cell (1 ha)

**Latvian name:** Koku, krūmu un izcirtumu platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, the trees, shrubs and clear cuts from the [Landscape classification](#) are selected (values between 600 and 700 are reclassified to value 1; all others are set to 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs -----
```

```

if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# General_Trees_cell.tif    egv_449 ----
kokimezi=ifel(simple_landscape>=600&simple_landscape<700,1,0)
i2e_rez=egvtools::input2egv(input=kokimezi,
                           egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                           summary_function = "average",
                           missing_job = "FillOutput",
                           outlocation = "./RasterGrids_100m/2024/RAW/",
                           outfilename = "General_Trees_cell.tif",
                           layername = "egv_449",
                           idw_weight = 2,
                           plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(kokimezi)
rm(i2e_rez)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Trees_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.450 General\_Trees\_r500

**filename:** General\_Trees\_r500.tif

**layername:** egv\_450

**English name:** Fractional cover of Trees, Shrubs, Clear-cuts within the 0.5 km landscape

**Latvian name:** Koku, krūmu un izcirtumu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the

output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_Trees_cell.tif"),
  layer_prefixes = c("General_Trees"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_Trees_r500.tif    egv_450
slanis=rast("./RasterGrids_100m/2024/RAW/General_Trees_r500.tif")
names(slanis)="egv_450"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_Trees_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Trees_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.451 General\_Trees\_r1250

**filename:** General\_Trees\_r1250.tif

**layername:** egv\_451

**English name:** Fractional cover of Trees, Shrubs, Clear-cuts within the 1.25 km landscape

**Latvian name:** Koku, krūmu un izcirtumu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_Trees_cell.tif"),
  layer_prefixes = c("General_Trees"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_Trees_r1250.tif  egv_451
slanis=rast("./RasterGrids_100m/2024/RAW/General_Trees_r1250.tif")
names(slanis)="egv_451"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_Trees_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Trees_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.452 General\_Trees\_r3000

**filename:** General\_Trees\_r3000.tif

**layername:** egv\_452

**English name:** Fractional cover of Trees, Shrubs, Clear-cuts within the 3 km landscape

**Latvian name:** Koku, krūmu un izcirtumu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes:::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path  = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   = c("./RasterGrids_100m/2024/Raw/General_Trees_cell.tif"),
  layer_prefixes = c("General_Trees"),
  output_dir     = "./RasterGrids_100m/2024/Raw/",
  n_workers      = 6,
  radii          = c("r3000"),
  radius_mode    = "sparse",
  extract_fun    = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# General_Trees_r3000.tif  egv_452
slanis=rast("./RasterGrids_100m/2024/Raw/General_Trees_r3000.tif")
names(slanis)="egv_452"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/Raw/General_Trees_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Trees_r3000.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.453 General\_Trees\_r10000

**filename:** General\_Trees\_r10000.tif

**layername:** egv\_453

**English name:** Fractional cover of Trees, Shrubs, Clear-cuts within the 10 km landscape

**Latvian name:** Koku, krūmu un izcirtumu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_Trees_cell.tif"),
  layer_prefixes = c("General_Trees"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii        = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_Trees_r10000.tif egv_453
slanis=rast("./RasterGrids_100m/2024/RAW/General_Trees_r10000.tif")
names(slanis)="egv_453"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_Trees_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Trees_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.454 General\_TreesOutsideForests\_cell

**filename:** General\_TreesOutsideForests\_cell.tif

**layername:** egv\_454

**English name:** Fractional cover of Tree covered areas Outside Forests within the analysis cell (1 ha)

**Latvian name:** Ar kokiem klāto teritoriju ārpus mežiem platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, the tree covered areas outside forest stands from the [Landscape classification](#) are selected (value 640 is reclassified to value 1; all others are set to 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# General_TreesOutsideForests_cell.tif egv_454 ----
kokiarpulse=ifel(simple_landscape==640,1,0)
i2e_rez=egvtools::input2egv(input=kokiarpulse,
                           egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                           summary_function = "average",
                           missing_job = "FillOutput",
                           outlocation = "./RasterGrids_100m/2024/RAW/",
                           outfile = "General_TreesOutsideForests_cell.tif",
                           layername = "egv_454",
                           idw_weight = 2,
                           plot_gaps = FALSE,plot_final = TRUE)

i2e_rez
rm(kokiarpulse)
rm(i2e_rez)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_TreesOutsideForests_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
```

```

writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.455 General\_TreesOutsideForests\_r500

**filename:** General\_TreesOutsideForests\_r500.tif

**layername:** egv\_455

**English name:** Fractional cover of Tree covered areas Outside Forests within the 0.5 km landscape

**Latvian name:** Ar kokiem klāto teritoriju ārpus mežiem platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_TreesOutsideForests_cell.tif"),
  layer_prefixes = c("General_TreesOutsideForests"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_TreesOutsideForests_r500.tif  egv_455
slanis=rast("./RasterGrids_100m/2024/RAW/General_TreesOutsideForests_r500.tif")
names(slanis)="egv_455"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_TreesOutsideForests_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_TreesOutsideForests_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)

```

```

centrets=slanis$videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.456 General\_TreesOutsideForests\_r1250

**filename:** General\_TreesOutsideForests\_r1250.tif

**layername:** evg\_456

**English name:** Fractional cover of Tree covered areas Outside Forests within the 1.25 km landscape

**Latvian name:** Ar kokiem klāto teritoriju ārpus mežiem platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path   = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers    = c("./RasterGrids_100m/2024/RAW/General_TreesOutsideForests_cell.tif"),
  layer_prefixes = c("General_TreesOutsideForests"),
  output_dir     = "./RasterGrids_100m/2024/RAW/",
  n_workers      = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing  = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_TreesOutsideForests_r1250.tif evg_456
slanis=rast("./RasterGrids_100m/2024/RAW/General_TreesOutsideForests_r1250.tif")
names(slanis)="evg_456"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_TreesOutsideForests_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_TreesOutsideForests_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)

```

```

saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.457 General\_TreesOutsideForests\_r3000

**filename:** General\_TreesOutsideForests\_r3000.tif

**layername:** evg\_457

**English name:** Fractional cover of Tree covered areas Outside Forests within the 3 km landscape

**Latvian name:** Ar kokiem klāto teritoriju ārpus mežiem platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/General_TreesOutsideForests_cell.tif"),
  layer_prefixes = c("General_TreesOutsideForests"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 6,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 40 * 1024^3)

# General_TreesOutsideForests_r3000.tif evg_457
slanis=rast("./RasterGrids_100m/2024/Raw/General_TreesOutsideForests_r3000.tif")
names(slanis)="evg_457"
slanis2=project(slanis,template100)
writeRaster(slanis2,
    "./RasterGrids_100m/2024/Raw/General_TreesOutsideForests_r3000.tif",
    overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

nosaukums="General_TreesOutsideForests_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.458 General\_TreesOutsideForests\_r10000

**filename:** General\_TreesOutsideForests\_r10000.tif

**layername:** egv\_458

**English name:** Fractional cover of Tree covered areas Outside Forests within the 10 km landscape

**Latvian name:** Ar kokiem klāto teritoriju ārpus mežiem platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadri_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_TreesOutsideForests_cell.tif"),
  layer_prefixes = c("General_TreesOutsideForests"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

# General_TreesOutsideForests_r10000.tif    egv_458
slanis=rast("./RasterGrids_100m/2024/RAW/General_TreesOutsideForests_r10000.tif")
names(slanis)="egv_458"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_TreesOutsideForests_r10000.tif",
  overwrite=TRUE)

```

```

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_TreesOutsideForests_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.459 General\_Water\_cell

**filename:** General\_Water\_cell.tif

**layername:** egv\_459

**English name:** Fractional cover of Waterbodies within the analysis cell (1 ha)

**Latvian name:** Ūdenstilpju platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, the waters from the [Landscape classification](#) are selected (value 200 is reclassified to value 1; all others are set to 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}
if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# General_Water_cell.tif    egv_459 ----
udens=ifel(simple_landscape==200,1,0)
i2e_rez=egvtools::input2egv(input=udens,
                            egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                            summary_function = "average",
                            missing_job = "FillOutput",
                            outlocation = "./RasterGrids_100m/2024/RAW/",
                            outfilename = "General_Water_cell.tif",

```

```

        layername = "egv_459",
        idw_weight = 2,
        plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(udens)
rm(i2e_rez)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Water_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.460 General\_Water\_r500

**filename:** General\_Water\_r500.tif

**layername:** egv\_460

**English name:** Fractional cover of Waterbodies within the 0.5 km landscape

**Latvian name:** Ūdenstilpju platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_Water_cell.tif"),
  layer_prefixes = c("General_Water"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 40 * 1024^3)

```

```

# General_Water_r500.tif      egv_460
slanis=rast("./RasterGrids_100m/2024/RAW/General_Water_r500.tif")
names(slanis)="egv_460"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_Water_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Water_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.461 General\_Water\_r1250

**filename:** General\_Water\_r1250.tif

**layername:** egv\_461

**English name:** Fractional cover of Waterbodies within the 1.25 km landscape

**Latvian name:** Ūdenstilpu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadri_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_Water_cell.tif"),
  layer_prefixes = c("General_Water"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",

```

```

fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# General_Water_r1250.tif  egv_461
slanis=rast("./RasterGrids_100m/2024/RAW/General_Water_r1250.tif")
names(slanis)="egv_461"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/General_Water_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Water_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.462 General\_Water\_r3000

**filename:** General\_Water\_r3000.tif

**layername:** egv\_462

**English name:** Fractional cover of Waterbodies within the 3 km landscape

**Latvian name:** Ūdenstilpu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/General_Water_cell.tif"),
  layer_prefixes = c("General_Water"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 6,

```

```

radii      = c("r3000"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 40 * 1024^3)

# General_Water_r3000.tif  egv_462
slanis=rast("./RasterGrids_100m/2024/RAW/General_Water_r3000.tif")
names(slanis)="egv_462"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_Water_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Water_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.463 General\_Water\_r10000

**filename:** General\_Water\_r10000.tif

**layername:** egv\_463

**English name:** Fractional cover of Waterbodies within the 10 km landscape

**Latvian name:** Ūdenstilpju platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   = c("./RasterGrids_100m/2024/RAW/General_Water_cell.tif"),

```

```

layer_prefixes = c("General_Water"),
output_dir    = "./RasterGrids_100m/2024/RAW/",
n_workers     = 6,
radii         = c("r10000"),
radius_mode   = "sparse",
extract_fun   = "mean",
fill_missing   = TRUE,
IDW_weight    = 2,
future_max_size = 40 * 1024^3)

# General_Water_r10000.tif  egv_463
slanis=rast("./RasterGrids_100m/2024/RAW/General_Water_r10000.tif")
names(slanis)="egv_463"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/General_Water_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="General_Water_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.464 Wetlands\_Bogs\_cell

**filename:** Wetlands\_Bogs\_cell.tif

**layername:** egv\_464

**English name:** Fractional cover of Raised Bogs within the analysis cell (1 ha)

**Latvian name:** Augsto purvu platības īpatsvars analīzē ūnā (1 ha)

**Procedure:** Derived from the [Bogs and Mires: EDI](#), where bogs are classified as 1 with 0 elsewhere. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}

# template ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# Wetlands_Bogs_cell.tif    egv_464 ----
bogs=rast("./RasterGrids_10m/2024/EDI_BogsYN.tif")
i2e_rez=egvtools::input2egv(input=bogs,
                            egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                            summary_function = "average",

```

```

        missing_job = "FillOutput",
        outlocation = "./RasterGrids_100m/2024/RAW/",
        outfilename = "Wetlands_Bogs_cell.tif",
        layername = "egv_464",
        idw_weight = 2,
        plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(bogs)
rm(i2e_rez)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Wetlands_Bogs_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.465 Wetlands\_Bogs\_r500

**filename:** Wetlands\_Bogs\_r500.tif

**layername:** egv\_465

**English name:** Fractional cover of Raised Bogs within the 0.5 km landscape

**Latvian name:** Augsto purvu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadri_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Wetlands_Bogs_cell.tif"),
  layer_prefixes = c("Wetlands_Bogs"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",

```

