Modelling species distribution

Ian Ondo

2023-05-22

Contents

Package notes:
We need to install the following set of R packages to successfully run the codes in this vignette:

- **geodata** (Hijmans et al. 2023)
- **dismo** (Hijmans et al. 2021)
- **usdm** (Naimi et al. 2014)
- **ade4** (Dray, Dufour, and Chessel 2007)
- **ggplot2** (Wickham 2016)
- **kableExtra** (Zhu 2023)
- **dplyr** (Wickham et al. 2023)

Introduction

In this vignette, we will focus on modelling the distribution of *Abies spectabilis* (hereafter *A. spectabilis*) with the **UsefulPlants** package. We will walk through the modelling process and introduce key functions and R codes used to assist each individual steps.

I. Data preparation

Our occurrence dataset contains 40 rows of occurrence locations of *A. spectabilis* curated beforehand (see vignette **Data gathering and pre-processing**. The first few rows look like this:

Table 1: Occurrence dataset of *A. spectabilis*

species	decimalLongitude	decimalLatitude	year	countryCode	basisOfRecord
Abies spectabilis	88.45	27.55	**	IND	PRESERVED_SPECIMEN
Abies spectabilis	88.85	27.47	**	CHN	PRESERVED_SPECIMEN
Abies spectabilis	85.29	28.85	1975	CHN	PRESERVED_SPECIMEN
Abies spectabilis	88.90	27.48	1975	CHN	PRESERVED_SPECIMEN
Abies spectabilis	87.12	28.65	1959	CHN	PRESERVED_SPECIMEN
Abies spectabilis	87.76	28.36	1975	CHN	PRESERVED_SPECIMEN

Note:
Only relevant columns are shown

For simplicity, we are going to use the R package **geodata** to obtain a set of environmental layers. Various climate, elevation and soil-related raster datasets can be directly downloaded from R with this package.

```
library(geodata)

# setup a (temporary) directory to store your environmental raster layers
my_env_tmp_dir = tempdir()

# spatial resolution
my_res = 10 # 10 minutes degree resolution

# download global dataset of WorldClim bioclimatic variables
wc = geodata::worldclim_global(var="bio",
                               res=my_res,
                               path = my_env_tmp_dir)

# download global dataset of SRTM elevation model
alt = geodata::elevation_global(res=my_res,
                                path = my_env_tmp_dir)

# download the human footprint index
hfp = geodata::footprint(year = 2009,
                  path = my_env_tmp_dir)

# uncomment below to download soil-related variable, but be mindful,
# global downloads at 30 seconds resolution take a while !
#
# download global dataset of soil organic carbon and pH at 30-60cm depth
# soc = geodata::soil_world(var="soc",depth=60, path=my_env_tmp_dir)
# pH = geodata::soil_world(var="phh2o", depth=60, path=my_env_tmp_dir)
```

Defining the training area

We need to delineate the geographic area in which we will train our model. This training area should ideally represent the environmental conditions available or accessible to the species. In **UsefulPlants**, it extends to the boundaries of the biomes or ecoregions where species' occurrence locations were found. We are going to use the function `make_geographic_domain` which relies on *Terrestrial Ecoregions Of the World* (TEOW) and some user-defined settings to create the training area from our occurrence data points.

```
library(UsefulPlants)

# create your training area
my_training_area <- make_geographic_domain(

  # path to the occurrence records file
  loc_dat=params$occ_file,

  # optional vector of longitude/latitude
  coordHeaders = c("decimalLongitude","decimalLatitude"),

  # build alpha-hull from the set of points
  do.alpha_hull = TRUE,

  # do not dissolve borders between polygons (i.e. biomes or ecoregions)
  dissolve = FALSE,

  # build the alpha-hull with at least 95% of the points
  fraction = 0.95,

  # get biomes whose intersection area with the alpha-hull >= 10% (of their area),
  # otherwise get ecoregions
  min_area_cover = 0.1,

  # get biomes with >= 10 points inside, otherwise get ecoregions
  min_occ_number = 10,

  # optional terrestrial land vector maps
  land_file = RGeodata::terrestrial_lands,

  # run quietly. set to TRUE to display stepwise detailed information
  verbose=FALSE
)
```

Internally, `make_geographic_domain`:

- Built an alpha-hull around occurrence points (include at least 95% of points by default)
- Detected biomes in the TEOW dataset that intersect with the alpha-hull
- Made sure that these biomes include >= N points (default is N=10) and that the intersection area with the alpha-hull >= X% (default is X=10% of the biome area)
- Made sure to include ecoregions of points excluded by the alpha-hull, or geographic outliers if they exist.

