Final Project - GIS

Thomas Beier, Florian Franz, Konstantin Seeger

10.04.2022

Spatial prediction of forest microclimate using LiDAR data

Using the `lidr` package by J.-R. Roussel for processing LiDAR data and creating a forest microclimate model.

For information see the book (https://r-lidar.github.io/lidRbook/index.html), the package documentation (https://cran.r-project.org/web/packages/lidR/index.html) and also this publication (https://www.sciencedirect.com/science/article/pii/S0034425720304314).

Read las file

```
# The las file must be in the folder data/raw!!!
las_files <- list.files(envrmt$path_raw,
                        pattern = glob2rx("*.las"),
                        full.names = TRUE)

las <- readLAS(las_files[1])

# Assign a coord. ref. syst. (CRS)
# In this case UTM zone 32N
epsg_number <- 25832
crs(las) <- epsg_number

# Check las file
las_check(las)
```

```
### Some plotting...

# Basic 3D plot
# plot(las)

# Example cross section 2D plot (along a transect)
p1 <- c(477500, 5632500) # these are coordinates
p2 <- c(478217.5, 5632500 ) # these are coordinates
las_tr <- clip_transect(las, p1, p2, width = 4, xz = TRUE)

ggplot(las_tr@data, aes(X,Z, color = Z)) +
  geom_point(size = 0.5) +
  coord_equal() +
  theme_minimal() +
  scale_color_gradientn(colours = height.colors(25))
```

Predictor set

List of variables used for forest microclimate prediction:

- Canopy height (CHM)
- Standard metrics (mean, max, sd…)
- Mean height of first returns
- Maxiumum height of first returns
- Standard deviation of first returns
- Point density
- Pulse density
- Leaf area index (LAI)
- Elevation (DTM)
- Slope
- Exposition
- Topographic position index (TPI)

All of the predictors are calculated at the grid level within 1 m x 1 m pixels.

Canopy height model (CHM)

```r
# Height normalization within the point cloud using invert distance weighting
norm_las <- normalize_height(las, knnidw())

# Check if all ground points are 0
hist(filter_ground(norm_las)$Z,
     breaks = seq(-0.45, 0.45, 0.01),
     main = "", xlab = "Elevation")

if (!file.exists(paste0(envrmt$path_predictors, "/chm.RDS"))) {
  # Calculate CHM using pit-free algorithm (we can discuss about this
  #--> which algorithm we want to use)
  chm <- grid_canopy(norm_las, res = 1.0,
                     pitfree(thresholds = c(0, 2, 5, 10, 15),
                             max_edge = c(0, 1.5)))

  saveRDS(chm, paste0(envrmt$path_predictors, "/chm.RDS"))

} else {

  chm <- readRDS(paste0(envrmt$path_predictors, "/chm.RDS"))

}
```

```r
# tmap plot
library(tmap)
tm_shape(chm) +
tm_raster(title = "Pitfree CHM 1 m² cells", palette = height.colors(20)) +
  tm_grid() +
  tm_layout(legend.outside = TRUE)
```

Standard metrics of the canopy

```r
if (!file.exists(paste0(envrmt$path_predictors, "/stdmetrics.RDS"))) {

  chm_stdmetrics <- grid_metrics(norm_las, .stdmetrics, res = 1.0)

  saveRDS(chm_stdmetrics, paste0(envrmt$path_predictors, "/stdmetrics.RDS"))

} else {

  chm_stdmetrics <- readRDS(paste0(envrmt$path_predictors, "/stdmetrics.RDS"))

}

# plot(chm_stdmetrics, col = height.colors(20))
```

First returns mean height, maxiumum height and standard deviation of the height from the first returns

```r
# Function for calculating mean, max and standard deviation of first returns
first_return_metrics <- function(x) {
  list(mean = mean(x),
       max = max(x),
       sd(x))
}


if (!file.exists(paste0(envrmt$path_predictors, "/first_returns.RDS"))) {

  return_mean_max_sd <- grid_metrics(norm_las, func = ~first_return_metrics(Z),
                                     res = 1.0, filter = ~ReturnNumber == 1L)

  saveRDS(return_mean_max_sd, paste0(envrmt$path_predictors, "/first_returns.RDS"))

} else {

  return_mean_max_sd <- readRDS(paste0(envrmt$path_predictors, "/first_returns.RDS"))

}
```

Point and pulse density

```r
if (!file.exists(paste0(envrmt$path_predictors, "/point_density.RDS"))) {

  point_density <- grid_metrics(norm_las, ~length(Z)/1, res = 1.0)
  pulse_density <- grid_metrics(norm_las, ~length(Z)/1, res = 1.0,
                                filter = ~ReturnNumber == 1L)

  saveRDS(point_density, paste0(envrmt$path_predictors, "/point_density.RDS"))
  saveRDS(pulse_density, paste0(envrmt$path_predictors, "/pulse_density.RDS"))

} else {

  point_density <- readRDS(paste0(envrmt$path_predictors, "/point_density.RDS"))
  pulse_density <- readRDS(paste0(envrmt$path_predictors, "/pulse_density.RDS"))

}
```