```

fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 20 * 1024^3)

# Wetlands_Bogs_r500.tif    egv_465
slanis=rast("./RasterGrids_100m/2024/RAW/Wetlands_Bogs_r500.tif")
names(slanis)="egv_465"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Wetlands_Bogs_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Wetlands_Bogs_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.466 Wetlands\_Bogs\_r1250

**filename:** Wetlands\_Bogs\_r1250.tif

**layername:** egv\_466

**English name:** Fractional cover of Raised Bogs within the 1.25 km landscape

**Latvian name:** Augsto purvu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Wetlands_Bogs_cell.tif"),
  layer_prefixes = c("Wetlands_Bogs"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,

```

```

radii      = c("r1250"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 20 * 1024^3)

# Wetlands_Bogs_r1250.tif  egv_466
slanis=rast("./RasterGrids_100m/2024/RAW/Wetlands_Bogs_r1250.tif")
names(slanis)="egv_466"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Wetlands_Bogs_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Wetlands_Bogs_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.467 Wetlands\_Bogs\_r3000

**filename:** Wetlands\_Bogs\_r3000.tif

**layername:** egv\_467

**English name:** Fractional cover of Raised Bogs within the 3 km landscape

**Latvian name:** Augsto purvu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   = c("./RasterGrids_100m/2024/RAW/Wetlands_Bogs_cell.tif"),

```

```

layer_prefixes = c("Wetlands_Bogs"),
output_dir    = "./RasterGrids_100m/2024/RAW/",
n_workers     = 12,
radii         = c("r3000"),
radius_mode   = "sparse",
extract_fun   = "mean",
fill_missing   = TRUE,
IDW_weight    = 2,
future_max_size = 20 * 1024^3)

# Wetlands_Bogs_r3000.tif  egv_467
slanis=rast("./RasterGrids_100m/2024/RAW/Wetlands_Bogs_r3000.tif")
names(slanis)="egv_467"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Wetlands_Bogs_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Wetlands_Bogs_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.468 Wetlands\_Bogs\_r10000

**filename:** Wetlands\_Bogs\_r10000.tif

**layername:** egv\_468

**English name:** Fractional cover of Raised Bogs within the 10 km landscape

**Latvian name:** Augsto purvu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",

```

```

tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers = c("./RasterGrids_100m/2024/RAW/Wetlands_Bogs_cell.tif"),
layer_prefixes = c("Wetlands_Bogs"),
output_dir = "./RasterGrids_100m/2024/RAW/",
n_workers = 12,
radii = c("r10000"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 20 * 1024^3)

# Wetlands_Bogs_r10000.tif evg_468
slanis=rast("./RasterGrids_100m/2024/RAW/Wetlands_Bogs_r10000.tif")
names(slanis)="egv_468"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Wetlands_Bogs_r10000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Wetlands_Bogs_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.469 Wetlands\_Mires\_cell

**filename:** Wetlands\_Mires\_cell.tif

**layername:** egv\_469

**English name:** Fractional cover of Transitional Mires within the analysis cell (1 ha)

**Latvian name:** Pārejas purvu platības īpatsvars analīzē šūnā (1 ha)

**Procedure:** Derived from the [Bogs and Mires: EDI](#), where transitional mires are classified as 1 with 0 elsewhere. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}

# template ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# Wetlands_Mires_cell.tif  evg_469 ----
mires=rast("./RasterGrids_10m/2024/EDI_TransitionalMiresYN.tif")

```

```

i2e_rez=egvtools::input2egv(input=mires,
                           egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                           summary_function = "average",
                           missing_job = "FillOutput",
                           outlocation = "./RasterGrids_100m/2024/Raw/",
                           outfilename = "Wetlands_Mires_cell.tif",
                           layername = "egv_469",
                           idw_weight = 2,
                           plot_gaps = FALSE,plot_final = TRUE)
rm(mires)
rm(i2e_rez)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Wetlands_Mires_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.470 Wetlands\_Mires\_r500

**filename:** Wetlands\_Mires\_r500.tif

**layername:** egv\_470

**English name:** Fractional cover of Transitional Mires within the 0.5 km landscape

**Latvian name:** Pārejas purvu platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/Raw/Wetlands_Mires_cell.tif"),
  layer_prefixes = c("Wetlands_Mires"),
  output_dir    = "./RasterGrids_100m/2024/Raw/",
  n_workers     = 12,

```

```

radii      = c("r500"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 20 * 1024^3)

# Wetlands_Mires_r500.tif  egv_470
slanis=rast("./RasterGrids_100m/2024/RAW/Wetlands_Mires_r500.tif")
names(slanis)="egv_470"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Wetlands_Mires_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Wetlands_Mires_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.471 Wetlands\_Mires\_r1250

**filename:** Wetlands\_Mires\_r1250.tif

**layername:** egv\_471

**English name:** Fractional cover of Transitional Mires within the 1.25 km landscape

**Latvian name:** Pārejas purvu platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Wetlands_Mires_cell.tif"),

```

```

layer_prefixes = c("Wetlands_Mires"),
output_dir    = "./RasterGrids_100m/2024/RAW/",
n_workers     = 12,
radii         = c("r1250"),
radius_mode   = "sparse",
extract_fun   = "mean",
fill_missing   = TRUE,
IDW_weight    = 2,
future_max_size = 20 * 1024^3)

# Wetlands_Mires_r1250.tif  egv_471
slanis=rast("./RasterGrids_100m/2024/RAW/Wetlands_Mires_r1250.tif")
names(slanis)="egv_471"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Wetlands_Mires_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Wetlands_Mires_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.472 Wetlands\_Mires\_r3000

**filename:** Wetlands\_Mires\_r3000.tif

**layername:** egv\_472

**English name:** Fractional cover of Transitional Mires within the 3 km landscape

**Latvian name:** Pārejas purvu platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",

```

```

tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers = c("./RasterGrids_100m/2024/RAW/Wetlands_Mires_cell.tif"),
layer_prefixes = c("Wetlands_Mires"),
output_dir = "./RasterGrids_100m/2024/RAW/",
n_workers = 12,
radii = c("r3000"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 20 * 1024^3)

# Wetlands_Mires_r3000.tif evg_472
slanis=rast("./RasterGrids_100m/2024/RAW/Wetlands_Mires_r3000.tif")
names(slanis)="evg_472"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Wetlands_Mires_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Wetlands_Mires_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.473 Wetlands\_Mires\_r10000

**filename:** Wetlands\_Mires\_r10000.tif

**layername:** evg\_473

**English name:** Fractional cover of Transitional Mires within the 10 km landscape

**Latvian name:** Pārejas purvu platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates -----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----

```

```

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Wetlands_Mires_cell.tif"),
  layer_prefixes = c("Wetlands_Mires"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Wetlands_Mires_r10000.tif egv_473
slanis=rast("./RasterGrids_100m/2024/RAW/Wetlands_Mires_r10000.tif")
names(slanis)="egv_473"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Wetlands_Mires_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Wetlands_Mires_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.474 Wetlands\_ReedSedgeRushBeds\_cell

**filename:** Wetlands\_ReedSedgeRushBeds\_cell.tif

**layername:** egv\_474

**English name:** Fractional cover of Reed-, Sedge-, Rush-, Beds within the analysis cell (1 ha)

**Latvian name:** Niedrāju, grīslāju, meldrāju platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** First, the reed, sedge and rush beds from the [Landscape classification](#) are selected (value 720 is reclassified to value 1; all others are set to 0). The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(sf)) {install.packages("sf"); require(sf)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

if(!require(sfarrow)) {install.packages("sfarrow"); require(sfarrow)}
if(!require(readxl)) {install.packages("readxl"); require(readxl)}
if(!require(raster)) {install.packages("raster"); require(raster)}
if(!require(fasterize)) {install.packages("fasterize"); require(fasterize)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
rastrs10=raster(template10)

nulls10=rast("./Templates/TemplateRasters/nulls_LV10m_10km.tif")
nulls100=rast("./Templates/TemplateRasters/nulls_LV100m_10km.tif")

# codes ----
kodi=read_excel("./Geodata/2024/LAD/KulturuKodi_2024.xlsx")
kodi$kods=as.character(kodi$kods)
# LAD ----
lad=sfarrow::st_read_parquet("./Geodata/2024/LAD/Lauki_2024.parquet")
lad$yes=1
lad=lad %>%
  left_join(kodi,by=c("PRODUCT_CODE"="kods"))

# simple landscape ----
simple_landscape=rast("RasterGrids_10m/2024/Ainava_vienk_mask.tif")

# Wetlands_ReedSedgeRushBeds_cell.tif    egv_474 ----
reedsedgerush=ifel(simple_landscape==720,1,0)

i2e_rez=egvtools::input2egv(input=reedsedgerush,
                           egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                           summary_function = "average",
                           missing_job = "FillOutput",
                           outlocation = "./RasterGrids_100m/2024/RAW/",
                           outfilename = "Wetlands_ReedSedgeRushBeds_cell.tif",
                           layername = "egv_474",
                           idw_weight = 2,
                           plot_gaps = FALSE,plot_final = TRUE)
i2e_rez
rm(reedsedgerush)
rm(i2e_rez)
rm(simple_landscape)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Wetlands_ReedSedgeRushBeds_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.475 Wetlands\_ReedSedgeRushBeds\_r500

**filename:** Wetlands\_ReedSedgeRushBeds\_r500.tif

**layername:** egv\_475

**English name:** Fractional cover of Reed-, Sedge-, Rush-, Beds within the 0.5 km landscape

**Latvian name:** Niedrāju, grīslāju, meldrāju platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes:::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Wetlands_ReedSedgeRushBeds_cell.tif"),
  layer_prefixes = c("Wetlands_ReedSedgeRushBeds"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Wetlands_ReedSedgeRushBeds_r500.tif  egv_475
slanis=rast("./RasterGrids_100m/2024/RAW/Wetlands_ReedSedgeRushBeds_r500.tif")
names(slanis)="egv_475"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Wetlands_ReedSedgeRushBeds_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Wetlands_ReedSedgeRushBeds_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.476 Wetlands\_ReedSedgeRushBeds\_r1250

**filename:** Wetlands\_ReedSedgeRushBeds\_r1250.tif

**layername:** egv\_476

**English name:** Fractional cover of Reed-, Sedge-, Rush-, Beds within the 1.25 km landscape

**Latvian name:** Niedrāju, grīslāju, meldrāju platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Wetlands_ReedSedgeRushBeds_cell.tif"),
  layer_prefixes = c("Wetlands_ReedSedgeRushBeds"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii        = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

# Wetlands_ReedSedgeRushBeds_r1250.tif egv_476
slanis=rast("./RasterGrids_100m/2024/RAW/Wetlands_ReedSedgeRushBeds_r1250.tif")
names(slanis)="egv_476"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Wetlands_ReedSedgeRushBeds_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Wetlands_ReedSedgeRushBeds_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.477 Wetlands\_ReedSedgeRushBeds\_r3000

**filename:** Wetlands\_ReedSedgeRushBeds\_r3000.tif

**layername:** evg\_477

**English name:** Fractional cover of Reed-, Sedge-, Rush-, Beds within the 3 km landscape

**Latvian name:** Niedrāju, grīslāju, meldrāju platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Wetlands_ReedSedgeRushBeds_cell.tif"),
  layer_prefixes = c("Wetlands_ReedSedgeRushBeds"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii        = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 20 * 1024^3)

# Wetlands_ReedSedgeRushBeds_r3000.tif  evg_477
slanis=rast("./RasterGrids_100m/2024/RAW/Wetlands_ReedSedgeRushBeds_r3000.tif")
names(slanis)="evg_477"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Wetlands_ReedSedgeRushBeds_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Wetlands_ReedSedgeRushBeds_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.478 Wetlands\_ReedSedgeRushBeds\_r10000

