Preparing spatial data for species distribution modelling in R

Simon Kapitza* Adriano Fernandez Kaulich†

November 28, 2014

Abstract

This report serves as documentation for an Rscript produced during the Masters module "Best practice for data analysis and R: writing and coding geographic and statistical analyses for theses and reports". The Rscript loads and processes a user-specified species distribution shapefile and a multitude of user-specified environmentally relevant parameters in raster format, combining both file formats into one data frame representing equal area raster cells. The resulting data frame can be used for macro-ecological analysis, such as species distribution modelling. The aim is to automise a relatively easy GIS task, so that users, who are more interested in statistics and less in GIS, are able to prepare their data set with minimal user input and without relying on conventional GIS software. Excerpts from the Rscript are interlaced in this report. The entire Rscript is appended.

Contents

*MtrklNr.: 3710110
†MtrklNr.: 2920170

1 Introduction

The aim of this project is to facilitate the use of R as GIS-Software. GIS-Software is often costly. Hence, the free statistical software R becomes a worthwhile alternative. In order to help students and researchers explore R as an alternative to GIS, many common GIS functions need to be implemented in a comfortable way in R.
In this document we explain how our Rscript can help researchers prepare their data for species distribution modeling (SDM). Data used in SDM is often presented as a spatial grid (raster-data), with every grid cell containing data about the presence of the species and environmental parameters like altitude and land cover or precipitation and temperature records. Our Rscript brings data of this type into one grid of user-specified properties.

2 Rscript documentation

The first step is to acquire all the necessary data for the analysis. This requires the user to input some basic parameters like the species name (in Latin), the desired cell size of the grid (in meters) and the EPSG-code for the coordinate reference system best suited to the extent of the study site (equal area projection).

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

setwd("/Users/foo/bar")     # set the working directory
species <- c("Dama dama")   # species name must be the Latin name
parameters <- c("alt", "tmin", "tmax", "tmean", "prec", "bio")
                            # all possible WordClim.org parameters
target_resolution <- 50000  # example with target grid cell size of 50km
crs <- "+init=epsg:3035"    # example with Europe's equal area projection

#---------------------------------
```

Spatial data for almost every non-domesticated mammalian, amphibian, reptilian and marine fish species is accessible at `www.iucnredlist.org`. A function was written that redirects the user to the download page of the requested species at the website of the IUCN. Unfortunately a user login is required, otherwise it would have been possible to directly download the spatial data and load it into R. The user is required to move the downloaded species distribution folder into the set working directory. It is possible to manually download all species distribution shapefiles in bulk (over 2 GB of data in shapefiles) from the website of the IUCN, which could be added as an additional feature to the present workflow. Raster data for climate records is available at `www.worldclim.org` without a user login, allowing us to directly download and load the data into a temporary directory for use in the workflow. The same applies for land-cover data from `www.fao.org`. Both data sets will be deleted eventually.
(Start the workflow: Load all libraries in section 2. of the Rscript and specify your species, WorldClim-parameters, your target resolution and the equal area coordinate reference system of your intended study site in section 3. In order to download all required files and be prompted with a download link for your species' distribution shapefile, execute sections 4. - 5. in the Rscript)

Once all the data is available in R, the species is plotted onto a 2D plane (WGS84 Lat/Lon; EPSG:4326) together with the borders of countries in the distribution area to facilitate the selection of the area of interest. The world borders shapefile was downloaded from `www.thematicmapping.org`. The user is then asked to select the study area (*AoI*) on the plotted maps.The extent of a rectangular box surrounding the *AoI* is then extracted using the function `gEnvelope()`. The box is then converted to a Formal Class SpatialPolygon and reprojected into the user-specified CRS. Since the user-defined CRS is metric, the `extent()`-function allows us to extract the dimensions of the *AoI* and calculate the desired raster grid (dividing the *aoi's* extent by the desired cell size, e.g. 50 km). The grid size has to be a multiple of the desired cell size. If this condition is not met, the original extent will be rounded to the next biggest multiple of the specified cell size (e.g. if the original extent between xmin

and xmax from `gEnvelope()` is 887 km, the final grid will be extended to 900km, in order to allow 18 cells of 50 km on the x-axis). In order to round to multiples, the `mround()` function by Alberto Santini: `https://gist.github.com/albertosantini/3638434` was applied.