Leaf area index (LAI)

```
# Calculate Leaf area index with the "canopyLazR" package. Installation can be
# found on the following github page: https://github.com/akamoske/canopyLazR
# Convert .laz or .las file into a voxelized lidar array

if (!file.exists(paste0(envrmt$path_predictors, "/lai.tif"))) {

  laz_data <- laz.to.array(laz.file.path = file.path(envrmt$path_raw,"las_mof.las"),
                          voxel.resolution = 1,
                          z.resolution = 1,
                          use.classified.returns = TRUE)

  # Level the voxelized array to mimic a canopy height model
  level_canopy <- canopy.height.levelr(lidar.array = laz_data)

  # Estimate Leaf Area Density (LAD) for each voxel in leveled array
  lad_estimates <- machorn.lad(leveld.lidar.array = level_canopy,
                              voxel.height = 1,
                              beer.lambert.constant = NULL)

  # Convert the LAD array into a single raster stack
  lad_raster <- lad.array.to.raster.stack(lad.array = lad_estimates,
                                          laz.array = laz_data,
                                          epsg.code = 25832)


  # Create a single LAI raster from the LAD raster stack
  lai_raster <- raster::calc(lad_raster, fun = sum, na.rm = TRUE)



  saveRDS(lai_raster, paste0(envrmt$path_predictors, "/lai.tif"))

} else {

  lai_raster <- raster(paste0(envrmt$path_predictors, "/lai.tif"))

}
```

Elevation

```r
# Calculate DTM using invert distance weighting
if (!file.exists(paste0(envrmt$path_predictors, "/dtm.RDS"))) {

  dtm <- grid_terrain(las, res = 1.0, algorithm = knnidw(k = 6L, p = 2))

  saveRDS(dtm, paste0(envrmt$path_predictors, "/dtm.RDS"))

} else {

  dtm <- readRDS(paste0(envrmt$path_predictors, "/dtm.RDS"))

}
```

Slope, exposition and TPI

```r
if (!file.exists(paste0(envrmt$path_predictors, "/slope.RDS"))) {

  slope <- raster::terrain(dtm, opt = "slope", unit = "degrees", neighbors = 8)
  exposition <- raster::terrain(dtm, opt = "aspect", unit = "degrees", neighbors = 8)
  tpi <- raster::terrain(dtm, opt = "tpi")

  saveRDS(slope, paste0(envrmt$path_predictors, "/slope.RDS"))
  saveRDS(exposition, paste0(envrmt$path_predictors, "/exposition.RDS"))
  saveRDS(tpi, paste0(envrmt$path_predictors, "/tpi.RDS"))

} else {

  slope <- readRDS(paste0(envrmt$path_predictors, "/slope.RDS"))
  exposition <- readRDS(paste0(envrmt$path_predictors, "/exposition.RDS"))
  tpi <- readRDS(paste0(envrmt$path_predictors, "/tpi.RDS"))

}
```

