Wallace Session 2020-08-09

Please find below the R code history from your *Wallace* v1.0.6.2 session.

You can reproduce your session results by running this R Markdown file in RStudio.

Each code block is called a “chunk”, and you can run them either one-by-one or all at once by choosing an option in the “Run” menu at the top-right corner of the “Source” pane in RStudio.

For more detailed information see <http://rmarkdown.rstudio.com>).

### Package installation

Wallace uses the following R packages that must be installed and loaded before starting.

library(spocc)  
library(spThin)  
library(dismo)  
library(rgeos)  
library(ENMeval)  
library(dplyr)

Wallace also includes several functions developed to help integrate different packages and some additional functionality. For this reason, it is necessary to load the file functions.R, The function system.file() finds this script, and source() loads it.

source(system.file('shiny/funcs', 'functions.R', package = 'wallace'))

## Record of analysis for *Trochodendron aralioides*.

### Obtain Occurrence Data

The search for occurrences was limited to 1000 records. Obtain occurrence records of the selected species from the gbif database.

# query selected database for occurrence records  
results <- spocc::occ(query = "Trochodendron aralioides", from = "gbif", limit = 1000, has\_coords = TRUE)  
# retrieve data table from spocc object  
results.data <- results[["gbif"]]$data[[formatSpName("Trochodendron aralioides")]]  
# remove rows with duplicate coordinates  
occs.dups <- duplicated(results.data[c('longitude', 'latitude')])  
occs <- results.data[!occs.dups,]  
# make sure latitude and longitude are numeric (sometimes they are characters)  
occs$latitude <- as.numeric(occs$latitude)  
occs$longitude <- as.numeric(occs$longitude)  
# give all records a unique ID  
occs$occID <- row.names(occs)

### Process Occurrence Data

Remove the occurrence localities with the following IDs: 801, 785, 287, 787, 690, 225, 297, 224, 242, 366, 396.

# remove the rows that match the occIDs selected  
occs <- occs %>% filter(!(occID %in% c(801, 785, 287, 787, 690, 225, 297, 224, 242, 366, 396)))

The following code recreates the polygon used to select occurrences to keep in the analysis.

selCoords <- data.frame(x = c(119.6737, 119.6737, 124.6414, 124.5315, 119.6737), y = c(25.344, 21.7901, 21.8615, 25.3937, 25.344))  
selPoly <- sp::SpatialPolygons(list(sp::Polygons(list(sp::Polygon(selCoords)), ID=1)))  
occs.xy <- occs[c('longitude', 'latitude')]  
sp::coordinates(occs.xy) <- ~ longitude + latitude  
intersect <- sp::over(occs.xy, selPoly)  
intersect.rowNums <- as.numeric(which(!(is.na(intersect))))  
occs <- occs[intersect.rowNums, ]

Spatial thinning selected. Thin distance selected is 10 km.

output <- spThin::thin(occs, 'latitude', 'longitude', 'name', thin.par = 10, reps = 100, locs.thinned.list.return = TRUE, write.files = FALSE, verbose = FALSE)

Since spThin did 100 iterations, there are 100 different variations of how it thinned your occurrence localities. As there is a stochastic element in the algorithm, some iterations may include more localities than the others, and we need to make sure we maximize the number of localities we proceed with.

# find the iteration that returns the max number of occurrences  
maxThin <- which(sapply(output, nrow) == max(sapply(output, nrow)))  
# if there's more than one max, pick the first one  
maxThin <- output[[ifelse(length(maxThin) > 1, maxThin[1], maxThin)]]   
# subset occs to match only thinned occs  
occs <- occs[as.numeric(rownames(maxThin)),]

### Obtain Environmental Data

Using WorldClim (<http://www.worldclim.org/>) bioclimatic dataset at resolution of 2.5 arcmin.

# get WorldClim bioclimatic variable rasters  
envs <- raster::getData(name = "worldclim", var = "bio", res = 2.5, lat = 22.87, lon = 122.191)  
# change names rasters variables  
envRes <- 2.5  
if (envRes == 0.5) {  
 i <- grep('\_', names(envs))  
 editNames <- sapply(strsplit(names(envs)[i], '\_'), function(x) x[1])  
 names(envs)[i] <- editNames  
}  
i <- grep('bio[0-9]$', names(envs))  
editNames <- paste('bio', sapply(strsplit(names(envs)[i], 'bio'), function(x) x[2]), sep='0')  
names(envs)[i] <- editNames  
# subset by those variables selected  
envs <- envs[[c('bio01', 'bio02', 'bio03', 'bio04', 'bio05', 'bio06', 'bio07', 'bio08', 'bio09', 'bio10', 'bio11', 'bio12', 'bio13', 'bio14', 'bio15', 'bio16', 'bio17', 'bio18', 'bio19')]]  
# extract environmental values at occ grid cells  
locs.vals <- raster::extract(envs[[1]], occs[, c('longitude', 'latitude')])  
# remove occs without environmental values  
occs <- occs[!is.na(locs.vals), ]

### Process Environmental Data

Background selection technique chosen as Minimum Convex Polygon.

occs.xy <- occs[c('longitude', 'latitude')]  
sp::coordinates(occs.xy) <- ~ longitude + latitude  
bgExt <- mcp(occs.xy)

Buffer size of the study extent polygon defined as 1 degrees.

bgExt <- rgeos::gBuffer(bgExt, width = 1)

Mask environmental variables by Minimum Convex Polygon, and take a random sample of background values from the study extent. As the sample is random, your results may be different than those in the session. If there seems to be too much variability in these background samples, try increasing the number from 10,000 to something higher (e.g. 50,000 or 100,000). The better your background sample, the less variability you’ll have between runs.