**filename:** Wetlands\_ReedSedgeRushBeds\_r10000.tif

**layername:** egv\_478

**English name:** Fractional cover of Reed-, Sedge-, Rush-, Beds within the 10 km landscape

**Latvian name:** Niedrāju, grīslāju, meldrāju platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# Templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii ----
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/Wetlands_ReedSedgeRushBeds_cell.tif"),
  layer_prefixes = c("Wetlands_ReedSedgeRushBeds"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 12,
  radii        = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 20 * 1024^3)

# Wetlands_ReedSedgeRushBeds_r10000.tif egv_478
slanis=rast("./RasterGrids_100m/2024/RAW/Wetlands_ReedSedgeRushBeds_r10000.tif")
names(slanis)="egv_478"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/Wetlands_ReedSedgeRushBeds_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Wetlands_ReedSedgeRushBeds_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.479 EO\_NDMI-LYmed-average\_cell

**filename:** EO\_NDMI-LYmed-average\_cell.tif

**layername:** egv\_479

**English name:** Median vegetation water content index (NDMI) for the last year within the analysis cell (1 ha)

**Latvian name:** Mediānā pēdējā gada ūdens satura veģetācijā indeksa (NDMI) vērtība analīzes šūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The arithmetic mean value at the analysis cell is calculated using the workflow `egvtools::input2egv()`. To protect against potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “last year” is 2024.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EO_NDMI-LYmed-average_cell.tif ----
egvrez=input2egv(input="./Geodata/2024/S2indices/Mosaics/EO_NDMI-LYmedian.tif",
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/RAW/",
  outfilename = "EO_NDMI-LYmed-average_cell.tif",
  layername = "egv_479",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)
egvrez

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="EO_NDMI-LYmed-average_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.480 EO\_NDMI-LYmedian-iqr\_cell

**filename:** EO\_NDMI-LYmedian-iqr\_cell.tif

**layername:** egv\_480

**English name:** Spatial variability of last year’s median vegetation water content index (NDMI) within the analysis cell (1 ha)

**Latvian name:** Telpiskā variabilitāte pēdējā gada mediānajai ūdens satura veģetācijā indeksa (NDMI) vērtībai analīzes šūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The workflow `egvtools::input2egv()` is used to calculate Q1 and Q3 for every cell. To protect against potential data loss at the edges, inverse distance weighted (power = 2) gap filling is implemented. Next, Q1 is subtracted from Q3. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “last year” is 2024.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EO_NDMI-LYmedian-iqr_cell.tif ----

p25rez=input2egv(input="./Geodata/2024/S2indices/Mosaics/EO_NDMI-LYmedian.tif",
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "q1",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/",
  outfilename = "draza_p25.tif",
  layername = "egv_480",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)
p25rez_r=rast("./RasterGrids_100m/2024/draza_p25.tif")

p75rez=input2egv(input="./Geodata/2024/S2indices/Mosaics/EO_NDMI-LYmedian.tif",
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "q3",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/",
  outfilename = "draza_p75.tif",
  layername = "egv_480",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)
p75rez_r=rast("./RasterGrids_100m/2024/draza_p75.tif")

iqr_rez=p75rez_r-p25rez_r
iqr_rez
plot(iqr_rez)

writeRaster(iqr_rez,
  "./RasterGrids_100m/2024/Raw/EO_NDMI-LYmedian-iqr_cell.tif",
  overwrite=TRUE)

unlink("./RasterGrids_100m/2024/draza_p75.tif")
unlink("./RasterGrids_100m/2024/draza_p25.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="EO_NDMI-LYmedian-iqr_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.481 EO\_NDMI-STiqr-median\_cell

**filename:** EO\_NDMI-STiqr-median\_cell.tif

**layername:** egv\_481

**English name:** Average short-term seasonality of vegetation water content index (NDMI) within the analysis cell (1 ha)

**Latvian name:** Sezonālītē pēdējo gadu vidējai ūdens saturā veģetācijā indeksa (NDMI) vērtībai analīzē ūdens saturā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The arithmetic mean value at the analysis cell is calculated using the workflow `egvtools::input2egv()`. To protect against potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “short-term” refers to the last five years (2020-2024).

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EO_NDMI-STiqr-median_cell.tif ----

egvrez=input2egv(input="./Geodata/2024/S2indices/Mosaics/E0_NDMI-STiqr.tif",
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "average",
                  missing_job = "FillOutput",
                  outlocation = "./RasterGrids_100m/2024/Raw/",
                  outfilename = "E0_NDMI-STiqr-median_cell.tif",
                  layername = "egv_481",
                  idw_weight = 2,
                  plot_gaps = FALSE,
                  plot_final = FALSE)
egvrez

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="E0_NDMI-STiqr-median_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)
```

## 6.482 EO\_NDMI-STmedian-average\_cell

**filename:** E0\_NDMI-STmedian-average\_cell.tif

**layername:** egv\_482

**English name:** Median short-term vegetation water content index (NDMI) within the analysis cell (1 ha)

**Latvian name:** Mediānā pēdējo gadu ūdens saturā veģetācijā indeksa (NDMI) vērtība analīzē ūdens saturā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The arithmetic mean value at the analysis cell is calculated using the workflow `egvtools::input2egv()`. To protect against potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “short-term” refers to the last five years (2020-2024).

```
# libs ----
```

```

if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EO_NDMI-STmedian-average_cell.tif ----

egvrez=input2egv(input=".~/Geodata/2024/S2indices/Mosaics/EO_NDMI-STmedian.tif",
  egv_template= ".~/Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/RAW/",
  outfilename = "EO_NDMI-STmedian-average_cell.tif",
  layername = "egv_482",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)
egvrez

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="EO_NDMI-STmedian-average_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.483 EO\_NDMI-STmedian-iqr\_cell

**filename:** EO\_NDMI-STmedian-iqr\_cell.tif

**layername:** egv\_483

**English name:** Spatial variability of short-term median vegetation water content index (NDMI) within the analysis cell (1 ha)

**Latvian name:** Telpiskā variabilitāte pēdējo gadu mediānajai ūdens saturā veģetācijā indeksa (NDMI) vērtībai analīzēs šūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The workflow `egvtools::input2egv()` is used to calculate Q1 and Q3 for every cell. To protect against potential data loss at the edges, inverse distance weighted (power = 2) gap filling is implemented. Next, Q1 is subtracted from Q3. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “short-term” refers to the last five years (2020-2024).

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EO_NDMI-STmedian-iqr_cell.tif ----

p25rez=input2egv(input=".~/Geodata/2024/S2indices/Mosaics/EO_NDMI-STmedian.tif",
  egv_template= ".~/Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "q1",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/",
  outfilename = "draza_p25.tif",

```

```

    layername = "egv_483",
    idw_weight = 2,
    plot_gaps = FALSE,
    plot_final = FALSE)
p25rez_r=rast("./RasterGrids_100m/2024/draza_p25.tif")

p75rez=input2egv(input="./Geodata/2024/S2indices/Mosaics/E0_NDMI-STmedian.tif",
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "q3",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/",
  outfilename = "draza_p75.tif",
  layername = "egv_483",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)
p75rez_r=rast("./RasterGrids_100m/2024/draza_p75.tif")

iqr_rez=p75rez_r-p25rez_r
iqr_rez
plot(iqr_rez)

writeRaster(iqr_rez,
  "./RasterGrids_100m/2024/Raw/E0_NDMI-STmedian-iqr_cell.tif",
  overwrite=TRUE)

unlink("./RasterGrids_100m/2024/draza_p75.tif")
unlink("./RasterGrids_100m/2024/draza_p25.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="E0_NDMI-STmedian-iqr_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.484 EO\_NDMI-STp25-min\_cell

**filename:** EO\_NDMI-STp25-min\_cell.tif

**layername:** egv\_484

**English name:** Minimum short-term 25th percentile of vegetation water content index (NDMI) within the analysis cell (1 ha)

**Latvian name:** Minimālā 25. procentiles pēdējo gadu ūdens satura veģetācijā indeksa (NDMI) vērtība analīzēs šūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The minimum value at the analysis cell is calculated using the workflow `egvtools::input2egv()`. To protect against potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “short-term” refers to the last five years (2020-2024).

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# EO_NDMI-STp25-min_cell.tif ----

egvrez=input2egv(input=".~/Geodata/2024/S2indices/Mosaics/EO_NDMI-STp25.tif",
  evg_template= ".~/Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "min",
  missing_job = "FillOutput",
  outlocation = ".~/RasterGrids_100m/2024/Raw/",
  outfilename = "EO_NDMI-STp25-min_cell.tif",
  layername = "egv_484",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)
egvrez

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="EO_NDMI-STp25-min_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.485 EO\_NDMI-STp75-max\_cell

**filename:** EO\_NDMI-STp75-max\_cell.tif

**layername:** egv\_485

**English name:** Maximum short-term 75th percentile of vegetation water content index (NDMI) within the analysis cell (1 ha)

**Latvian name:** Maksimālā 75. procentīles pēdējo gadu ūdens saturā veģetācijā indeksa (NDMI) vērtība analīzēs šūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The maximum value at the analysis cell is calculated using the workflow `egvtools::input2egv()`. To protect against potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “short-term” refers to the last five years (2020-2024).

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# EO_NDMI-STp75-max_cell.tif ----

egvrez=input2egv(input=".~/Geodata/2024/S2indices/Mosaics/EO_NDMI-STp75.tif",
  evg_template= ".~/Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "min",
  missing_job = "FillOutput",

```

```

        outlocation = "./RasterGrids_100m/2024/RAW/",
        outfilename = "E0_NDMI-STp75-max_cell.tif",
        layername = "egv_485",
        idw_weight = 2,
        plot_gaps = FALSE,
        plot_final = FALSE)
egvrez

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="E0_NDMI-STp75-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.486 EO\_NDVI-LYmedian-average\_cell

**filename:** EO\_NDVI-LYmedian-average\_cell.tif

**layername:** egv\_486

**English name:** Median vegetation index (NDVI) for the last year within the analysis cell (1 ha)

**Latvian name:** Mediānā pēdējā gada veģetācijas indeksa (NDVI) vērtība analīzē ūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The arithmetic mean value at the analysis cell is calculated using the workflow `egvtools::input2egv()`. To protect against potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “last year” is 2024.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EO_NDVI-LYmedian-average_cell.tif ----

egvrez=input2egv(input="./Geodata/2024/S2indices/Mosaics/E0_NDVI-LYmedian.tif",
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "average",
                  missing_job = "FillOutput",
                  outlocation = "./RasterGrids_100m/2024/RAW/",
                  outfilename = "EO_NDVI-LYmedian-average_cell.tif",
                  layername = "egv_486",
                  idw_weight = 2,
                  plot_gaps = FALSE,
                  plot_final = FALSE)
egvrez

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="EO_NDVI-LYmedian-average_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)

```

```

videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.487 EO\_NDVI-LYmedian-iqr\_cell

**filename:** EO\_NDVI-LYmedian-iqr\_cell.tif

**layername:** egv\_487

**English name:** Spatial variability of last year's median vegetation index (NDVI) within the analysis cell (1 ha)

**Latvian name:** Telpiskā variabilitāte pēdējā gada mediānajai veģetācijas indeksa (NDVI) vērtībai analīzes šūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The workflow `egvtools::input2egv()` is used to calculate Q1 and Q3 for every cell. To protect against potential data loss at the edges, inverse distance weighted (power = 2) gap filling is implemented. Next, Q1 is subtracted from Q3. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “last year” is 2024.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EO_NDVI-LYmedian-iqr_cell.tif ----

p25rez=input2egv(input=".~/Geodata/2024/S2indices/Mosaics/E0_NDVI-LYmedian.tif",
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "q1",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/",
  outfilename = "draza_p25.tif",
  layername = "egv_487",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)
p25rez_r=rast("./RasterGrids_100m/2024/draza_p25.tif")

p75rez=input2egv(input=".~/Geodata/2024/S2indices/Mosaics/E0_NDVI-LYmedian.tif",
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "q3",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/",
  outfilename = "draza_p75.tif",
  layername = "egv_487",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)
p75rez_r=rast("./RasterGrids_100m/2024/draza_p75.tif")

iqr_rez=p75rez_r-p25rez_r
iqr_rez
plot(iqr_rez)

writeRaster(iqr_rez,
  "./RasterGrids_100m/2024/Raw/E0_NDVI-LYmedian-iqr_cell.tif",
  overwrite=TRUE)

```

```

overwrite=TRUE)

unlink("./RasterGrids_100m/2024/draza_p75.tif")
unlink("./RasterGrids_100m/2024/draza_p25.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="EO_NDVI-LYmedian-iqr_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.488 EO\_NDVI-STiqr-median\_cell

**filename:** EO\_NDVI-STiqr-median\_cell.tif

**layername:** egv\_488

**English name:** Average short-term seasonality of vegetation index (NDVI) within the analysis cell (1 ha)

**Latvian name:** Sezonālītē pēdējo gadu vidējai veģetācijas indeksa (NDVI) vērtībai analīzes šūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The arithmetic mean value at the analysis cell is calculated using the workflow `egvtools::input2egv()`. To protect against potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “short-term” refers to the last five years (2020-2024).