Create final predictor stack with temperature (from the climate station data) as response variable

```r
# Create a raster stack of all predictors
if (!file.exists(paste0(envrmt$path_predictors, "/predictor_stack.RDS"))) {

  predictors <- raster::stack(chm, chm_stdmetrics, return_mean_max_sd,
                              point_density, pulse_density, lai_raster,
                              dtm, slope, exposition, tpi)

  # Rename layer names --> only where it's necessary
  predictors@layers[[1]]@data@names <- "chm"
  predictors@layers[[58]]@data@names <- "return_mean"
  predictors@layers[[59]]@data@names <- "return_max"
  predictors@layers[[60]]@data@names <- "return_sd"
  predictors@layers[[61]]@data@names <- "point_density"
  predictors@layers[[62]]@data@names <- "pulse_density"
  predictors@layers[[63]]@data@names <- "lai"
  predictors@layers[[64]]@data@names <- "dtm"

  saveRDS(predictors, file.path(envrmt$path_predictors, "/predictor_stack.RDS"))

} else {

  predictors <- readRDS(paste0(envrmt$path_predictors, "/predictor_stack.RDS"))

}

if (!file.exists(paste0(envrmt$path_processed, "/df_predictors_response.RDS"))) {

  # Read climate data and select timespan (June, July, August)
  climate_data <- readRDS(paste0(envrmt$path_raw, "/climate_stations_combined.RDS"))
  climate_data <- climate_data[order(climate_data$date),]
  climate_data <- climate_data %>%
    filter(date >= "2020-06-01 00:00:00" & date <= "2020-08-31 23:00:00")

  # Read shapefile core study trees
  trees <- sf::read_sf(paste0(envrmt$path_raw, "/core_study_trees.shp"))

  # Merge climate data and trees based on the tree_id
  climate_data <- climate_data %>% rename(tree_id = cst_id)
  trees_climate_merged <- merge(climate_data, trees, by = "tree_id")
  trees_climate_merged <- trees_climate_merged[order(trees_climate_merged$date),]

  # Create one final DataFrame with the predictor values for each station
  df_final <- data.frame(matrix(ncol = 67, nrow =  nrow(trees_climate_merged)))
  colnames(df_final) <- names(predictors)

  for (i in 1:67) { # change to 67 if LAI is included

    df_final[i] <- raster::extract(predictors[[i]], sf::st_as_sf(trees_climate_merged))

    }

  # Add temperature and the tree_id
  df_final$temp <- trees_climate_merged$temp
  df_final$tree_id <- trees_climate_merged$tree_id
  df_final$date <- trees_climate_merged$date

  # Save as RDS
  saveRDS(df_final, file.path(envrmt$path_processed, "/df_predictors_response.RDS"))

} else {

  df_final <- readRDS(paste0(envrmt$path_processed, "/df_predictors_response.RDS"))

}
```

```r
# Add climate station
clim_stat <- readRDS(paste0(envrmt$path_raw, "/klimastation_wiese_hourly.RDS"))
clim_stat <- clim_stat[,-c(14,24)]
df_final_merged <- merge(df_final, clim_stat, by.x = "date", by.y = "date_time_hourly")
```

```r
# Adding to the raster predictor stack the wiese station as a raster where every
# value is the same. Time used: 2020-06-12 18:00:00
if (!file.exists(paste0(envrmt$path_processed, "/predictor_stack_plus_wiese.RDS"))) {

  # Raster stack with predictors + wiese station
  predictor_stack <- predictors

  for (b in 2:29){
    create_raster <- raster(ncol=ncol(predictor_stack), nrow=nrow(predictor_stack),
                            ext = extent(predictor_stack),crs = crs(predictor_stack))
    values(create_raster) <- clim_stat[[19,b]]
    names(create_raster) <- names(clim_stat[b])
    predictor_stack <- addLayer(predictor_stack, create_raster)
  }

  saveRDS(predictor_stack, file.path(envrmt$path_processed,
                                     "/predictor_stack_plus_wiese.RDS"))

} else {

  predictor_stack <- readRDS(paste0(envrmt$path_processed,
                                    "/predictor_stack_plus_wiese.RDS"))

}
```

```r
# Using the wiese station as independent validation station.

if (!file.exists(paste0(envrmt$path_processed, "/df_predictors_wiese.RDS"))) {
  # Assigning coordinates of wiese station
  northing <- 5632136
  easting <- 477694

  # adding them to the df
  clim_stat2 <- st_sfc(st_point(cbind(easting, northing)),crs = epsg_number)

  clim_stat$geometry <- clim_stat2

  # Extract the predictors from predictor stack
  df_predictors_wiese <- data.frame(matrix(ncol = 67, nrow =  nrow(clim_stat)))
  colnames(df_predictors_wiese) <- names(predictors)

  # Extract the predictor values
  for (i in 1:67) {

    df_predictors_wiese[i] <- raster::extract(predictors[[i]], sf::st_as_sf(clim_stat))

  }
  df_predictors_wiese$date <- clim_stat$date_time_hourly

  # add the predictors from the station
  df_predictors_wiese_merged <- merge(df_predictors_wiese, clim_stat,
                                      by.x = "date", by.y = "date_time_hourly")

  # Remove geometry column which is unneeded
  df_predictors_wiese_merged$geometry <- NULL

  df_predictors_wiese_merged$zkurt <- 0
  df_predictors_wiese_merged$zentropy <- 0

  df_predictors_wiese_merged <- df_predictors_wiese_merged[1:1921,]

  saveRDS(df_predictors_wiese_merged, file.path(envrmt$path_processed,
                                                "/df_predictors_wiese.RDS"))

} else {

  df_predictors_wiese_merged <- readRDS(paste0(envrmt$path_processed,
                                               "/df_predictors_wiese.RDS"))
  df_predictors_wiese_merged <- df_predictors_wiese_merged[1:1921,]
}
```