Let's have a look at the geographic region selected to be sure that it makes sense by overlaying our occurrence records on the map.

```
  # plot the region selected and overlay your occurrence data points
  mplt <- my_training_area %>% ggplot2::ggplot() +
    ggplot2::geom_sf(ggplot2::aes(fill = REGION_NAME)) +
    ggplot2::geom_point(data = occ_data,
                        ggplot2::aes(x = decimalLongitude, y = decimalLatitude),
                        size = 2,
                        shape = 4,
                        fill = "black") +
    ggplot2::labs(fill = "Biomes/Ecoregions",) +
    ggplot2::theme_bw() +
    ggplot2::theme(
    text=ggplot2::element_text(family="serif"),
    legend.title = ggplot2::element_text(size=7, face = "bold"),
    legend.key.size = unit(0.4,'cm'),
    legend.text = ggplot2::element_text(size=5),
    axis.text=ggplot2::element_text(colour="black", size=6),
    axis.title=ggplot2::element_text(colour="black", size=8,face="bold")
    ) +
    ggplot2::xlab("Longitude") + ggplot2::ylab("Latitude") +
    ggplot2::scale_fill_hue(l=40)

mplt
```

> For this vignette, we intentionally displayed the biomes and ecoregions composing the training area of our species by setting **dissolve=FALSE**, however, for the rest of the analysis make sure to set **dissolve=TRUE** when creating your training area to merge all biomes and ecoregions into one geographic region.

The region selected seems okay, but if not satisfied with it, try to change some parameters e.g. `fraction`, `min_area_cover` or `min_occ_number` until you get the desired training area.

Selecting environmental predictors

First of all, the environmental rasters were downloaded at different spatial grain, extent or projection, so to ensure that all datasets spatially match, we need to:

- **clip** all layers to the training area previously defined to speed up the computations.
- **resample** all layers using the geographic information of a raster taken as reference for the study area.

We use the function `maskCover` to clip the rasters. It does a better job to approximate the irregular shape of the training area by using the intersection of the entire grid cell with the training area polygon rather than the grid cell's centroid like other functions.

```
# clip climate data
wc_clipped = maskCover(wc, my_training_area)

# clip elevation data
```

```r
alt_clipped = maskCover(alt, my_training_area)

# clip human footprint
hfp_clipped = maskCover(hfp, my_training_area)

# resample elevation and human footprint layers using a bioclimatic layer as a reference raster
alt_resampled <- raster::resample(
  alt_clipped,
  wc_clipped[[1]],
  method = "ngb"  # keep discrete values
)

hfp_resampled <- raster::resample(
  hfp_clipped,
  wc_clipped[[1]]
)
```

Now, let's compute three more topography-related variables, the *land slope (S)*, the *Topographic Ruggedness Index (TRI)* and the *terrain roughness (R)*

```r
# compute S, TRI and R
topo <- raster::terrain(alt_resampled, opt=c("slope","TRI","roughness"))

# add to climate and human footprint layers
env_data <- raster::stack(wc_clipped, hfp_resampled, topo)
```

Variable selection is not a linear process, one needs to balance statistical evidence with ecological meaning, and this usually entails going back and forth between different steps until finding a convincing set of variables.

> A good practice for selecting environmental predictors in plant studies, starts by gathering a large set of factors known to influence the physiological response of plants to environmental conditions such as factors related to nutrition (availability of water and nutrients) and energy (availability of light). Then, progressively removing redundant information (i.e. factors highly correlated) while keeping:
>
> - direct factors over proxies
> - factors whose effects are easier to interpret
> - annual trend and seasonal effects over monthly extremes (for large scale studies)
> - limiting factors and stress effects over maxima.

There are many ways to do a variable selection, but here is one example, of what a variable selection may look like.