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EO_NDVI-STiqr-median_cell.tif ----

egvrez=input2egv(input="./Geodata/2024/S2indices/Mosaics/EO_NDVI-STiqr.tif",
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "EO_NDVI-STiqr-median_cell.tif",
  layername = "egv_488",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)

egvrez

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="EO_NDVI-STiqr-median_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)

```

```

centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.489 EO\_NDVI-STmedian-average\_cell

**filename:** EO\_NDVI-STmedian-average\_cell.tif

**layername:** egv\_489

**English name:** Median short-term vegetation index (NDVI) within the analysis cell (1 ha)

**Latvian name:** Mediānā pēdējo gadu veģetācijas indeksa (NDVI) vērtība analīzes šūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The arithmetic mean value at the analysis cell is calculated using the workflow `egvtools::input2egv()`. To protect against potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “short-term” refers to the last five years (2020-2024).

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EO_NDVI-STmedian-average_cell.tif ----

egvrez=input2egv(input=".~/Geodata/2024/S2indices/Mosaics/EO_NDVI-STmedian.tif",
                  egv_template= ".~/Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "average",
                  missing_job = "FillOutput",
                  outlocation = ".~/RasterGrids_100m/2024/RAW/",
                  outfilename = "EO_NDVI-STmedian-average_cell.tif",
                  layername = "egv_489",
                  idw_weight = 2,
                  plot_gaps = FALSE,
                  plot_final = FALSE)
egvrez

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="EO_NDVI-STmedian-average_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.490 EO\_NDVI-STmedian-iqr\_cell

**filename:** EO\_NDVI-STmedian-iqr\_cell.tif

**layername:** egv\_490

**English name:** Spatial variability of short-term median vegetation index (NDVI) within the analysis cell (1 ha)

**Latvian name:** Telpiskā variabilitāte pēdējo gadu mediānajai veģetācijas indeksa (NDVI) vērtībai analīzē ūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The workflow `egvtools::input2egv()` is used to calculate Q1 and Q3 for every cell. To protect against potential data loss at the edges, inverse distance weighted (power = 2) gap filling is implemented. Next, Q1 is subtracted from Q3. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “short-term” refers to the last five years (2020-2024).

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EO_NDVI-STmedian-iqr_cell.tif ----

p25rez=input2egv(input=".~/Geodata/2024/S2indices/Mosaics/EO_NDVI-STmedian.tif",
  egv_template= ".~/Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "q1",
  missing_job = "FillOutput",
  outlocation = ".~/RasterGrids_100m/2024/",
  outfilename = "draza_p25.tif",
  layername = "egv_490",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)
p25rez_r=rast("./RasterGrids_100m/2024/draza_p25.tif")

p75rez=input2egv(input=".~/Geodata/2024/S2indices/Mosaics/EO_NDVI-STmedian.tif",
  egv_template= ".~/Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "q3",
  missing_job = "FillOutput",
  outlocation = ".~/RasterGrids_100m/2024/",
  outfilename = "draza_p75.tif",
  layername = "egv_490",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)
p75rez_r=rast("./RasterGrids_100m/2024/draza_p75.tif")

iqr_rez=p75rez_r-p25rez_r
iqr_rez
plot(iqr_rez)

writeRaster(iqr_rez,
  "./RasterGrids_100m/2024/Raw/EO_NDVI-STmedian-iqr_cell.tif",
  overwrite=TRUE)

unlink("./RasterGrids_100m/2024/draza_p75.tif")
unlink("./RasterGrids_100m/2024/draza_p25.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="EO_NDVI-STmedian-iqr_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
```

```

merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.491 EO\_NDVI-STp25-min\_cell

**filename:** EO\_NDVI-STp25-min\_cell.tif

**layername:** egv\_491

**English name:** Minimum short-term 25th percentile of vegetation index (NDVI) within the analysis cell (1 ha)

**Latvian name:** Minimālā 25. procentiles pēdējo gadu veģetācijas indeksa (NDVI) vērtība analīzes šūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The minimum value at the analysis cell is calculated using the workflow `egvtools::input2egv()`. To protect against potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “short-term” refers to the last five years (2020-2024).

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EO_NDVI-STp25-min_cell.tif ----

egvrez=input2egv(input=".~/Geodata/2024/S2indices/Mosaics/EO_NDVI-STp25.tif",
                  egv_template= ".~/Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "min",
                  missing_job = "FillOutput",
                  outlocation = ".~/RasterGrids_100m/2024/RAW/",
                  outfilename = "EO_NDVI-STp25-min_cell.tif",
                  layername = "egv_491",
                  idw_weight = 2,
                  plot_gaps = FALSE,
                  plot_final = FALSE)
egvrez

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="EO_NDVI-STp25-min_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.492 EO\_NDVI-STp75-max\_cell

**filename:** EO\_NDVI-STp75-max\_cell.tif

**layername:** egv\_492

**English name:** Maximum short-term 75th percentile of vegetation index (NDVI) within the analysis cell (1 ha)

**Latvian name:** Maksimālā 75. procentiles pēdējo gadu veģetācijas indeksa (NDVI) vērtība analīzes šūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The maximum value at the analysis cell is calculated using the workflow `egvtools::input2egv()`. To protect against potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “short-term” refers to the last five years (2020-2024).

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EO_NDVI-STp75-max_cell.tif ----

egvrez=input2egv(input="./Geodata/2024/S2indices/Mosaics/E0_NDVI-STp75.tif",
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "min",
                  missing_job = "FillOutput",
                  outlocation = "./RasterGrids_100m/2024/RAW/",
                  outfilename = "E0_NDVI-STp75-max_cell.tif",
                  layername = "egv_492",
                  idw_weight = 2,
                  plot_gaps = FALSE,
                  plot_final = FALSE)
egvrez

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="E0_NDVI-STp75-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)
```

## 6.493 EO\_NDWI-LYmedian-average\_cell

**filename:** EO\_NDWI-LYmedian-average\_cell.tif

**layername:** egv\_493

**English name:** Median water index (NDWI) for the last year within the analysis cell (1 ha)

**Latvian name:** Mediānā pēdējā gada ūdens indeksa (NDWI) vērtība analīzes šūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The arithmetic mean value at the analysis cell is calculated using the workflow `egvtools::input2egv()`. To protect against potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “last year” is 2024.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}
```

```

egvrez=input2egv(input="./Geodata/2024/S2indices/Mosaics/E0_NDWI-LYmedian.tif",
  evg_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/RAW/",
  outfilename = "E0_NDWI-LYmedian-average_cell.tif",
  layername = "egv_493",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)
egvrez

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="E0_NDWI-LYmedian-average_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.494 EO\_NDWI-LYmedian-iqr\_cell

**filename:** EO\_NDWI-LYmedian-iqr\_cell.tif

**layername:** egv\_494

**English name:** Spatial variability of last year's median water index (NDWI) within the analysis cell (1 ha)

**Latvian name:** Telpiskā variabilitāte pēdējā gada mediānajai ūdens indeksa (NDWI) vērtībai analīzes šūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The workflow `egvtools::input2egv()` is used to calculate Q1 and Q3 for every cell. To protect against potential data loss at the edges, inverse distance weighted (power = 2) gap filling is implemented. Next, Q1 is subtracted from Q3. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “last year” is 2024.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EO_NDWI-LYmedian-iqr_cell.tif ----

p25rez=input2egv(input="./Geodata/2024/S2indices/Mosaics/E0_NDWI-LYmedian.tif",
  evg_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "q1",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/",
  outfilename = "draza_p25.tif",
  layername = "egv_494",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)
p25rez_r=rast("./RasterGrids_100m/2024/draza_p25.tif")

```

```

p75rez=input2egv(input="./Geodata/2024/S2indices/Mosaics/E0_NDWI-LYmedian.tif",
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "q3",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/",
  outfilename = "draza_p75.tif",
  layername = "egv_494",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)
p75rez_r=rast("./RasterGrids_100m/2024/draza_p75.tif")

iqr_rez=p75rez_r-p25rez_r
iqr_rez
plot(iqr_rez)

writeRaster(iqr_rez,
  "./RasterGrids_100m/2024/Raw/E0_NDWI-LYmedian-iqr_cell.tif",
  overwrite=TRUE)

unlink("./RasterGrids_100m/2024/draza_p75.tif")
unlink("./RasterGrids_100m/2024/draza_p25.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="E0_NDWI-LYmedian-iqr_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.495 EO\_NDWI-STiqr-median\_cell

**filename:** EO\_NDWI-STiqr-median\_cell.tif

**layername:** egv\_495

**English name:** Average short-term seasonality of water index (NDWI) within the analysis cell (1 ha)

**Latvian name:** Sezonālītātē pēdējo gadu vidējai ūdens indeksa (NDWI) vērtībai analīzes šūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The arithmetic mean value at the analysis cell is calculated using the workflow `egvtools::input2egv()`. To protect against potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “short-term” refers to the last five years (2020-2024).

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EO_NDWI-STiqr-median_cell.tif ----

egvrez=input2egv(input="./Geodata/2024/S2indices/Mosaics/E0_NDWI-STiqr.tif",
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",

```

```

    missing_job = "FillOutput",
    outlocation = "./RasterGrids_100m/2024/RAW/",
    outfilename = "E0_NDWI-STiqr-median_cell.tif",
    layername = "egv_495",
    idw_weight = 2,
    plot_gaps = FALSE,
    plot_final = FALSE)
egvrez

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="E0_NDWI-STiqr-median_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.496 EO\_NDWI-STmedian-average\_cell

**filename:** E0\_NDWI-STmedian-average\_cell.tif

**layername:** egv\_496

**English name:** Median short-term water index (NDWI) within the analysis cell (1 ha)

**Latvian name:** Mediānā pēdējo gadu ūdens indeksa (NDWI) vērtība analīzes šūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The arithmetic mean value at the analysis cell is calculated using the workflow `egvtools::input2egv()`. To protect against potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “short-term” refers to the last five years (2020-2024).

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# E0_NDWI-STmedian-average_cell.tif ----

egvrez=input2egv(input="./Geodata/2024/S2indices/Mosaics/E0_NDWI-STmedian.tif",
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "average",
                  missing_job = "FillOutput",
                  outlocation = "./RasterGrids_100m/2024/RAW/",
                  outfilename = "E0_NDWI-STmedian-average_cell.tif",
                  layername = "egv_496",
                  idw_weight = 2,
                  plot_gaps = FALSE,
                  plot_final = FALSE)
egvrez

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="E0_NDWI-STmedian-average_cell.tif"