Cleaning the data

```
if (!file.exists(paste0(envrmt$path_model_training_data, "/df_clean_with_station.RDS"))) {
  # Removing NA's
  df_final_na <- df_final_merged[rowSums(is.na(df_final_merged[ , c(2:68,71:98)])) == 0, ]

  # Balancing the data so that from every station the same amount of data is used
  downTrainDF <- downSample(x = df_final_na[, -70],y = as.factor(df_final_na$tree_id),
                            yname = "tree_id")

  traintmp = downTrainDF
  # filter zero or near-zero values
  nzv = nearZeroVar(traintmp)
  if (length(nzv) > 0) traintmp = traintmp[, -nzv]

  # Removing rows with "inf" value
  traintmp <- traintmp[!is.infinite(rowSums(traintmp[,c(2:67,69:92)])),]

  # Remove temporary the non-predictor columns.
  traintmp_rmv = traintmp[ , !(names(traintmp) %in% c("temp","date","tree_id"))]
  tDF = traintmp

  # filter correlations that are > cor_cutoff
  filt = findCorrelation(cor(traintmp_rmv, use = "complete"), cutoff = 0.9)
  traintmp_rmv = traintmp_rmv[,-filt]
  # re-add the necessary variables for model training
  traintmp_rmv$temp = tDF$temp
  traintmp_rmv$tree_id = tDF$tree_id
  traintmp_rmv$date = tDF$date
  # remove rows with NA
  traintmp_rmv = traintmp_rmv[complete.cases(traintmp_rmv) ,]

  # check manually if there are still NA values around
  summary(traintmp_rmv)
  sapply(traintmp_rmv, function(y) sum(length(which(is.na(y)))))

  df_final <- traintmp_rmv

  saveRDS(df_final,paste0(envrmt$path_model_training_data,
                          "/df_clean_with_station.RDS"))

} else {

  df_final <- readRDS(paste0(envrmt$path_model_training_data,
                             "/df_clean_with_station.RDS"))
}
```

Split dataset into train and test data

```
if (!file.exists(file.path(envrmt$path_model_training_data,
                           "df_train_with_station.RDS"))) {

  # Set seed
  set.seed(11)

  # Split data into 80% for training and 20% for testing
  train_index <- caret::createDataPartition(df_final$temp, p = 0.8, list = FALSE)
  training <- df_final[train_index,]
  testing <- df_final[-train_index,]
  saveRDS(training, file.path(envrmt$path_model_training_data,
                              "df_train_with_station.RDS"))
  saveRDS(testing, file.path(envrmt$path_model_training_data,
                             "df_test_with_station.RDS"))

} else {

  training <- readRDS(file.path(envrmt$path_model_training_data,
                                "df_train_with_station.RDS"))
  testing <- readRDS(file.path(envrmt$path_model_training_data,
                               "df_test_with_station.RDS"))
}
```