PCA

We can start our selection by performing a Principal component Analysis (PCA) to identify variables that most drive the environmental conditions of our study area (variables contributing the most to the first axis of the PCA). This will give a first broad idea of which variables best describe our training area.

```r
# keep the first two principal components (PC)
ndim = 2

# perform PCA
pca <- ade4::dudi.pca(df=env_data[],
                      center= TRUE, # (optional) remove the mean
                      scale=TRUE,
                      scannf=F, # do not plot the screeplot
                      nf=ndim) # nf is the number of dimensions

# Compute inertia explained by each of dimension
inertia = pca$eig/sum(pca$eig)*100
m <- data.frame(Comp=paste0('Dim',1:length(inertia)), inertia=inertia)

# plot eigenvalues
bplt <- ggplot2::ggplot(data=m, ggplot2::aes(x=Comp, y=inertia)) +
  ggplot2::geom_bar(stat ="identity", ggplot2::aes(fill=as.factor(inertia)),
                    show.legend = FALSE) +
  ggplot2::geom_text(ggplot2::aes(label=paste0(round(inertia,2),"%"),
                                  y=inertia+0.1, fontface='bold'),
                                  vjust=0,
                                  color="black",
                                  position = ggplot2::position_dodge(1), size=4.5) +
  ggplot2::theme_classic() +
  ggplot2::labs(title="Barplot of Eigenvalues",
                subtitle=paste0("Using the first ",ndim," dimensions"),
                caption=paste0("The two first coordinates (dimensions) explain ",
                               round(sum(inertia[1:2]),2),"% of the variability")) +
  ggplot2::theme(
    text=ggplot2::element_text(family="serif"),
    plot.title = ggplot2::element_text(face="bold", size=14, hjust = 0),
    axis.text=ggplot2::element_text(colour="black", size=12),
    axis.title=ggplot2::element_text(colour="black", size=16,face="bold")
  ) +
  ggplot2::xlab("Dimensions") + ggplot2::ylab("Percentage of explained variances") +
  ggplot2::scale_fill_brewer(palette='Blues') +
  ggplot2::scale_x_discrete(limits=paste0('Dim',1:ndim))

bplt
```

```r
# Evaluation of the absolute contribution of a variable to an axis
cont=ade4::inertia.dudi(pca,col.inertia=TRUE)$col.abs

# Sort in decreasing order
ctr=cont[order(cont[,1],decreasing=FALSE),]

## Representation of the contribution of each variables to an axis x
bplt_axis_1 <- ggplot2::ggplot(data=ctr, ggplot2::aes(x=Comp, y=inertia)) +
  ggplot2::geom_bar(stat ="identity", ggplot2::aes(fill=as.factor(inertia)), show.legend = FALSE) +
  ggplot2::geom_text(ggplot2::aes(label=paste0(round(inertia,2),"%"),
                                  y=inertia+0.1, fontface='bold'),
                                  vjust=0,
                                  color="black",
```

```
                                  position = ggplot2::position_dodge(1), size=4.5) +
  ggplot2::theme_classic() +
  ggplot2::labs(title="Barplot of variables' contribution",
                subtitle=paste0("Using the first ",ndim," dimensions"),
                caption=paste0("The two first coordinates (dimensions) explain ",
                               round(sum(inertia[1:2]),2),"% of the variability")) +
  ggplot2::theme(
    text=ggplot2::element_text(family="serif"),
    plot.title = ggplot2::element_text(face="bold", size=14, hjust = 0),
    axis.text=ggplot2::element_text(colour="black", size=12),
    axis.title=ggplot2::element_text(colour="black", size=16,face="bold")
  ) +
  ggplot2::xlab("Dimensions") + ggplot2::ylab("% of contribution") +
  ggplot2::scale_fill_brewer(palette='Blues') +
  ggplot2::scale_x_discrete(limits=paste0('Dim',1:ndim))

bplt
```

Collinearity

Then, we can decide to remove predictors highly correlated to others, by looking at the Variance Inflation Factor (VIF) of each predictor. In the literature, a VIF $>$ 10 (or 5) usually indicate a collinearity issue. We use the package **usdm** which has two functions called `vifcor` and `vifstep` that implement different strategies to deal with collinearity among variables.

```
# detect collineary issues iteratively using a stepwise method
vif_step_method <- usdm::vifstep(env_data,
             th = 10, # VIF threshold
             maxobservations=6000) # maximum number of grid cells to sample

# detect collinearity issues using 2-by-2 correlations
vif_cor_method <- usdm::vifcor(env_data,
             th = 0.7, # correlation threshold
             maxobservations=6000)

# select a method and exclude variables from the set of candidate predictors
env_data_selected <- usdm::exclude(env_data, vif=vif_cor_method)
```

> We removed collinearity among our environmental predictors by using the function **`vifcor`** which uses both VIF values and a correlation threshold. The function detects pairs of highly correlated variables (i.e. with correlation $>$ 0.7 for example), then excludes the variable of the pair with the highest VIF value.