```

```

ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.497 EO\_NDWI-STmedian-iqr\_cell

**filename:** EO\_NDWI-STmedian-iqr\_cell.tif

**layername:** egv\_497

**English name:** Spatial variability of short-term median water index (NDWI) within the analysis cell (1 ha)

**Latvian name:** Telpiskā variabilitāte pēdējo gadu mediānajai ūdens indeksa (NDWI) vērtībai analīzes šūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The workflow `egvtools::input2egv()` is used to calculate Q1 and Q3 for every cell. To protect against potential data loss at the edges, inverse distance weighted (power = 2) gap filling is implemented. Next, Q1 is subtracted from Q3. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “short-term” refers to the last five years (2020-2024).

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EO_NDWI-STmedian-iqr_cell.tif ----

p25rez=input2egv(input="./Geodata/2024/S2indices/Mosaics/EO_NDWI-STmedian.tif",
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "q1",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/",
  outfilename = "draza_p25.tif",
  layername = "egv_497",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)
p25rez_r=rast("./RasterGrids_100m/2024/draza_p25.tif")

p75rez=input2egv(input="./Geodata/2024/S2indices/Mosaics/EO_NDWI-STmedian.tif",
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "q3",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/",
  outfilename = "draza_p75.tif",
  layername = "egv_497",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)
p75rez_r=rast("./RasterGrids_100m/2024/draza_p75.tif")

iqr_rez=p75rez_r-p25rez_r
iqr_rez
plot(iqr_rez)

```

```

writeRaster(iqr_rez,
  "./RasterGrids_100m/2024/RAW/E0_NDWI-STmedian-iqr_cell.tif",
  overwrite=TRUE)

unlink("./RasterGrids_100m/2024/draza_p75.tif")
unlink("./RasterGrids_100m/2024/draza_p25.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="E0_NDWI-STmedian-iqr_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.498 EO\_NDWI-STp25-min\_cell

**filename:** EO\_NDWI-STp25-min\_cell.tif

**layername:** egv\_498

**English name:** Minimum short-term 25th percentile of water index (NDWI) within the analysis cell (1 ha)

**Latvian name:** Minimālā 25. procentiles pēdējo gadu ūdens indeksa (NDWI) vērtība analīzes šūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The minimum value at the analysis cell is calculated using the workflow `egvtools::input2egv()`. To protect against potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “short-term” refers to the last five years (2020-2024).

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EO_NDWI-STp25-min_cell.tif ----

egvrez=input2egv(input="./Geodata/2024/S2indices/Mosaics/E0_NDWI-STp25.tif",
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "min",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/RAW/",
  outfilename = "E0_NDWI-STp25-min_cell.tif",
  layername = "egv_498",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)
egvrez

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="E0_NDWI-STp25-min_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)

```

```

slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.499 EO\_NDWI-STp75-max\_cell

**filename:** EO\_NDWI-STp75-max\_cell.tif

**layername:** egv\_499

**English name:** Maximum short-term 75th percentile of water index (NDWI) within the analysis cell (1 ha)

**Latvian name:** Maksimālā 75. procentiles pēdējo gadu ūdens indeksa (NDWI) vērtība analīzes šūnā (1 ha)

**Procedure:** Directly follows [preprocessing](#). The maximum value at the analysis cell is calculated using the workflow `egvtools::input2egv()`. To protect against potential data loss at edge cells, inverse distance weighted (power = 2) gap filling is implemented. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error. The “short-term” refers to the last five years (2020-2024).

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EO_NDWI-STp75-max_cell.tif ----

egvrez=input2egv(input="./Geodata/2024/S2indices/Mosaics/EO_NDWI-STp75.tif",
  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "min",
  missing_job = "FillOutput",
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "EO_NDWI-STp75-max_cell.tif",
  layername = "egv_499",
  idw_weight = 2,
  plot_gaps = FALSE,
  plot_final = FALSE)
egvrez

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="EO_NDWI-STp75-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.500 SoilChemistry\_ESDAC-CN\_cell

**filename:** SoilChemistry\_ESDAC-CN\_cell.tif

**layername:** egv\_500

**English name:** Average value of Topsoil Carbon-Nitrogen ratio (ESDAC v2.0) within the analysis cell (1 ha)

**Latvian name:** Augsnes virskārtas oglekļa-slāpekļa attiecība (ESDAC v2.0) analīzes šūnā (1 ha)

**Procedure:** Directly derived from the [Soil chemistry](#). Processed using the workflow `egvtools::downscale2egv()` with `fill_gaps = TRUE`, performing inverse distance weighted (`power = 2`) filling of gaps at the border and `smooth = FALSE`. This is done to preserve the original values as much as possible (bilinear interpolation is involved when projecting from a 500 m resolution to a 100 m resolution in a different CRS). Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# CN ----

egv=downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = "./Geodata/2024/Soils/ESDAC/chemistry/chemistry/CN/CN.tif",
  out_path     = "./RasterGrids_100m/2024/Raw/",
  file_name    = "SoilChemistry_ESDAC-CN_cell.tif",
  layer_name   = "egv_500",
  fill_gaps    = TRUE,
  smooth       = FALSE,
  plot_result  = TRUE)
egv

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilChemistry_ESDAC-CN_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.501 SoilChemistry\_ESDAC-CaCo3\_cell

**filename:** SoilChemistry\_ESDAC-CaCo3\_cell.tif

**layername:** egv\_501

**English name:** Average value of Topsoil Calcium Carbonate Content (ESDAC v2.0) within the analysis cell (1 ha)

**Latvian name:** Augsnes virskārtas kalcija karbonātu saturs (ESDAC v2.0) analīzes šūnā (1 ha)

**Procedure:** Directly derived from the [Soil chemistry](#). Processed using the workflow `egvtools::downscale2egv()` with `fill_gaps = TRUE`, performing inverse distance weighted (`power = 2`) filling of gaps at the border and `smooth = FALSE`. This is done to preserve the original values as much as possible (bilinear interpolation is involved when projecting from a 500 m resolution to a 100 m resolution in a different CRS). Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# CaCO3 ----

egv=downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = "./Geodata/2024/Soils/ESDAC/chemistry/chemistry/Caco3/CaCO3.tif",
  out_path     = "./RasterGrids_100m/2024/Raw/",
  file_name    = "SoilChemistry_ESDAC-CaCO3_cell.tif",
  layer_name   = "egv_501",
  fill_gaps    = TRUE,
  smooth       = FALSE,
  plot_result  = TRUE)
egv

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilChemistry_ESDAC-CaCO3_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnvorze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnvorze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.502 SoilChemistry\_ESDAC-K\_cell

**filename:** SoilChemistry\_ESDAC-K\_cell.tif

**layername:** egv\_502

**English name:** Average value of Topsoil Potassium Content (ESDAC v2.0) within the analysis cell (1 ha)

**Latvian name:** Augsnes virskārtas kālīja saturs (ESDAC v2.0) analīzes šūnā (1 ha)

**Procedure:** Directly derived from the [Soil chemistry](#). Processed using the workflow `egvtools::downscale2egv()` with `fill_gaps = TRUE`, performing inverse distance weighted (power = 2) filling of gaps at the border and `smooth = FALSE`. This is done to preserve the original values as much as possible (bilinear interpolation is involved when projecting from a 500 m resolution to a 100 m resolution in a different CRS). Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}


# K ----

egv=downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = "./Geodata/2024/Soils/ESDAC/chemistry/chemistry/K/K.tif",