Function that removes random amount of TreeTalker, calculates a random forest ranger model and calculates the RMSE for the weather station and each TreeTalker.

```
# Function to calculate rmse with less code
rmse <- function(predicted,observed,round = 2){
  return(round(sqrt(mean((predicted - observed)^2, na.rm = TRUE)), round))
```

```r
}

# Function to remove stations, calculate a ranger model and calculate average rmse.
#More info in function description and return.
model_results <- function(training_data,testing_data,wiese_validation,
                          predictor_stack_full,nr_stations_out,first_seed,
                          second_seeds,validation_runs,rounding = 2){
  #'@title model_results.
  #'@description Function calculates a ML ranger model
  #'while a definite amount of stations are removed.
  #'@param training_data Training data for the ML model.
  #'@param testing_data Testing data for the ML model.
  #'@param wiese_validation Independent wiese station.
  #'@param predictor_stack_full Raster predictor stack that includes the lidar predictors
  #'and the predictors from the wiese station. It is used to predict the microclimate for
  #'the whole study area.
  #'@param nr_stations_out Number of stations that will randomly be removed.
  #'@param first_seed One seed that is used for randomly removing tree stations.
  #'@param second_seeds Needs to be the same number of seeds as "validation_runs".
  #'These seeds are used for different model runs.
  #'@param validation_runs Number of runs the model will be excecuted for the same
  #'removed stations. In our case this number should usually be 3.
  #'@param rounding Is standard "2". RMSE value results are rounded to this position
  #'after decimal point.
  #'@return Function returns a dataframe that gives the average rmse value for number
  #'of validation runs. It returns the RMSE of all stations, the stations removed, the
  #'stations still in and also the RMSE of each station. It also provides the used seeds
  #'so that they are reproducible and the number of stations left out and which these
  #'stations are.

  # Selecting randomly tree stations
  tree_id <- unique(as.character(training_data$tree_id))
  set.seed(first_seed)
  stations_remove <- sample(tree_id,nr_stations_out)


  # Removing the rows with the selected stations
  for (k in seq(length(stations_remove))){
    training_data <- training_data[training_data$tree_id != stations_remove[k],]
    training_data <- droplevels(training_data)
  }

  # Create a dataframe in which the different results will be saved.
  df_matrix <- matrix(ncol = 1, nrow = validation_runs)
  df_results <- data.frame(df_matrix)

  # Define predictors and response variable
  predictors <- training_data[,c(1:46)]

  response <- training_data[,"temp"]

  seeds = second_seeds
```

```r
# Itterate over the seeds
for (x in 1:validation_runs){

  # Define parameters for the LLOCV.
  llocv <- CreateSpacetimeFolds(training_data, spacevar = "tree_id",
                                k = 10, class = "temp")

  # Control the parameters for later training
  # --> the folds of the LLOCV are passed as an index
  set.seed(seeds[x])
  ctrl <- trainControl(method = "cv", index = llocv$index,
                   savePredictions = TRUE, allowParallel = TRUE)

  # Control the parameters for model tuning
  # Tuneable parameter for random forests in the package 'ranger':
  # mtry = Number of variables to possibly split at in each node
  # splitrule = Splitting rule. For classification "gini", "extratrees" or "hellinger"
  # min.node.size = Minimal node size
  # https://www.rdocumentation.org/packages/ranger/versions/0.13.1/topics/ranger
  # http://topepo.github.io/caret/available-models.html
  tgrid <- expand.grid(
    mtry = c(5:10),
    splitrule = "extratrees",
    min.node.size = c(5,10,15)
  )

  set.seed(seeds[x])

  # Run a model
  model <- train(predictors,
        response,
        method = "ranger",
        metric = "RMSE",
        num.trees = 100,
        tuneGrid = tgrid,
        trControl = ctrl,
        importance = 'permutation')

  df_results$run[x] <- x
  df_results$first_seed[x] <- first_seed
  df_results$second_seed[x] <- seeds[x]

  # Deleting unneeded column
  if ("df_matrix" %in% colnames(df_results)){
    df_results$df_matrix <- NULL
  }

  # Predict TreeTalker loggers
  predicted = stats::predict(object = model, newdata = testing_data)

  val_df = data.frame(ID = dplyr::pull(testing_data, "tree_id"),
                      Observed = dplyr::pull(testing_data, "temp"),
                      Predicted = predicted)
```

```r
# Predict whole study area
predicted_raster <- raster::predict(predictor_stack_full, model,
                                    na.rm = TRUE,progress = "text")

writeRaster(predicted_raster,paste0(envrmt$path_prediction,"/",
                                    length(stations_remove),"/prediction_",
                                    first_seed,"_",length(stations_remove),
                                    "_",x,".tif"))

df_results$amount_station_out[x] <- length(stations_remove)
df_results$left_out[x] <- list(substr(stations_remove, 12, 13))

df_results$rmse_gesamt[x] <- rmse(val_df$Predicted,val_df$Observed,rounding)

# Predict weather station
predicted_wiese = stats::predict(object = model, newdata = wiese_validation)

val_df_wiese = data.frame(ID = dplyr::pull(wiese_validation, "date"),
                  Observed = dplyr::pull(wiese_validation, "Ta_10m"),
                  Predicted = predicted_wiese)

df_results$rmse_wiese_station[x] <- rmse(val_df_wiese$Predicted,
                                         val_df_wiese$Observed,rounding)


# Calculate RMSE of removed loggers
rmse_removed_stations <- val_df[val_df$ID %in% stations_remove, ]
df_results$rmse_left_out[x] <- rmse(rmse_removed_stations$Predicted,
                                    rmse_removed_stations$Observed,rounding)

# Calculate RMSE of non-removed loggers.
rmse_stations_in <- val_df[!val_df$ID %in% stations_remove, ]
df_results$rmse_station_in[x] <- rmse(rmse_stations_in$Predicted,
                                      rmse_stations_in$Observed,rounding)

df_rmse_stat <- data.frame(station = unique(val_df$ID))

# Calculate the RMSE for each individual station
for (n in 1:length(unique(val_df$ID))){
  rmse_removed_stations_si <- val_df[val_df$ID %in%
                                       df_rmse_stat$station[n], ]
  stationen_rmse <- rmse(rmse_removed_stations_si$Predicted,
                         rmse_removed_stations_si$Observed)
  df_rmse_stat$rmse[n] <- stationen_rmse
}

tabs_zsm <- xtabs(rmse~station, df_rmse_stat)

# Add to a new dataframe
df_table <- as.data.frame.matrix(t(tabs_zsm))

# Connect the results with the results from the previous dataframe
```

```
    if (x == 1){