II. Model training & evaluation

We are going to train our model with the set of environmental predictors previously selected and the maximum entropy algorithm MaxEnt (Phillips, Anderson, and Schapire 2006).

```
# if you do not have it yet, download the last version of Maxent (3.4.1)
# It will be copied into the dismo/java folder
success <- rmaxent::get_maxent(quiet=TRUE)
success # if < 0, the download/copy failed.
```

In **UsefulPlants**, MaxEnt is tuned and evaluated using the *masked geographically structured approach* (Radosavljevic and Anderson 2014), which is a variant of the *k-fold* cross-validation that provides a better ability to detect over-fitting.

Data partitioning

We use the function `make_geographic_block` to spatially segregate occurrence records into k=3 geographical bins with approximately the same number of points.

```
# (optional) rasterise the training area
raster_template  <- tryCatch(
  raster::raster(my_training_area, res = grid_res),
  # if any error occur, convert training area to 'sp' object
  error=function(err){
    raster::raster(as(my_training_area,"Spatial"), res = grid_res)
    }
  )
bg <- fasterize::fasterize(sf::st_cast(domain,"MULTIPOLYGON"),
                           raster_template) - 1 # -1 to ensure background is a distinct block

# read occurrence data
occ_data = read.csv(params$occ_file)

# define the number of geographic blocks to create
K = 3

# Partition both occurrence records and training area into K geographic blocks
my_geo_blocks = make_geographic_block(occ_data, # occurrence records data
                      k = K,                    # number of geographic blocks
                      bg = bg,                  # optional (i.e. can be left out)
                      grid_res = 0.1666667,     # resolution in decimal lat/lon
                      sf=TRUE,                  # return a sf object
                      verbose=FALSE)            # run quietly
```

Internally, `make_geographic_block`:

- Used an unsupervised classification algorithm to cluster occurrence points into equal group size given their geographic coordinates/position
- Built convex-hulls around each cluster
- Made sure clusters do not overlap

Model tuning

MaxEnt will be trained iteratively using *k-1(=2)* bins and tested in the last bin. At each iteration, we will compute a range of evaluation metrics among: (i) the corrected Akaike Information criterion (AIC_c), (ii) the tenth percentile of the training omission rate (OR_{10}), the (iii) the Area Under the Curve of the receiver operating characteristic (AUC), and (iv) the maximum of the True Skill Statistics (TSS).