```

```

out_path = "./RasterGrids_100m/2024/RAW/",
file_name = "SoilChemistry_ESDAC-K_cell.tif",
layer_name = "egv_502",
fill_gaps = TRUE,
smooth = FALSE,
plot_result = TRUE)
egv

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilChemistry_ESDAC-K_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.503 SoilChemistry\_ESDAC-N\_cell

**filename:** SoilChemistry\_ESDAC-N\_cell.tif

**layername:** egv\_503

**English name:** Average value of Topsoil Nitrogen Content (ESDAC v2.0) within the analysis cell (1 ha)

**Latvian name:** Augsnes virskārtas slāpekļa saturs (ESDAC v2.0) analīzes šūnā (1 ha)

**Procedure:** Directly derived from the [Soil chemistry](#). Processed using the workflow `egv-tools::downscale2egv()` with `fill_gaps = TRUE`, performing inverse distance weighted (power = 2) filling of gaps at the border and `smooth = FALSE`. This is done to preserve the original values as much as possible (bilinear interpolation is involved when projecting from a 500 m resolution to a 100 m resolution in a different CRS). Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# N ----

egv=downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = "./Geodata/2024/Soils/ESDAC/chemistry/chemistry/N/N.tif",
  out_path = "./RasterGrids_100m/2024/RAW/",
  file_name = "SoilChemistry_ESDAC-N_cell.tif",
  layer_name = "egv_503",
  fill_gaps = TRUE,
  smooth = FALSE,
  plot_result = TRUE)
egv

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

nosaukums="SoilChemistry_ESDAC-N_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.504 SoilChemistry\_ESDAC-P\_cell

**filename:** SoilChemistry\_ESDAC-P\_cell.tif

**layername:** egv\_504

**English name:** Average value of Topsoil Phosphorous Content (ESDAC v2.0) within the analysis cell (1 ha)

**Latvian name:** Augsnes virskārtas fosfora saturs (ESDAC v2.0) analīzes šūnā (1 ha)

**Procedure:** Directly derived from the [Soil chemistry](#). Processed using the workflow `egv-tools::downscale2egv()` with `fill_gaps = TRUE`, performing inverse distance weighted (power = 2) filling of gaps at the border and `smooth = FALSE`. This is done to preserve the original values as much as possible (bilinear interpolation is involved when projecting from a 500 m resolution to a 100 m resolution in a different CRS). Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# P ----

egv=downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = "./Geodata/2024/Soils/ESDAC/chemistry/chemistry/P/P.tif",
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name   = "SoilChemistry_ESDAC-P_cell.tif",
  layer_name  = "egv_504",
  fill_gaps   = TRUE,
  smooth      = FALSE,
  plot_result = TRUE)
egv

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilChemistry_ESDAC-P_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.505 SoilChemistry\_ESDAC-phH2O\_cell

**filename:** SoilChemistry\_ESDAC-phH20\_cell.tif

**layername:** evg\_505

**English name:** Average value of Topsoil pH reaction in water (ESDAC v2.0) within the analysis cell (1 ha)

**Latvian name:** Augsnes virskārtas reakcija (pH) ūdens šķīdumā (ESDAC v2.0) analīzes šūnā (1 ha)

**Procedure:** Directly derived from the [Soil chemistry](#). Processed using the workflow `egvtools::downscale2egv()` with `fill_gaps = TRUE`, performing inverse distance weighted (power = 2) filling of gaps at the border and `smooth = FALSE`. This is done to preserve the original values as much as possible (bilinear interpolation is involved when projecting from a 500 m resolution to a 100 m resolution in a different CRS). Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# pH_H2O ----

egv=downscale2egv(
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  grid_path    = "./Templates/TemplateGrids/tikls1km_sauzeme.parquet",
  rawfile_path = "./Geodata/2024/Soils/ESDAC/chemistry/chemistry/pH_H20/pH_H20.tif",
  out_path     = "./RasterGrids_100m/2024/RAW/",
  file_name   = "SoilChemistry_ESDAC-phH20_cell.tif",
  layer_name   = "evg_505",
  fill_gaps    = TRUE,
  smooth       = FALSE,
  plot_result  = TRUE)
egv

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilChemistry_ESDAC-phH20_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.506 SoilTexture\_Clay\_cell

**filename:** SoilTexture\_Clay\_cell.tif

**layername:** evg\_506

**English name:** Fractional cover of Clay Soils within the analysis cell (1 ha)

**Latvian name:** Augsnes granulometriskās klases “māls” platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** Derived from the [Soil texture product](#). First, the layer is reclassified so that the class of interest is 1 and the other classes are 0. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During

aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# input ----
combttext=rast("./RasterGrids_10m/2024/SoilTXT_combined.tif")

# EGVs cell ----

# SoilTexture_Clay_cell.tif egv_506

clay10=ifel(combttext==3,1,0)

input2egv(input=clay10,
  egv_template="./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  idw_weight = 2,
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "SoilTexture_Clay_cell.tif",
  layername="egv_506",
  return_visible = TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Clay_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.507 SoilTexture\_Clay\_r500

**filename:** SoilTexture\_Clay\_r500.tif

**layername:** egv\_507

**English name:** Fractional cover of Clay Soils within the 0.5 km landscape

**Latvian name:** Augsnes granulometriskās klases “māls” platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EGVs radii ----

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/SoilTexture_Clay_cell.tif"),
  layer_prefixes = c("SoilTexture_Clay"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 5,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 5 * 1024^3)

# SoilTexture_Clay_r500.tif egv_507

slanis=rast("./RasterGrids_100m/2024/RAW/SoilTexture_Clay_r500.tif")
names(slanis)="egv_507"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/SoilTexture_Clay_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Clay_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.508 SoilTexture\_Clay\_r1250

**filename:** SoilTexture\_Clay\_r1250.tif

**layername:** egv\_508

**English name:** Fractional cover of Clay Soils within the 1.25 km landscape

**Latvian name:** Augsnes granulometriskās klases “māls” platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EGVs radii ----

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/SoilTexture_Clay_cell.tif"),
  layer_prefixes = c("SoilTexture_Clay"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 5,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 5 * 1024^3)

# SoilTexture_Clay_r1250.tif    egv_508

slanis=rast("./RasterGrids_100m/2024/RAW/SoilTexture_Clay_r1250.tif")
names(slanis)="egv_508"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/SoilTexture_Clay_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Clay_r1250.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.509 SoilTexture\_Clay\_r3000

**filename:** SoilTexture\_Clay\_r3000.tif

**layername:** egv\_509

**English name:** Fractional cover of Clay Soils within the 3 km landscape

**Latvian name:** Augsnes granulometriskās klases “māls” platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name.

Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EGVs radii ----

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/SoilTexture_Clay_cell.tif"),
  layer_prefixes = c("SoilTexture_Clay"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 5,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 5 * 1024^3)

# SoilTexture_Clay_r3000.tif      egv_509

slanis=rast("./RasterGrids_100m/2024/RAW/SoilTexture_Clay_r3000.tif")
names(slanis)="egv_509"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/SoilTexture_Clay_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Clay_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra:::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.510 SoilTexture\_Clay\_r10000

**filename:** SoilTexture\_Clay\_r10000.tif

**layername:** egv\_510

**English name:** Fractional cover of Clay Soils within the 10 km landscape

**Latvian name:** Augsnes granulometriskās klases “māls” platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name.

Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EGVs radii ----

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path   = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/SoilTexture_Clay_cell.tif"),
  layer_prefixes = c("SoilTexture_Clay"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 5,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 5 * 1024^3)

# SoilTexture_Clay_r10000.tif  egv_510

slanis=rast("./RasterGrids_100m/2024/RAW/SoilTexture_Clay_r10000.tif")
names(slanis)="egv_510"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/SoilTexture_Clay_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Clay_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.511 SoilTexture\_Organic\_cell

**filename:** SoilTexture\_Organic\_cell.tif

**layername:** egv\_511

**English name:** Fractional cover of Organic Soils within the analysis cell (1 ha)

**Latvian name:** Augsnes granulometriskās klases “organiskās augsnes” platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** Derived from the [Soil texture product](#). First, the layer is reclassified so that the class of interest is 1 and the other classes are 0. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During

aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# input ----
combttext=rast("./RasterGrids_10m/2024/SoilTXT_combined.tif")

# EGVs cell ----

# SoilTexture_Organic_cell.tif  egv_511

org10=ifel(combttext==4,1,0)

input2egv(input=org10,
  egv_template="./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  idw_weight = 2,
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "SoilTexture_Organic_cell.tif",
  layername="egv_511",
  return_visible = TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Organic_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.512 SoilTexture\_Organic\_r500

**filename:** SoilTexture\_Organic\_r500.tif

**layername:** egv\_512

**English name:** Fractional cover of Organic Soils within the 0.5 km landscape

**Latvian name:** Augsnes granulometriskās klases “organiskās augsnes” platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EGVs radii ----

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/SoilTexture_Organic_cell.tif"),
  layer_prefixes = c("SoilTexture_Organic"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 5,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 5 * 1024^3)

# SoilTexture_Organic_r500.tif egv_512

slanis=rast("./RasterGrids_100m/2024/RAW/SoilTexture_Organic_r500.tif")
names(slanis)="egv_512"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/SoilTexture_Organic_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Organic_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.513 SoilTexture\_Organic\_r1250

**filename:** SoilTexture\_Organic\_r1250.tif

**layername:** egv\_513

**English name:** Fractional cover of Organic Soils within the 1.25 km landscape

**Latvian name:** Augsnes granulometriskās klases “organiskās augsnes” platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EGVs radii ----

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/SoilTexture_Organic_cell.tif"),
  layer_prefixes = c("SoilTexture_Organic"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 5,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 5 * 1024^3)

# SoilTexture_Organic_r1250.tif egv_513

slanis=rast("./RasterGrids_100m/2024/RAW/SoilTexture_Organic_r1250.tif")
names(slanis)="egv_513"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/SoilTexture_Organic_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Organic_r1250.tif"
ielasianas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasianas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.514 SoilTexture\_Organic\_r3000

**filename:** SoilTexture\_Organic\_r3000.tif

**layername:** egv\_514

**English name:** Fractional cover of Organic Soils within the 3 km landscape

**Latvian name:** Augsnes granulometriskās klases “organiskās augsnes” platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name.

Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EGVs radii ----

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/SoilTexture_Organic_cell.tif"),
  layer_prefixes = c("SoilTexture_Organic"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 5,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 5 * 1024^3)

# SoilTexture_Organic_r3000.tif egv_514

slanis=rast("./RasterGrids_100m/2024/RAW/SoilTexture_Organic_r3000.tif")
names(slanis)="egv_514"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/SoilTexture_Organic_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Organic_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra:::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.515 SoilTexture\_Organic\_r10000

**filename:** SoilTexture\_Organic\_r10000.tif

**layername:** egv\_515

**English name:** Fractional cover of Organic Soils within the 10 km landscape

**Latvian name:** Augsnes granulometriskās klases “organiskās augsnes” platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name.

Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EGVs radii ----

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/SoilTexture_Organic_cell.tif"),
  layer_prefixes = c("SoilTexture_Organic"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 5,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 5 * 1024^3)

# SoilTexture_Organic_r10000.tif      egv_515

slanis=rast("./RasterGrids_100m/2024/RAW/SoilTexture_Organic_r10000.tif")
names(slanis)="egv_515"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/SoilTexture_Organic_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Organic_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra:::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.516 SoilTexture\_Sand\_cell

**filename:** SoilTexture\_Sand\_cell.tif

**layername:** egv\_516

**English name:** Fractional cover of Sand Soils within the analysis cell (1 ha)

**Latvian name:** Augsnes granulometriskās klases “smilts” platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** Derived from the [Soil texture product](#). First, the layer is reclassified so that the class of interest is 1 and the other classes are 0. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing

values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# input ----
combttext=rast("./RasterGrids_10m/2024/SoilTXT_combined.tif")

# EGVs cell ----

# SoilTexture_Sand_cell.tif egv_516

sand10=ifel(combttext==1,1,0)
plot(sand10)

input2egv(input=sand10,
  egv_template="./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  idw_weight = 2,
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "SoilTexture_Sand_cell.tif",
  layername="egv_516",
  return_visible = TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Sand_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.517 SoilTexture\_Sand\_r500

**filename:** SoilTexture\_Sand\_r500.tif

**layername:** egv\_517

**English name:** Fractional cover of Sand Soils within the 0.5 km landscape

**Latvian name:** Augsnes granulometriskās klases “smilts” platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EGVs radii ----

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/SoilTexture_Sand_cell.tif"),
  layer_prefixes = c("SoilTexture_Sand"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 5,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 5 * 1024^3)

# SoilTexture_Sand_r500.tif egv_517

slanis=rast("./RasterGrids_100m/2024/RAW/SoilTexture_Sand_r500.tif")
names(slanis)="egv_517"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/SoilTexture_Sand_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Sand_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.518 SoilTexture\_Sand\_r1250

**filename:** SoilTexture\_Sand\_r1250.tif

**layername:** egv\_518

**English name:** Fractional cover of Sand Soils within the 1.25 km landscape

**Latvian name:** Augsnes granulometriskās klases “smilts” platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EGVs radii ----

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/SoilTexture_Sand_cell.tif"),
  layer_prefixes = c("SoilTexture_Sand"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 5,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 5 * 1024^3)

# SoilTexture_Sand_r1250.tif      egv_518

slanis=rast("./RasterGrids_100m/2024/RAW/SoilTexture_Sand_r1250.tif")
names(slanis)="egv_518"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/SoilTexture_Sand_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Sand_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.519 SoilTexture\_Sand\_r3000

**filename:** SoilTexture\_Sand\_r3000.tif

**layername:** egv\_519

**English name:** Fractional cover of Sand Soils within the 3 km landscape

**Latvian name:** Augsnes granulometriskās klases “smilts” platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EGVs radii ----

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/SoilTexture_Sand_cell.tif"),
  layer_prefixes = c("SoilTexture_Sand"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 5,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 5 * 1024^3)

# SoilTexture_Sand_r3000.tif      egv_519

slanis=rast("./RasterGrids_100m/2024/RAW/SoilTexture_Sand_r3000.tif")
names(slanis)="egv_519"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/SoilTexture_Sand_r3000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Sand_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.520 SoilTexture\_Sand\_r10000

**filename:** SoilTexture\_Sand\_r10000.tif

**layername:** egv\_520

**English name:** Fractional cover of Sand Soils within the 10 km landscape

**Latvian name:** Augsnes granulometriskās klases “smilts” platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EGVs radii ----

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/SoilTexture_Sand_cell.tif"),
  layer_prefixes = c("SoilTexture_Sand"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 5,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 5 * 1024^3)

# SoilTexture_Sand_r10000.tif  egv_520

slanis=rast("./RasterGrids_100m/2024/RAW/SoilTexture_Sand_r10000.tif")
names(slanis)="egv_520"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/SoilTexture_Sand_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Sand_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.521 SoilTexture\_Silt\_cell

**filename:** SoilTexture\_Silt\_cell.tif

**layername:** egv\_521

**English name:** Fractional cover of Silt Soils within the analysis cell (1 ha)

**Latvian name:** Augsnes granulometriskās klases “smilšmāls un mālsmilts” platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** Derived from the [Soil texture product](#). First, the layer is reclassified so that the class of interest is 1 and the other classes are 0. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing

values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template10=rast("./Templates/TemplateRasters/LV10m_10km.tif")
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# input ----
combtext=rast("./RasterGrids_10m/2024/SoilTXT_combined.tif")

# EGVs cell ----

# SoilTexture_Silt_cell.tif egv_521

silt10=ifel(combtext==2,1,0)

input2egv(input=silt10,
  egv_template="./Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  idw_weight = 2,
  outlocation = "./RasterGrids_100m/2024/Raw/",
  outfilename = "SoilTexture_Silt_cell.tif",
  layername="egv_521",
  return_visible = TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Silt_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.522 SoilTexture\_Silt\_r500

**filename:** SoilTexture\_Silt\_r500.tif

**layername:** egv\_522

**English name:** Fractional cover of Silt Soils within the 0.5 km landscape

**Latvian name:** Augsnes granulometriskās klases “smilšmāls un mālsmilts” platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EGVs radii ----

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/SoilTexture_Silt_cell.tif"),
  layer_prefixes = c("SoilTexture_Silt"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 5,
  radii         = c("r500"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 5 * 1024^3)

# SoilTexture_Silt_r500.tif egv_522

slanis=rast("./RasterGrids_100m/2024/RAW/SoilTexture_Silt_r500.tif")
names(slanis)="egv_522"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/SoilTexture_Silt_r500.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Silt_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.523 SoilTexture\_Silt\_r1250

**filename:** SoilTexture\_Silt\_r1250.tif

**layername:** egv\_523

**English name:** Fractional cover of Silt Soils within the 1.25 km landscape

**Latvian name:** Augsnes granulometriskās klases “smilšmāls un mālsmilts” platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name.

Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EGVs radii ----

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/SoilTexture_Silt_cell.tif"),
  layer_prefixes = c("SoilTexture_Silt"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 5,
  radii         = c("r1250"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight    = 2,
  future_max_size = 5 * 1024^3)

# SoilTexture_Silt_r1250.tif      egv_523

slanis=rast("./RasterGrids_100m/2024/RAW/SoilTexture_Silt_r1250.tif")
names(slanis)="egv_523"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/SoilTexture_Silt_r1250.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Silt_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra:::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.524 SoilTexture\_Silt\_r3000

**filename:** SoilTexture\_Silt\_r3000.tif

**layername:** egv\_524

**English name:** Fractional cover of Silt Soils within the 3 km landscape

**Latvian name:** Augsnes granulometriskās klases “smilšmāls un mālsmilts” platības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name.

Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EGVs radii ----

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/SoilTexture_Silt_cell.tif"),
  layer_prefixes = c("SoilTexture_Silt"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 5,
  radii         = c("r3000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 5 * 1024^3)

# SoilTexture_Silt_r3000.tif      egv_524

slanis=rast("./RasterGrids_100m/2024/RAW/SoilTexture_Silt_r3000.tif")
names(slanis)="egv_524"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/SoilTexture_Silt_r3000.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Silt_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.525 SoilTexture\_Silt\_r10000

**filename:** SoilTexture\_Silt\_r10000.tif

**layername:** egv\_525

**English name:** Fractional cover of Silt Soils within the 10 km landscape

**Latvian name:** Augsnes granulometriskās klases “smilšmāls un mālsmilts” platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the

output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# EGVs radii ----

radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers  = c("./RasterGrids_100m/2024/RAW/SoilTexture_Silt_cell.tif"),
  layer_prefixes = c("SoilTexture_Silt"),
  output_dir    = "./RasterGrids_100m/2024/RAW/",
  n_workers     = 5,
  radii         = c("r10000"),
  radius_mode   = "sparse",
  extract_fun   = "mean",
  fill_missing   = TRUE,
  IDW_weight   = 2,
  future_max_size = 5 * 1024^3)

# SoilTexture_Silt_r10000.tif  egv_525

slanis=rast("./RasterGrids_100m/2024/RAW/SoilTexture_Silt_r10000.tif")
names(slanis)="egv_525"
slanis2=project(slanis,template100)
writeRaster(slanis2,
  "./RasterGrids_100m/2024/RAW/SoilTexture_Silt_r10000.tif",
  overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="SoilTexture_Silt_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)
```