      df_results <- cbind(df_results, df_table)
    } else if (x > 1){
      # Adding the rmse values of each station to dataframe
      for (i in 1:18){
        rmse_each_station <- df_table[[i]]
        df_results[x,i+9] <- rmse_each_station
      }
    }

  }

  # Mean the results of the different seeds for the same omitted stations
  df_mean_results <- df_results[1,c(4:5)]
  df_mean_results$first_seed <- df_results$first_seed[1]
  df_mean_results$second_seeds <- list(df_results$second_seed)
  df_mean_results[,5:26] <- sapply(df_results[,c(6:27)],FUN=mean)
  names(df_mean_results)[5:26] <- names(df_results[c(6:27)])

  return(df_mean_results)

}
```

```
if (!file.exists(paste0(envrmt$path_prediction, "/model_results.RDS"))) {

  # Testing the function.
  result <- model_results(training,testing,df_predictors_wiese_merged,
                          predictor_stack,2,98,c(64,13,649),3,2)

  # This result will be saved in the "prediction" folder as "model_results.RDS"
  # and each new model run can be added to this dataframe.

  saveRDS(result, paste0(envrmt$path_prediction, "/model_results.RDS"))

  df_full_results <- readRDS(paste0(envrmt$path_prediction,
                                    "/model_results.RDS"))

  # Now the dataframe where all the results are stored is loaded and by calling
  # the "model_results" function and afterwards using the "rbind" function as
  # demonstrated below you can just add all the results into one dataframe.

  result <- model_results(training,testing,df_predictors_wiese_merged,
                          predictor_stack,2,42,c(29,18,98),3,2)

  # Add the new results to the exisiting dataframe.
  df_full_results <- rbind(df_full_results, result)# "results" is the output of
  #the "model_results" function and "df_full_results" the loaded
  # result dataframe.

  # Save the updated results
  #saveRDS(df_full_results, paste0(envrmt$path_prediction,
  #"/model_results.RDS"))
```

```
result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,2,12,c(6,81,45),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,2,75,c(46,87,38),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,2,531,c(12,21,82),3,2)

df_full_results <- rbind(df_full_results, result)


# 8 stations removed
result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,8,67,c(41,53,19),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,8,81,c(61,28,43),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,8,80,c(2,51,78),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,8,103,c(70,99,19),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,8,19,c(25,5,95),3,2)

df_full_results <- rbind(df_full_results, result)

# 16 Stations removed
result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,16,97,c(23,18,37),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,16,9,c(65,50,7),3,2)

df_full_results <- rbind(df_full_results, result)
```

```r
result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,16,53,c(12,75,14),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,16,86,c(86,91,71),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,16,100,c(25,19,79),3,2)

df_full_results <- rbind(df_full_results, result)

# 1 Station removed

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,1,3,c(56,12,92),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,1,72,c(35,39,48),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,1,39,c(71,14,65),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,1,59,c(43,95,53),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,1,18,c(62,72,16),3,2)

df_full_results <- rbind(df_full_results, result)

# 17 Stations removed

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,17,71,c(64,24,13),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,17,51,c(75,43,68),3,2)

df_full_results <- rbind(df_full_results, result)
```

```r
result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,17,79,c(76,1,7),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,17,30,c(24,56,95),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,17,4,c(65,34,13),3,2)

df_full_results <- rbind(df_full_results, result)

# 3 Stations removed

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,3,97,c(53,18,74),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,3,819,c(27,83,62),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,3,64,c(41,34,5),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,3,17,c(14,64,41),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,3,85,c(63,75,89),3,2)

df_full_results <- rbind(df_full_results, result)

# 4 Stations removed

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,4,10,c(14,54,23),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,4,43,c(61,73,52),3,2)

df_full_results <- rbind(df_full_results, result)
```

```
result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,4,96,c(59,53,41),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,4,32,c(25,74,24),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,4,61,c(94,47,40),3,2)

df_full_results <- rbind(df_full_results, result)

# 5 Stations removed

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,5,59,c(94,33,29),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,5,18,c(90,11,4),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,5,73,c(46,42,91),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,5,51,c(65,24,25),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,5,96,c(52,64,86),3,2)

df_full_results <- rbind(df_full_results, result)

# 6 Stations removed

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,6,91,c(64,15,51),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,6,51,c(35,41,6),3,2)

df_full_results <- rbind(df_full_results, result)
```

```
result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,6,62,c(45,34,98),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,6,89,c(83,56,51),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,6,73,c(75,86,13),3,2)

df_full_results <- rbind(df_full_results, result)

# 7 Stations removed

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,7,10,c(25,46,81),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,7,23,c(52,71,59),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,7,52,c(3,99,64),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,7,29,c(93,45,22),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,7,74,c(74,63,86),3,2)

df_full_results <- rbind(df_full_results, result)

# 9 Stations removed

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,9,26,c(38,73,7),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,9,95,c(12,56,41),3,2)

df_full_results <- rbind(df_full_results, result)
```

```r
result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,9,85,c(14,43,11),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,9,63,c(53,64,74),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,9,52,c(85,20,40),3,2)