# crop the environmental rasters by the background extent shape  
envsBgCrop <- raster::crop(envs, bgExt)  
# mask the background extent shape from the cropped raster  
envsBgMsk <- raster::mask(envsBgCrop, bgExt)  
# sample random background points  
bg.xy <- dismo::randomPoints(envsBgMsk, 10000)  
# convert matrix output to data frame  
bg.xy <- as.data.frame(bg.xy)

### Partition Occurrence Data

Occurrence data is now partitioned for cross-validation, a method that iteratively builds a model on all but one group and evaluates that model on the left-out group.

For example, if the data is partitioned into 3 groups A, B, and C, a model is first built with groups A and B and is evaluated on C. This is repeated by building a model with B and C and evaluating on A, and so on until all combinations are done.

Cross-validation operates under the assumption that the groups are independent of each other, which may or may not be a safe assumption for your dataset. Spatial partitioning is one way to ensure more independence between groups.

You selected to partition your occurrence data by the method.

occs.xy <- occs[c('longitude', 'latitude')]  
group.data <- ENMeval::get.block(occ=occs.xy, bg.coords=bg.xy)

# pull out the occurrence and background partition group numbers from the list  
occs.grp <- group.data[[1]]  
bg.grp <- group.data[[2]]

### Build and Evaluate Niche Model

You selected the maxent model.

# define the vector of regularization multipliers to test  
rms <- seq(1, 3, 0.5)  
# iterate model building over all chosen parameter settings  
e <- ENMeval::ENMevaluate(occs.xy, envsBgMsk, bg.coords = bg.xy, RMvalues = rms, fc = c('L', 'LQ', 'H', 'LQH'),   
 method = 'user', occs.grp, bg.grp, clamp = TRUE, algorithm = "maxent.jar")  
  
# unpack the results data frame, the list of models, and the RasterStack of raw predictions  
evalTbl <- e@results  
evalMods <- e@models  
names(evalMods) <- e@results$settings  
evalPreds <- e@predictions

# view response curves for environmental variables with non-zero coefficients  
dismo::response(evalMods[["L\_2"]], var = c('bio02', 'bio09', 'bio14', 'bio18', 'bio19'))

# view ENMeval results  
ENMeval::eval.plot(evalTbl, value = "avg.test.AUC")

# Select your model from the models list  
mod <- evalMods[["L\_2"]]

# generate raw prediction  
pred <- evalPreds[["L\_2"]]

# plot the model prediction  
plot(pred)

### Project Niche Model

You selected to project your model. First define a polygon with the coordinates you chose, then crop and mask your predictor rasters. Finally, predict suitability values for these new raster cells based on the model you selected.

projCoords <- data.frame(x = c(120.2674, 118.7951, 119.6741, 123.1461, 123.6735, 120.2674), y = c(25.8427, 23.6076, 21.2314, 21.1699, 25.6547, 25.8427))  
projPoly <- sp::SpatialPolygons(list(sp::Polygons(list(sp::Polygon(projCoords)), ID=1)))

### Project Niche Model to New Time

Now download the future climate variables chosen with *Wallace*, crop and mask them by projPoly, and use the dismo::predict() function to predict the values for the new time based on the model selected.

envsFuture <- raster::getData("CMIP5", var = "bio", res = 2.5, rcp = 45, model = "CC", year = 50)  
  
predsProj <- raster::crop(envsFuture, projPoly)  
predsProj <- raster::mask(predsProj, projPoly)  
  
# rename future climate variable names  
names(predsProj) <- paste0('bio', sprintf("%02d", 1:19))  
# select climate variables  
predsProj <- raster::subset(predsProj, names(envs))

# predict model  
proj <- dismo::predict(mod, predsProj, args = c('outputformat=raw'))

# plot the model prediction  
plot(proj)

### Calculate Environmental Similarity

To visualize the environmental difference between the occurrence localities and your selected projection extent, calculate a multidimensional environmental similarity surface (MESS). High negative values mean great difference, whereas high positive values mean great similarity. Interpreting the projected suitability for areas with high negative values should be done with extreme caution, as they are outside the environmental range of the occurrence localities.

# extract environmental values from occurrence localities and background -- these were the values that went into the model  
names(bg.xy) <- names(occs.xy)  
all.xy <- rbind(occs.xy, bg.xy)  
occEnvVals <- raster::extract(envs, all.xy)  
# compare these values with the projection extent (envsMsk)  
proj.mess <- dismo::mess(predsProj, occEnvVals)  
plot(proj.mess)