## 6.526 Terrain\_ASL-average\_cell

**filename:** Terrain\_ASL-average\_cell.tif

**layername:** egv\_526

**English name:** Average value of height Above Sea Level (m) within the analysis cell (1 ha)

**Latvian name:** Augstums virs jūras līmeņa (m) analīzes šūnā (1 ha)

**Procedure:** Derived from the [Digital elevation/terrain models](#). Processed using the workflow `egv-tools::input2egv()`. Inverse distance weighted (power = 2) gap filling is implemented to protect against

potential data loss at edge cells. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# Terrain_ASL-average_cell.tif  egv_526

input2egv(input=".~/Geodata/2024/DEM/mozDEM_10m.tif",
          egv_template=".~/Templates/TemplateRasters/LV100m_10km.tif",
          summary_function = "average",
          missing_job = "FillOutput",
          idw_weight = 2,
          outlocation = ".~/RasterGrids_100m/2024/RAW/",
          outfilename = "Terrain_ASL-average_cell.tif",
          layername="egv_526",
          return_visible = TRUE,
          plot_final = TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Terrain_ASL-average_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)
```

## 6.527 Terrain\_Aspect-average\_cell

**filename:** Terrain\_Aspect-average\_cell.tif

**layername:** egv\_527

**English name:** Average value of Terrain Aspect (degree) within the analysis cell (1 ha)

**Latvian name:** Nogāzes vidējais vērsuma virziens (grādi) analīzes šūnā (1 ha)

**Procedure:** Derived from the [Terrain products](#). Processed using the workflow `egvtools::input2egv()`. Inverse distance weighted (power = 2) gap filling is implemented to protect against potential data loss at edge cells. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# Terrain_Aspect-average_cell.tif  egv_527
```

```



```

## 6.528 Terrain\_Aspect-iqr\_cell

**filename:** Terrain\_Aspect-iqr\_cell.tif

**layername:** egv\_528

**English name:** Variability of Terrain Aspect (degree) within the analysis cell (1 ha)

**Latvian name:** Nogāzes vērsuma (grādi) variabilitāte analīzēs šūnā (1 ha)

**Procedure:** Derived from the [Terrain products](#). The workflow `egvtools::input2egv()` is used to calculate Q1 and Q3 for every cell. To protect against potential data loss at the edges, inverse distance weighted (power = 2) gap filling is implemented. Next, Q1 is subtracted from Q3. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# Terrain_Aspect-iqr_cell.tif  egv_528
p25rez=input2egv(input=".~/RasterGrids_10m/2024/Terrain_Aspect_udeni2_10m.tif",
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "q1",
                  missing_job = "FillOutput",
                  outlocation = ".~/RasterGrids_100m/2024/",
                  outfilename = "draza_p25.tif",
                  layername = "egv_528",
                  idw_weight = 2)
p25rez_r=rast("./RasterGrids_100m/2024/draza_p25.tif")

p75rez=input2egv(input=".~/RasterGrids_10m/2024/Terrain_Aspect_udeni2_10m.tif",
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",

```

```

summary_function = "q3",
missing_job = "FillOutput",
outlocation = "./RasterGrids_100m/2024/",
outfilename = "draza_p75.tif",
layername = "egv_528",
idw_weight = 2)
p75rez_r=rast("./RasterGrids_100m/2024/draza_p75.tif")

iqr_rez=p75rez_r-p25rez_r
iqr_rez
plot(iqr_rez)

writeRaster(iqr_rez,
            "./RasterGrids_100m/2024/Raw/Terrain_Aspect-iqr_cell.tif",
            overwrite=TRUE)

unlink("./RasterGrids_100m/2024/draza_p75.tif")
unlink("./RasterGrids_100m/2024/draza_p25.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Terrain_Aspect-iqr_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.529 Terrain\_Dis-area\_cell

**filename:** Terrain\_Dis-area\_cell.tif

**layername:** egv\_529

**English name:** Fractional cover of Terrain Sinks within the analysis cell (1 ha)

**Latvian name:** Reljefa depresiju bez virszemes notecees platības īpatsvars analīzes šūnā (1 ha)

**Procedure:** Derived from the [Terrain products](#) depth-in-sinks layer, which is reclassified to a value of 1 in every cell with a positive value. The resulting layer is then aggregated to EGV resolution using the workflow `egvtools::input2egv()`, which calculates the arithmetic mean to determine the cover fraction. During aggregation, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# Terrain_Dis-area_cell.tif egv_529
dis=rast("./RasterGrids_10m/2024/Terrain_Dis_udeni2_10m.tif")
dis2=ifel(dis>0,1,dis)

```

```



```

## 6.530 Terrain\_Dis-area\_r500

**filename:** Terrain\_Dis-area\_r500.tif

**layername:** egv\_530

**English name:** Fractional cover of Terrain Sinks within the 0.5 km landscape

**Latvian name:** Reljefa depresiju bez virszemes notecees platības īpatsvars 0,5 km ainavā

**Procedure:** The cover fraction within a radius of 500 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii
radius_function(
    kvadrati_path = "./Templates/TemplateGrids/tiles/",
    radii_path    = "./Templates/TemplateGridPoints/tiles/",
    tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
    template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
    input_layers  = c("./RasterGrids_100m/2024/Raw/Terrain_Dis-area_cell.tif"),
    layer_prefixes = c("Terrain_Dis-area"),
    output_dir    = "./RasterGrids_100m/2024/Raw/",
    n_workers     = 5,
    radii        = c("r500"),

```

```

radius_mode  = "sparse",
extract_fun  = "mean",
fill_missing  = TRUE,
IDW_weight   = 2,
future_max_size = 5 * 1024^3)

# Terrain_Dis-area_r500.tif egv_530
slanis=rast("./RasterGrids_100m/2024/RAW/Terrain_Dis-area_r500.tif")
names(slanis)="egv_530"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Terrain_Dis-area_r500.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Terrain_Dis-area_r500.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.531 Terrain\_Dis-area\_r1250

**filename:** Terrain\_Dis-area\_r1250.tif

**layername:** egv\_531

**English name:** Fractional cover of Terrain Sinks within the 1.25 km landscape

**Latvian name:** Reljefa depresiju bez virszemes notecees platības īpatsvars 1,25 km ainavā

**Procedure:** The cover fraction within a radius of 1250 m around the analysis grid cell is calculated as the area-weighted sum of the [analysis cells](#) inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii
radius_function(
  kvadriati_path = "./Templates/TemplateGrids/tiles/",
  radii_path     = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path  = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path  = "./Templates/TemplateRasters/LV100m_10km.tif",
  input_layers   = c("./RasterGrids_100m/2024/RAW/Terrain_Dis-area_cell.tif"),
  layer_prefixes = c("Terrain_Dis-area"),

```

```

output_dir  = "./RasterGrids_100m/2024/RAW/",
n_workers   = 5,
radii       = c("r1250"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight  = 2,
future_max_size = 5 * 1024^3)

# Terrain_Dis-area_r1250.tif    egv_531
slanis=rast("./RasterGrids_100m/2024/RAW/Terrain_Dis-area_r1250.tif")
names(slanis)="egv_531"
slanis2=project(slanis,template100)
writeRaster(slanis2,
            "./RasterGrids_100m/2024/RAW/Terrain_Dis-area_r1250.tif",
            overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Terrain_Dis-area_r1250.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.532 Terrain\_Dis-area\_r3000

**filename:** Terrain\_Dis-area\_r3000.tif

**layername:** egv\_532

**English name:** Fractional cover of Terrain Sinks within the 3 km landscape

**Latvian name:** Reljefa depresiju bez virszemes noteceš plātības īpatsvars 3 km ainavā

**Procedure:** The cover fraction within a radius of 3000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",
  radii_path    = "./Templates/TemplateGridPoints/tiles/",
  tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
  template_path = "./Templates/TemplateRasters/LV100m_10km.tif",

```

```

input_layers = c("./RasterGrids_100m/2024/RAW/Terrain_Dis-area_cell.tif"),
layer_prefixes = c("Terrain_Dis-area"),
output_dir = "./RasterGrids_100m/2024/RAW/",
n_workers = 5,
radii = c("r3000"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 5 * 1024^3)

# Terrain_Dis-area_r3000.tif    egv_532
slanis=rast("./RasterGrids_100m/2024/RAW/Terrain_Dis-area_r3000.tif")
names(slanis)="egv_532"
slanis2=project(slanis,template100)
writeRaster(slanis2,
    "./RasterGrids_100m/2024/RAW/Terrain_Dis-area_r3000.tif",
    overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Terrain_Dis-area_r3000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
    filename=saglabasanas_cels,
    overwrite=TRUE)

```

## 6.533 Terrain\_Dis-area\_r10000

**filename:** Terrain\_Dis-area\_r10000.tif

**layername:** egv\_533

**English name:** Fractional cover of Terrain Sinks within the 10 km landscape

**Latvian name:** Reljefa depresiju bez virszemes noteceš platības īpatsvars 10 km ainavā

**Procedure:** The cover fraction within a radius of 10000 m around the analysis grid cell is calculated as the area-weighted sum of the **analysis cells** inside the buffer, using the workflow `egvtools::radius_function()`. During the calculation of the landscape metric, inverse distance weighted (power = 2) gap filling on the output is applied to ensure no missing values at the edges. Then the layer is rewritten to set its name. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# radii
radius_function(
  kvadrati_path = "./Templates/TemplateGrids/tiles/",

```

```

radii_path = "./Templates/TemplateGridPoints/tiles/",
tikls100_path = "./Templates/TemplateGrids/tikls100_sauzeme.parquet",
template_path = "./Templates/TemplateRasters/LV100m_10km.tif",
input_layers = c("./RasterGrids_100m/2024/RAW/Terrain_Dis-area_cell.tif"),
layer_prefixes = c("Terrain_Dis-area"),
output_dir = "./RasterGrids_100m/2024/RAW/",
n_workers = 5,
radii = c("r10000"),
radius_mode = "sparse",
extract_fun = "mean",
fill_missing = TRUE,
IDW_weight = 2,
future_max_size = 5 * 1024^3