df_full_results <- rbind(df_full_results, result)

# 10 Stations removed

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,10,12,c(23,32,81),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,10,27,c(92,99,4),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,10,14,c(29,49,64),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,10,85,c(11,31,50),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,10,73,c(46,51,79),3,2)

df_full_results <- rbind(df_full_results, result)

# 11 Stations removed

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,11,61,c(25,13,72),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,11,91,c(54,12,52),3,2)

df_full_results <- rbind(df_full_results, result)
```

```
result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,11,65,c(62,73,25),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,11,83,c(81,13,62),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,11,8,c(73,98,23),3,2)

df_full_results <- rbind(df_full_results, result)

# 12 Stations removed

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,12,19,c(92,27,82),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,12,75,c(52,95,89),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,12,24,c(11,53,78),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,12,96,c(82,52,42),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,12,85,c(61,64,39),3,2)

df_full_results <- rbind(df_full_results, result)

# 13 Stations removed

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,13,98,c(24,47,96),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,13,59,c(14,74,97),3,2)

df_full_results <- rbind(df_full_results, result)
```

```
result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,13,2,c(41,51,12),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,13,34,c(30,19,51),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,13,64,c(91,61,14),3,2)

df_full_results <- rbind(df_full_results, result)

# 14 Stations removed

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,14,11,c(54,12,95),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,14,6,c(26,62,94),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,14,22,c(61,51,16),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,14,29,c(4,73,95),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,14,86,c(89,44,75),3,2)

df_full_results <- rbind(df_full_results, result)

# 15 Stations removed

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,15,81,c(63,14,52),3,2)

df_full_results <- rbind(df_full_results, result)

result <- model_results(training,testing,df_predictors_wiese_merged,
                        predictor_stack,15,33,c(93,13,53),3,2)

df_full_results <- rbind(df_full_results, result)
```

```
  result <- model_results(training,testing,df_predictors_wiese_merged,
                          predictor_stack,15,48,c(84,61,41),3,2)

  df_full_results <- rbind(df_full_results, result)

  result <- model_results(training,testing,df_predictors_wiese_merged,
                          predictor_stack,15,92,c(11,41,72),3,2)

  df_full_results <- rbind(df_full_results, result)

  result <- model_results(training,testing,df_predictors_wiese_merged,
                          predictor_stack,15,16,c(81,17,70),3,2)

  df_full_results <- rbind(df_full_results, result)


} else {

  df_full_results <- readRDS(paste0(envrmt$path_prediction,
                                    "/model_results.RDS"))

}
```

```
# Calculating the mean of the raster layers. Averaging because for every
# combination three runs with different seeds were performed.

#After performing this code chunk the .tif files were removed (copied it to
# another folder in case we will need them again) and only the mean .tif files
# are kept (the results from this for-loop)

for (p in 1:17){

  subset_station_out <- df_full_results[which(
    df_full_results$amount_station_out == p), ]

  first_seeds <- subset_station_out$first_seed

  for (z in first_seeds){

    # Listing all .tif files in the folder
    files <- list.files(path=paste0(envrmt$path_prediction,"/",p,"/"),
                        pattern="*.tif$", full.names=F, recursive=FALSE)

    # Select all the data that matches with the string "Extract_string"
    Extract_string <- paste0("prediction_",z,"_")
    matches <- unique (grep(paste(Extract_string,collapse="|"),
                            files, value=TRUE))

    # Create raster stack from the selected .tif files
    prediction_stack <- stack(paste0(envrmt$path_prediction,"/",p,"/",
                                     matches,sep=''))

    # Calculate the mean from the raster stack
    prediction_mean <- calc(prediction_stack, fun = mean, na.rm = T)

    # Export the tif file
    writeRaster(prediction_mean,paste0(envrmt$path_prediction,"/",p,
                                       "/prediction_",z,"_",p,"_mean.tif"))
  }
}
# The .tif file names are based on the first seed (first number in the filename)
# and the number of left out stations (second number in the filenames)
```

```
# Plotting barplot of the rmse of the weather station and the boxplot
if (!file.exists(file.path(envrmt$path_prediction, "rmse_wiese_boxplot.png"))) {

  # Calculate mean RMSE per amount of stations left out for the 'Wiese station'
  df_full_results_grouped <- group_by(df_full_results, amount_station_out)

  df_rmse_gesamt_mean <- summarize(df_full_results_grouped,
                                 rmse_wiese_mean = round(mean(
                                   rmse_wiese_station)
                                                         , 2),
                                 rmse_wiese_sd = round(sd(rmse_wiese_station)
                                                       , 2))

  # Different plotting
  # Barplot with errorbars
  png(file = file.path(envrmt$path_prediction, "rmse_wiese_mean_barplot.png"),
      width =600, height = 400, res = 80)

  ggplot(df_rmse_gesamt_mean, aes(x = amount_station_out,
                                  y = rmse_wiese_mean)) +
    geom_bar(stat = "identity", fill = "black", alpha = 0.6) +
    scale_x_continuous(breaks = 1:17) +
    labs(x = "Number of loggers omitted", y = "Mean RMSE (°C)") +
    geom_errorbar(aes(x = amount_station_out,
                    ymin = rmse_wiese_mean-rmse_wiese_sd,
                    ymax = rmse_wiese_mean+rmse_wiese_sd),
                  width = 0.5, size = 0.8)

  dev.off()

  # Boxplots
  df_full_results$amount_station_out <-
    as.factor(df_full_results$amount_station_out)

  png(file = file.path(envrmt$path_prediction, "rmse_wiese_boxplot.png"),
      width =600, height = 400, res = 80)

  ggplot(df_full_results, aes(x = amount_station_out, y = rmse_wiese_station)) +
    geom_boxplot() +
    labs(x = "Number of loggers omitted", y = "RMSE (°C)")

  dev.off()
}
```

