```
#---------1. SCRIPT INSTRUCTIONS----------
# To minimise user interaction, run sections 4.-5. and 6.-7. at once. Specify parameters in section 3.


#---------2. LOAD NECESSARY PACKAGES----------

library(rgdal)
library(sp)
library(splancs)
library(rgeos)
library(spatial.tools)
library(raster)
library(foreign)
library(RCurl)


#---------3. SET CUSTOM PARAMETERS------------

setwd("/Users/Simon/Studium/MSC/Best Practice in R/DistributionProject/test_project") # set the working directory
species <- "Puma concolor" # species name must be latin
parameters <- c("alt", "tmin", "tmax", "prec", "bio", "tmean") # all possible WorldClim parameters
target_resolution <- 50000 # target grid cell size in m
crs <- "+init=epsg:32612" # adequate equal area projection of target area as PROJ.4 - string


#---------4. LOAD WORKFLOW FUNCTIONS----------

# 4. a) Function for the download of WorldClim parameters

getWorldClim <- function(par, res) {
  parlist <- list()
  for (i in 1:length(par)) {
    getData('worldclim', path=path.temp_data, download = T, var=par[i], res=res)
  }
}


# 4. b) Function that creates link to IUCN profile of intended species, where the species distribution file can be downloaded

```

```
IUCNdata <- function(name) {
  browseURL(paste("http://maps.iucnredlist.org/map.html?id=", ID,
     sep=""))
  message("Make sure to move the species folder into your working
     directory")
}


# 4. c) Function to round numbers to nearest multiple of a specific
   number (needed in creation of template raster) (by Alberto
   Santini: https://gist.github.com/albertosantini/3638434)

mround <- function(number, multiple) {
  # if number and multiple have different sign, returning an error.
  if (sign(number) != sign(multiple)) {
    stop("number and multiple have different sign")
  }
  n = number / multiple
  if (abs(n - trunc(n)) == 0.5) {
    if (sign(n) > 0) {
      n = n + 0.1
    } else {
      n = n - 0.1
    }
  }
  round(n) * multiple
}


#---------5. DOWNLOAD RAW DATA-------------


# 5. a) Set temporary folder to save WorldClim, land cover and
   worldborders shapefile. Deleted after each workflow completion to
    avoid possible conflicts


if (file.exists(paste(getwd(), "temp_data", sep="/"))) {
  path.temp_data <- paste(getwd(), "temp_data", sep = "/")
  unlink(paste(getwd(), "temp_data", sep = "/"), recursive = T)
  dir.create(path.temp_data)
  } else {
  path.temp_data <- paste(getwd(), "temp_data", sep = "/")
  dir.create(path.temp_data)
}
```

```r


# 5. b) Prompt link to species download page at IUCN.

# Extract species ID from file stored in github repository.
x <- getURL("https://raw.githubusercontent.com/achumani/
   Verbreitungsdaten-in-R/master/IUCN_Species_ID.csv")
SpeciesID <- read.csv(text = x, header = T)
ID <- as.character(SpeciesID$id_no[which(SpeciesID$binomial ==
   species)])

# Promt download link
if (file.exists(paste(getwd(), paste("species_", ID, sep = ""), sep
   = "/"))) {
  message("Download not necessary. The file already exists in your
     working directory.")
} else {
  IUCNdata(species)
  message("Please download the species distribution file from the
     IUCN-website that just popped up and move the downloaded file
     as is into your working directory. If you returned the entire
     download files code chunk, your specified files should be
     downloaded to your working directory in the meantime")
}


# 5. c) Download specified WorldClim data into temporary folder

par <- parameters
if(target_resolution < 20000) {
  res <- 5
} else {
  res <- 10
}
getWorldClim(par, res)


# 5. d) Download global land cover data

download.file("http://www.fao.org/geonetwork/srv/en/resources.get?id
   =47948&fname=GlcShare_v10_Dominant.zip&access=private", paste(
   path.temp_data, "GLCdom.zip", sep = "/"))
unzip(paste(path.temp_data, "GLCdom.zip", sep = "/"), exdir = paste(
   path.temp_data, "GLCdom", sep = "/"))
file.remove(paste(path.temp_data, "GLCdom.zip", sep = "/"))
```

```
113
114
115 # 5. e) Download global borders
116
117 download.file("http://thematicmapping.org/downloads/TM_WORLD_BORDERS
    -0.3.zip", paste(path.temp_data, "WorldBorders.zip", sep = "/"))
118 unzip(paste(path.temp_data, "WorldBorders.zip", sep = "/"), exdir =
    paste(path.temp_data, "WorldBorders", sep = "/"), overwrite = T)
119 file.remove(paste(path.temp_data, "WorldBorders.zip", sep = "/"))
120
121
122 #---------6. RASTERISATION-----------------
123
124 # 6. a) Select Area of Interest (needs to be within equal area
    projection specified above)
125
126 # Read species distribution and worldborders shapefiles and plot
    them into 0-device
127 worldborders <- readOGR(dsn = paste(path.temp_data, "WorldBorders",
    sep = "/"), layer = "TM_WORLD_BORDERS-0.3")
128 species.shp <- readOGR(dsn = paste(paste(getwd(), "species_", sep =
    "/"), ID, sep = ""), layer = paste("species_", ID, sep = ""))
129
130 # plot for AOI selection
131 plot(species.shp, col = "brown")