We are going to repeat this procedure for a range of β regularization coefficients (hereafter called β multipliers) and select the model with the β multiplier that obtain the best overall performance given the evaluation metrics selected.

```
# create a vector of beta multipliers to explore
beta_mult = c(1,6,10)

# specify the evaluation metrics to compute
eval_metrics=c("auc",           # AUC
               "omission_rate", # OR10
               "tss",           # TSS
               "ic")            # AICc

# specify a folder where the model outputs will be written
maxent_output_dir = dirname(raster::rasterTmpFile())

# create a list with maxent settings
maxent_settings <- list(
  path_to_maxent            = system.file('java/maxent.jar', package='dismo'),
  visible                   = FALSE,  # hide MaxEnt GUI
  writemess                 = FALSE,  # do not write MESS raster
  writebackgroundpredictions= TRUE,   # write background predictions
  maximumbackground         = 50000,  # sample up to 50,000 background points
  betamultiplier            = beta_mult,
  eval_metrics              = eval_metrics,
  prefixes                  = FALSE,
  threshold                 = FALSE,  # do not use threshold feature class
  hinge                     = TRUE,   # use threshold feature class
  outputformat              = 'raw',  # keep raw outputs (no transformation)
  outputgrids               = FALSE   # do not write ascii grids
)

# optionally compute a sampling bias prior layer to account for unevenly sampled
# locations within the study area

# get species coordinates (coordinates from higher taxonomic ranks or
# other related species can also be included)
coords <- occ_data %>%
  dplyr::select(c(decimalLongitude, decimalLatitude))

# count number of points per grid cell
rasterized_occ <- raster::rasterize(coords,
                                    bg,         # rasterised training area
                                    update=TRUE,
                                    fun='count')

# detect grid cells with presence locations
presences <- which(!is.na(raster::values(rasterized_occ)) &
```

```
                      raster::values(rasterized_occ) > 0L)
pres_locs <- raster::coordinates(rasterized_occ)[presences, ]

# compute density
dens <- MASS::kde2d(pres_locs[,1], pres_locs[,2],
                    n = c(nrow(rasterized_occ), ncol(rasterized_occ)),
                    h=2) # bandwidth
dens_rast <- raster::raster(dens)

# make sure the raster matches the training area raster
biasrast <- raster::resample(dens_ras, bg, method="ngb")

# save to temporary folder
sampbias_fn <- file.path(maxent_output_dir,"sampbiasrast.tif")
bias_rast <- maskCover(biasrast, my_training_area, filename=sampbias_fn)
# rm(bias_rast) if not needed anymore

# add the path to the sampling bias file to the list of settings for maxent
maxent_settings$biasfile <- sampbias_fn

# perform the block cross-validation
model_output <- block_cv_maxent(
                            # our occurrence dataset
                            loc_dat = occ_data,

                            # raster stack of environmental predictors
                            env_dat = env_data_selected,

                            # number of blocks
                            k=K,

                            # optional coordinates headers
                            coordHeaders=c("decimalLongitude",
                                           "decimalLatitude"),

                            # our rasterised training area
                            bg_masks = my_geo_blocks,

                            # output directory
                            outputdir = maxent_output_dir,

                            # (optional) species name
                            species_name = "Abies_spectabilis",

                            # list of settings for Maxent
                            maxent_settings=maxent_settings,

                            # name of the parameter that vary in maxent_settings
                            # list
                            varying_parameter_name="betamultiplier",

                            # disable parallel computing
                            do.parallel=FALSE)
```

The function `block_cv_maxent` returns a data.frame (`model_output`) that reports the score of each evaluation metric within each bin tested. Let's have a look at this:

Now, we can use the function `get_best_maxent_model` to select the best overall model.

```
best_model <- get_best_maxent(model_output, eval_metrics=eval_metrics)
```

Internally, `get_best_maxent_model`:

- Average evaluation metrics of each model across geographic bins
- Sort the models from the lowest information criteria →lowest OR_{10} →highest AUC →highest TSS.

III. Model projection

Now that we determined the best β multiplier for our model, we can train it using the full dataset (i.e. without withholding occurrence records) and project it across space.

References

Dray, Stéphane, Anne-Béatrice Dufour, and Daniel Chessel. 2007. "The ade4 Package – II: Two-Table and K-Table Methods." *R News* 7 (2): 47–52. https://cran.r-project.org/doc/Rnews/.

Hijmans, Robert J., Márcia Barbosa, Aniruddha Ghosh, and Alex Mandel. 2023. *Geodata: Download Geographic Data*. https://CRAN.R-project.org/package=geodata.

Hijmans, Robert J., Steven Phillips, John Leathwick, and Jane Elith. 2021. *Dismo: Species Distribution Modeling*. https://CRAN.R-project.org/package=dismo.

Naimi, Babak, Nicholas a.s. Hamm, Thomas A. Groen, Andrew K. Skidmore, and Albertus G. Toxopeus. 2014. "Where Is Positional Uncertainty a Problem for Species Distribution Modelling." *Ecography* 37: 191–203. https://doi.org/10.1111/j.1600-0587.2013.00205.x.

Phillips, Steven J, Robert P Anderson, and Robert E Schapire. 2006. "Maximum Entropy Modeling of Species Geographic Distributions." *Ecological Modelling* 190 (3-4): 231–59.

Radosavljevic, Aleksandar, and Robert P Anderson. 2014. "Making Better Maxent Models of Species Distributions: Complexity, Overfitting and Evaluation." *Journal of Biogeography* 41 (4): 629–43.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. 2023. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.

Zhu, Hao. 2023. *kableExtra: Construct Complex Table with 'Kable' and Pipe Syntax*.