Heatmap

```
# For colors
cols <- viridis(100)

if (!file.exists(file.path(envrmt$path_prediction,
                           "heatmap_treetalker_rmse.png"))) {
  # Creating a new variable to not overwrite the "main" result table.
  df_full_results2 <- df_full_results[,-c(2:8)]

  df_full_results2$amount_station_out <- as.factor(
    df_full_results2$amount_station_out)

  # Group by the "amount_station_out"
  df_results2_grouped <- df_full_results2 %>%
    group_by(amount_station_out) %>%
    summarise(across(everything(), mean))

  # Create a "long" dataframe
  df_results_grouped_long <- data.table::melt(df_results2_grouped,
                                              id.vars = c("amount_station_out"))

  png(file = file.path(envrmt$path_prediction, "heatmap_treetalker_rmse.png"),
      width =1124, height = 770, res = 80)

  # Create levelplot
  levelplot(value ~ amount_station_out*variable, data=df_results_grouped_long,
            xlab="Number of loggers omitted",
            ylab = "Loggers",
            col.regions = cols)

  dev.off()
}
```

```
# Plotting barplot of the mean rmse of the TreeTalker loggers
if (!file.exists(file.path(envrmt$path_prediction,
                           "mean_rmse_gesamt_barplot.png"))) {

  # Calculate the mean RMSE for each station
  mean_each_station <- aggregate(df_results_grouped_long[, 3], list(
    df_results_grouped_long$amount_station_out), mean)

  mean_each_station_sd <- aggregate(df_results_grouped_long[, 3], list(
    df_results_grouped_long$amount_station_out), sd)
  mean_each_station$sd <- mean_each_station_sd$x
  # Renaming of one column
  names(mean_each_station)[2] <- "mean_v"

  png(file = file.path(envrmt$path_prediction, "mean_rmse_gesamt_barplot.png"),
      width =600, height = 400, res = 80)
  ggplot(mean_each_station, aes(x=Group.1, y=mean_v)) +
    geom_bar(stat = "identity")+
    xlab("Number of loggers omitted")+
    ylab("Mean RMSE (°C)")+
    geom_errorbar(aes(x = Group.1,
                  ymin = mean_v-sd,
                  ymax = mean_v+sd),
                  width = 0.5, size = 0.8)

  dev.off()
}
```

```
# Mean the average of each number of omitted loggers for the whole study area
if (!file.exists(file.path(envrmt$path_prediction,
                           "full_prediction_study_area.png"))) {
  # Meaning the .tif files
  for (u in 1:17){

    files <- list.files(path=paste0(envrmt$path_prediction,"/",u,"/"),
                        pattern="*.tif$", full.names=F, recursive=FALSE)
    Extract_string <- paste0("mean.tif")
    matches <- unique (grep(paste(Extract_string,collapse="|"),
                            files, value=TRUE))

    # Create raster stack from the selected .tif files
    prediction_stack <- stack(paste0(envrmt$path_prediction,"/",u,"/",
                                     matches,sep=''))

    # Calculate the mean from the raster stack
    prediction_mean <- calc(prediction_stack, fun = mean, na.rm = T)

    if (u < 10){
      writeRaster(prediction_mean,paste0(envrmt$path_prediction,
                                         "/mean_prediction/prediction_mean_0",
                                         u,".tif"))
    } else if (u >= 10){
      writeRaster(prediction_mean,paste0(envrmt$path_prediction,
                                         "/mean_prediction/prediction_mean_",
                                         u,".tif"))
    }
  }

  # Plot the meant .tif files together
  files <- list.files(path=paste0(envrmt$path_prediction,"/mean_prediction/"),
                      pattern="*.tif$", full.names=F, recursive=FALSE)
  prediction_stack <- stack(paste0(envrmt$path_prediction,"/mean_prediction/",
                                   files,sep=''))

  rasterstacknames <- c("1","2","3","4","5","6","7","8","9",
                        "10","11","12","13","14","15","16","17")

  png(file = file.path(envrmt$path_prediction,
                       "full_prediction_study_area.png"),
                        width =1124, height = 770, res = 80)

  raster::spplot(prediction_stack,names.attr= rasterstacknames,
                 col.regions = plasma(16))

  dev.off()
}
```