132 plot(worldborders, add = T)
133
134 # clip a polygon from distribution map, convert to spatial polygons
    object
135 aoi <- getpoly(quiet = F)
136 aoi_sp <- Polygon(aoi)
137 aoi_sps <- Polygons(list(aoi_sp),ID = ID)
138 aoi_spls <- SpatialPolygons(list(aoi_sps), proj4string = CRS(species
    .shp@proj4string@projargs))
139
140
141 # 6. b) Build template raster covering the Area of Interest
142
143 # get the extent of the clipped polygon and convert to target
    projection
144 aoi_ext <- gEnvelope(aoi_spls)
145 aoi_ext_utm <- spTransform(aoi_ext, crs(crs))
146 ext <- extent(aoi_ext_utm)
147
148 # force target grid to be a multiple of desired grid size in order
```

```
   to get integer values for dimension
dimx <- (mround(length(ext@xmin:ext@xmax), target_resolution))/
   target_resolution
dimy <- (mround(length(ext@ymin:ext@ymax), target_resolution))/
   target_resolution

# build a template raster for the spatial_sync_raster function
aoi_rr_utm <- raster(ext, ncols = dimx, nrows = dimy, crs = crs)


# 6. c) fitting species distribution data into template raster

# crop to AOI extent
species_aoi_shp <- crop(species.shp, aoi_ext)

# transfrom to target crs
species_aoi_shp_utm <- spTransform(species_aoi_shp, crs(crs))

# rasterise species shapefile, extracting proportion of each raster
   cell covered by species shapefile
species_aoi_raster <- rasterize(species_aoi_shp_utm, aoi_rr_utm,
   getCover = T)
species_aoi_raster@data@values <- species_aoi_raster@data@values/100
species_aoi_raster@data@names <- species


# 6. d) fitting WorldClim data into template raster

# find WorldClim rasters in their download folder and save their
   target path and names in two lists at corresponding list
   positions.
rasterlist <- list.raster.files(paste(paste(path.temp_data,"wc", sep
    = "/"),res, sep = ""), pattern = "bil$")
names <- list.files(paste(paste(path.temp_data, "wc",sep = "/"), res
   , sep = ""), pattern = "bil$")

# load, crop, reproject and resample WorldClim raster to fit
   template raster
wc_processed <- list()
for (i in 1:length(rasterlist$raster_files)) {
  r <- raster(rasterlist$raster_files[[i]], sep = "") # loading
     longlat rasters
  rc <- crop(r, aoi_ext)
  rc_utm <- projectRaster(rc, crs = crs)
  wc_processed[[i]] <- resample(rc_utm, aoi_rr_utm, method = "
```

```
      bilinear")
  name <- paste("processed_", names[[i]], sep = "")
}

wc_processed <- stack(wc_processed)


# 6. e) fitting land cover data into template raster

# load Tiff, assign CRS, transform it to formal raster class and
   crop it
message("The following 5 steps can take up to 10 minutes to complete
   . Time for a cuppa!")
glc <- readGDAL(paste(path.temp_data, "GLCdom/glc_shv10_DOM.tif",
   sep = "/"))
crs(glc) <- crs(aoi_ext)
glc <- raster(glc)
glc_c <- crop(glc, aoi_ext)
glc_c_utm <- projectRaster(glc_c, crs = crs, method = "ngb")

#delete the massive glc raster
remove(glc)
gc()

# use utm template raster to create spatial polygons object as mask
   for value extraction by cell
rastercells <- rasterToPolygons(aoi_rr_utm)
lcraster <- extract(glc_c_utm, rastercells)

# set NA to 0 (to equalise length of land cover pixels per target
   cell)
for (i in 1:length(lcraster)) {
  lcraster[[i]][is.na(lcraster[[i]])] <- 0
}

# calculation of proportion of each class in each target cell
lcsharelist <- list()
for (i in 1:length(lcraster)) {
  lcshares <- numeric()
  for (j in 1:11) {
    lcshares[j] <- cbind(length(lcraster[[i]][lcraster[[i]]==j])/
       length(lcraster[[i]]))
  }
  lcsharelist[[i]] <- lcshares
}
```

```r

# creation of rasterstack from land cover shares
landcovernames <- c("ARTIFICIAL", "CROP", "GRASS", "TREE", "SHRUB",
   "HERBS", "MANGROVES", "SPARSE VEGETATION", "BARE", "SNOW", "WATER
   ")
lclayers <- stack()
for (i in 1: length(landcovernames)) {
  layer <- setValues(aoi_rr_utm, do.call(rbind, lcsharelist)[,i])
  names(layer) <- landcovernames[i]
  lclayers <- addLayer(lclayers, layer)
}


#----------7. SET UP FINAL DATASHEET--------

# combine species distribution raster, WorldCLim stack and land
   cover stack into one stack object
alldata <- addLayer(species_aoi_raster, lclayers, wc_processed)

# get values and save as dataframe
alldata_df <- as.data.frame(rasterToPoints(alldata))

# assign remaining column names, species-ID-column, add unique
   identifier column (CELLCODE)
colnames(alldata_df)[1:2] <- c("EOFORIGIN", "NOFORIGIN")
final <- cbind(ID = ID, CELLCODE = paste0(target_resolution/1000,"KM
   ","E", round(alldata_df$EOFORIGIN/1000), "N", round(alldata_df$
   NOFORIGIN/1000)), alldata_df)

# export sheet as csv with conditional name into working directory
write.csv(final, paste0("species_", ID,"_", target_resolution/1000,
   "KM_data.csv"))

# create geotiff of all rasterized data
writeRaster(alldata, paste0("species_", ID,"_", target_resolution/
   1000, "KM_data"), format = "GTiff", overwrite = TRUE)

# delete temp folder
unlink(paste(getwd(), "temp_data", sep = "/"), recursive = T)
```