```
#---------6. RASTERISATION-----------------

# 6. a) Select Area of Interest (needs to be within equal
        area projection specified above)

# Read species distribution and worldborders shapefiles and
  plot them into 0-device
worldborders <- readOGR(dsn = paste(path.temp_data, "WorldBorders",
                sep = "/"), layer = "TM_WORLD_BORDERS-0.3")
species.shp <- readOGR(dsn = paste(paste(getwd(), "species_", sep = "/"),
                ID, sep = ""), layer = paste("species_", ID, sep = ""))

# plot for AOI selection
plot(species.shp, col = "brown")
plot(worldborders, add = T)

# clip a polygon from distribution map, convert to spatial polygons object
aoi <- getpoly(quiet = F)
aoi_sp <- Polygon(aoi)
aoi_sps <- Polygons(list(aoi_sp),ID = ID)
aoi_spls <- SpatialPolygons(list(aoi_sps),
            proj4string = CRS(species.shp@proj4string@projargs))


# 6. b) Build template raster covering the Area of Interest

# get the extent of the clipped polygon and convert to target projection
aoi_ext <- gEnvelope(aoi_spls)
aoi_ext_utm <- spTransform(aoi_ext, crs(crs))
ext <- extent(aoi_ext_utm)

# 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)
```

The species distribution shapefile should be cropped to the extent of the *AoI* before it is processed into a raster. The unprojected polygon boxing the selected area of interest is used for this. The resulting cropped species shapefile is then projected into the target CRS. Rasterising this cropped and reprojected spatial shapefile takes into account the area the species distribution polygon covers in each cell. Cells located completely within the extent of the polygon will be assigned the value 1 for full coverage by the polygon. Along the edges of the polygon, the rasterised cells' values range from from 0 to 1, depending on how much of these cells is actually covered by the species shapefile. This is important to adequately weight their influence in further analysis (e.g. in a linear regression).
To crop your area of interest and calculate a template raster that is then filled cell-by-cell with your specified parameters, execute the entire section 6. of the Rscript. Be aware that processing land cover data may take a while (about 10 minutes).

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

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

# transform 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
```

Processing the rasterised environmental data starts with indexing the location of every rasterfile inside the folders they were downloaded to, using the function `list.raster.files()`. This list of file paths is used by a `for`-loop that loads every downloaded WorldClim raster and crops, reprojects and resamples them in accordance with the properties of the template raster. The files from `www.worldclim.org` are detected by `list.raster.files()` and loaded in alphabetical order. This would be fine except that the WorldClim monthly data are numbered for each month (e.g. tmin1, tmin2, tmin3 ... tmin10, tmin11, tmin12). The rasters for October, November and December are thus loaded before the rasters for February in the list of file-paths returned by the `list.raster.files`-function. As a workaround, an empty list with only the file names was created, using the `list.files`-function. Since the raster names are called into the loop in the same order as the rasters, they can be assigned to the rasters by using their indices. This way, the rasters get their original names while being on an index position that does not correspond to the ID of their month.
Processing itself includes the cropping, projecting and resampling of every rasterfile in `rasterlist` using the same template raster created for rasterising the species shapefile. The `resampling`-function offers two ways of estimating the new cell values: nearest-neighbour and bilinear resampling. Bilinear resampling yields better results but takes notably longer than the nearest neighbour option. The processed rasters were stored as raster stack.

```
# 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)
```

Global land-cover data are loaded using the `readGDAL()` function. This function did not seem to create a valid spatial object although its projection is in EPSG:4326. It was thus necessary to assign the EPSG:4326 projection to the global raster and then load it as Formal Class Raster using the `raster()` function. It was then cropped to *AoI*-size and reprojected in the specified target CRS. To extract the land-cover class values for the calculation of the share each land-cover class has in each target grid cell, the template raster was converted to a Spatial Polygon, using the `rasterToPolygons()` function. Values were extracted from the cropped land-cover raster using the `extract()` function and then saved as list. NA values were set to 0. The class proportions were calculated and saved in a list. The values were then

written into copies of the template raster, in order to create a raster stack with one layer for each land-cover class.

```
# 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 from the Rworkspace
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
}

# 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)
}
```

As a final step, a *.csv-file containing all data processed for the *AoI* as well as a corresponding multiband GeoTiff file for control of the correct execution of the Rscript are written into the working directory.

```
#----------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)
```

3 Short-comings

Unfortunately, it was not possible to automise the download of species shapefiles from the IUCN website. The function could be extended to get the questionnaire and user sign-up at IUCN into the R console and facilitate the download of according files straight into the working directory. An additional useful functionality of this workflow could be to automatically select the best CRS for any specified study area. However, this would require a lot of programming and it may be more appropriate to leave the CRS-selection to the user, as they would be able to decide on the basis of the conducted study. Areas of interest with extraordinary large extends along the x-axis will make it difficult to define an adequate coordinate reference system, thus making our script unable to run. The user must be aware that equal area projections are limited in their ability to represent truly equal areas when moving away from their center of projection. In case the species ID list gets modified by the IUCN, the github-hosted *.txt-file that relates the Latin name to the ID necessary for redirecting the user to the download page would also needed to be manually updated.

We did not provide nor did we research what environmental parameters are relevant or available to analyse marine life. But considering that the IUCN provides shapefiles for marine fish species, species distribution modeling data could be prepared for analysis by our Rscript as well.