# Terrain_Dis-area_r10000.tif  egv_533
slanis=rast("./RasterGrids_100m/2024/RAW/Terrain_Dis-area_r10000.tif")
names(slanis)="egv_533"
slanis2=project(slanis,template100)
writeRaster(slanis2,
           "./RasterGrids_100m/2024/RAW/Terrain_Dis-area_r10000.tif",
           overwrite=TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Terrain_Dis-area_r10000.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.534 Terrain\_Dis-max\_cell

**filename:** Terrain\_Dis-max\_cell.tif

**layername:** egv\_534

**English name:** Maximum Depth in Terrain Sink within the analysis cell (1 ha)

**Latvian name:** Reljefa depresiju lielākais dzīlums analīzes šūnā (1 ha)

**Procedure:** Derived from the [Terrain products](#). Processed using the workflow `egvtools::input2egv()`. Inverse distance weighted (power = 2) gap filling is implemented to protect against potential data loss at edge cells. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# Terrain_Dis-max_cell.tif  egv_534
input2egv(input="./RasterGrids_10m/2024/Terrain_Dis_udeni2_10m.tif",

```

```

egv_template="./Templates/TemplateRasters/LV100m_10km.tif",
summary_function = "max",
missing_job = "FillOutput",
idw_weight = 2,
outlocation = "./RasterGrids_100m/2024/RAW/",
outfilename = "Terrain_Dis-max_cell.tif",
layername="egv_534",
return_visible = TRUE,
plot_final = TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Terrain_Dis-max_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/RAW/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.535 Terrain\_Dis-mean\_cell

**filename:** Terrain\_Dis-mean\_cell.tif

**layername:** egv\_535

**English name:** Average Depth in Terrain Sink within the analysis cell (1 ha)

**Latvian name:** Reljefa depresiju vidējais dziļums analīzes šūnā (1 ha)

**Procedure:** Derived from the [Terrain products](#). Processed using the workflow `egvtools::input2egv()`. Inverse distance weighted (power = 2) gap filling is implemented to protect against potential data loss at edge cells. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# Terrain_Dis-mean_cell.tif egv_535
input2egv(input="./RasterGrids_10m/2024/Terrain_Dis_udeni2_10m.tif",
          egv_template="./Templates/TemplateRasters/LV100m_10km.tif",
          summary_function = "average",
          missing_job = "FillOutput",
          idw_weight = 2,
          outlocation = "./RasterGrids_100m/2024/RAW/",
          outfilename = "Terrain_Dis-mean_cell.tif",
          layername="egv_535",
          return_visible = TRUE,
          plot_final = TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

```

```

nosaukums="Terrain_Dis-mean_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/", nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/", nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis, fun="mean", na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets, fun="rms", na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.536 Terrain\_Slope-average\_cell

**filename:** Terrain\_Slope-average\_cell.tif

**layername:** egv\_536

**English name:** Average value of Terrain Slope (degree) within the analysis cell (1 ha)

**Latvian name:** Nogāzes slīpuma (grādi) vidējā vērtība analīzes šūnā (1 ha)

**Procedure:** Derived from the [Terrain products](#). Processed using the workflow `egvtools::input2egv()`. Inverse distance weighted (power = 2) gap filling is implemented to protect against potential data loss at edge cells. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# Terrain_Slope-average_cell.tif    egv_536
input2egv(input="./RasterGrids_10m/2024/Terrain_Slope_udeni2_10m.tif",
          egv_template="./Templates/TemplateRasters/LV100m_10km.tif",
          summary_function = "average",
          missing_job = "FillOutput",
          idw_weight = 2,
          outlocation = "./RasterGrids_100m/2024/Raw/",
          outfile = "Terrain_Slope-average_cell.tif",
          layername="egv_536",
          return_visible = TRUE,
          plot_final = TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Terrain_Slope-average_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/", nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/", nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis, fun="mean", na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets, fun="rms", na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
            filename=saglabasanas_cels,
            overwrite=TRUE)

```

## 6.537 Terrain\_Slope-iqr\_cell

**filename:** Terrain\_Slope-iqr\_cell.tif

**layername:** egv\_537

**English name:** Variability of Terrain Slope (degree) within the analysis cell (1 ha)

**Latvian name:** Nogāzes slīpuma (grādi) variabilitāte analīzes šūnā (1 ha)

**Procedure:** Derived from the [Terrain products](#). The workflow `egvtools::input2egv()` is used to calculate Q1 and Q3 for every cell. To protect against potential data loss at the edges, inverse distance weighted (power = 2) gap filling is implemented. Next, Q1 is subtracted from Q3. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```
# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# Terrain_Slope-iqr_cell.tif      egv_537
p25rez=input2egv(input="./RasterGrids_10m/2024/Terrain_Slope_udeni2_10m.tif",
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "q1",
                  missing_job = "FillOutput",
                  outlocation = "./RasterGrids_100m/2024/",
                  outfilename = "draza_p25.tif",
                  layername = "egv_537",
                  idw_weight = 2)
p25rez_r=rast("./RasterGrids_100m/2024/draza_p25.tif")

p75rez=input2egv(input="./RasterGrids_10m/2024/Terrain_Slope_udeni2_10m.tif",
                  egv_template= "./Templates/TemplateRasters/LV100m_10km.tif",
                  summary_function = "q3",
                  missing_job = "FillOutput",
                  outlocation = "./RasterGrids_100m/2024/",
                  outfilename = "draza_p75.tif",
                  layername = "egv_537",
                  idw_weight = 2)
p75rez_r=rast("./RasterGrids_100m/2024/draza_p75.tif")

iqr_rez=p75rez_r-p25rez_r
iqr_rez
plot(iqr_rez)

writeRaster(iqr_rez,
            "./RasterGrids_100m/2024/Raw/Terrain_Slope-iqr_cell.tif",
            overwrite=TRUE)

unlink("./RasterGrids_100m/2024/draza_p75.tif")
unlink("./RasterGrids_100m/2024/draza_p25.tif")

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Terrain_Slope-iqr_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
```

```

standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

## 6.538 Terrain\_TWI-average\_cell

**filename:** Terrain\_TWI-average\_cell.tif

**layername:** egv\_538

**English name:** Average value of Topographic Wetness Index (TWI) within the analysis cell (1 ha)

**Latvian name:** Topogrāfiskā mitruma indeksa (TWI) vidējā vērtība analīzes šūnā (1 ha)

**Procedure:** Derived from the [Terrain products](#). Processed using the workflow `egvtools::input2egv()`. Inverse distance weighted (power = 2) gap filling is implemented to protect against potential data loss at edge cells. Finally, the layer is standardised by subtracting the arithmetic mean and dividing by the root mean squared error.

```

# libs ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(egvtools)) {remotes::install_github("aavotins/egvtools"); require(egvtools)}

# templates ----
template100=rast("./Templates/TemplateRasters/LV100m_10km.tif")

# Terrain_TWI-average_cell.tif egv_538
input2egv(input=".~/RasterGrids_10m/2024/Terrain_TWI_udeni2_10m.tif",
  egv_template=".~/Templates/TemplateRasters/LV100m_10km.tif",
  summary_function = "average",
  missing_job = "FillOutput",
  idw_weight = 2,
  outlocation = ".~/RasterGrids_100m/2024/Raw/",
  outfilename = "Terrain_TWI-average_cell.tif",
  layername="egv_538",
  return_visible = TRUE,
  plot_final = TRUE)

# standardisation ----
if(!require(terra)) {install.packages("terra"); require(terra)}
if(!require(tidyverse)) {install.packages("tidyverse"); require(tidyverse)}

nosaukums="Terrain_TWI-average_cell.tif"
ielasisanas_cels=paste0("./RasterGrids_100m/2024/Raw/",nosaukums)
saglabasanas_cels=paste0("./RasterGrids_100m/2024/Scaled/",nosaukums)
slanis=rast(ielasisanas_cels)
videjais=global(slanis,fun="mean",na.rm=TRUE)
centrets=slanis-videjais[,1]
standartnovirze=terra::global(centrets,fun="rms",na.rm=TRUE)
merogots=centrets/standartnovirze[,1]
writeRaster(merogots,
  filename=saglabasanas_cels,
  overwrite=TRUE)

```

# **Chapter 7**

## **Data access**

When using code or data disclosed in this document, please cite our article and data repository:

- article reference: [to be added](#)
- repository reference: [to be added](#)

Standardised ecogeographical variables are available for download from the project's [Zenodo repository](#).

Layers can be interacted with in the Google Earth Engine [application](#).



# References

- Brown, C.F., Brumby, S.P., Guzder-Williams, B., Birch, T., Hyde, S.B., Mazzariello, J., Czerwinski, W., Pasquarella, V.J., Haertel, R., Ilyushchenko, S., Schwehr, K., Weisse, M., Stolle, F., Hanson, C., Guinan, O., Moore, R., Tait, A.M., 2022. Dynamic World, Near real-time global 10 m land use land cover mapping. *Scientific Data* 9, 251. <https://doi.org/10.1038/s41597-022-01307-4>
- Domisch, S., Amatulli, G., Jetz, W., 2015. Near-global freshwater-specific environmental variables for biodiversity analyses in 1 km resolution. *Scientific Data* 2:150073, 1–13. <https://doi.org/10.1038/sdata.2015.73>
- Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., Moore, R., 2017. Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment* 202, 18–27. <https://doi.org/10.1016/j.rse.2017.06.031>
- Hansen, M.C., Potapov, P.V., Moore, R., Hancher, M., Turubanova, S.A., Tyukavina, A., Thau, D., Stehman, S.V., Goetz, S.J., Loveland, T.R., Kommareddy, A., Egorov, A., Chini, L., Justice, C.O., Townshend, J.R.G., 2013. High-resolution Global maps of 21st-century forest cover change. *Science* 342, 850–853. <https://doi.org/10.1126/science.1244693>
- Hijmans, R.J., Cameron, S.E., Parra, J.L., Jones, P.G., Jarvis, A., 2005. Very high resolution interpolated climate surfaces for global land areas. *International Journal of Climatology* 25, 1965–1978. <https://doi.org/10.1002/joc.1276>
- Karger, D.N., Conrad, O., Böhner, J., Kawohl, T., Kreft, H., Soria-Auza, R.W., Zimmermann, N.E., Linder, H.P., Kessler, M., 2017. Data Descriptor: Climatologies at high resolution for the earth's land surface areas. *Scientific Data* 4:170122. <https://doi.org/10.1038/sdata.2017.122>
- Lehner, B., Grill, G., 2013. Global river hydrography and network routing: Baseline data and new approaches to study the world's large river systems. *Hydrological Processes* 27, 2171–2186. <https://doi.org/10.1002/hyp.9740>
- Lehner, B., Verdin, K., Jarvis, A., 2008. New global hydrography derived from spaceborne elevation data. *Eos, Transactions, American Geophysical Union* 89, 93–94. <https://doi.org/10.1029/2008EO100001>
- Panagos, P., Liedekerke, M.V., Borrelli, P., Königer, J., Ballabio, C., Orgiazzi, A., Lugato, E., Liakos, L., Hervas, J., Jones, A., Montanarella, L., 2022. European Soil Data Centre 2.0: Soil data and knowledge in support of the EU policies. *European Journal of Soil Science* 73, e13315. <https://doi.org/10.1111/ejss.13315>
- Shimada, M., Itoh, T., Motooka, T., Watanabe, M., Shiraishi, T., Thapa, R., Lucas, R., 2013. New global forest/non-forest maps from ALOS PALSAR data (2007–2010). *Remote Sensing of Environment* 155, 13–31. <https://doi.org/10.1016/j.rse.2014.04.014>
- Wang, L., Liu, H., 2006. An efficient method for identifying and filling surface depressions in digital elevation models for hydrologic analysis and modelling. *International Journal of Geographical Information Science* 20, 193–213. <https://doi.org/10.1080/13658810500433453>