Intro to NextGen ENM/SDM

Jesús Pinto-Ledezma

In this tutorial, we will construct some correlative models using data at species level (occurrence data points) and environmental data derived from remote sensing. We will download occurrence data for oak species distributed in the Americas from **GBIF** and environmental (temperature and precipitation) data from **NASA** and **CHIRPS** servers.

Note that part of the explanation of the different algorithms used here was extracted from the vignette of the {**dismo**} R package.

Set up your data and your working directory

Set up a working directory and put the two data files in that directory. Tell R that this is the directory you will be using, and read in your data:

```
setwd("Desktop/BII_NexGenSDM")

dir.create("Data")
# Raster data
dir.create("Data/raster")
dir.create("Data/raster/EOBioclim")
dir.create("Data/raster/LAI")
dir.create("Data/raster/Topography")
# Occurrence data
dir.create("Data/occ")
# Results
dir.create("Output")
```

Install and load the following packages.

```
packages <- c("raster", "sp", "rgeos", "rworldmap",
              "corrplot", "dismo", "rgdal", "maptools",
              "rgbif", "spThin", "corrplot", "sdmvspecies", "mmap")

# Install packages not yet installed
installed_packages <- packages %in% rownames(installed.packages())

if (any(installed_packages == FALSE)) {
  install.packages(packages[!installed_packages], dependencies = TRUE)
}

# Install the package from source
remotes::install_github("ropensci/scrubr")
remotes::install_git("https://gitup.uni-potsdam.de/macroecology/mecofun.git")
```

Although we installed several packages in the previous lines a package is missing, the R package {**sdm**}. This package is the engine for modeling the environmental niches (**shadows**) or Grinnellian niches. We will install

this package separately because it has several dependencies.

```
install.packages("sdm", dependencies = TRUE)
```

A last step for setting up the R package {**sdm**} is to use the command **installAll()** this will install any missing package and will guarantee the complete functionality if the package. Yes, this is annoying but is the way of how the package was designed.

```
sdm::installAll()
```

Once we get all the packages installed, we can load them to start our tutorial.

```
sapply(packages, require, character.only = TRUE)
library(scrubr)
library(mecofun)
library(sdm)
```

Prepare species data

Get some occurrence data for our species from GBIF, directly within R. This may take some time, given the number of occurrences for the selected species.

NOTE: we need to have an internet connection.

Set the vector with some species scientific names to download occurrence data from **GBIF**.

```
spp <- c("Quercus virginiana", "Quercus minima", "Quercus alba", "Quercus fusiformis")
```

Download data using the vector of species names

```
gbif_data <- occ_data(scientificName = spp, hasCoordinate = TRUE, limit = 2000)

#decrease the 'limit' if you just want to see how many records there are without waiting all the time t

# Save the raw data
save(gbif_data, file = "Data/occ/oaks_raw_occ.RDATA")
```

Inspect the downloaded data. Note that we have downloaded data for four species but for simplicity let's pick one species for this tutorial. I will select *Quercus minima* but you can select any other species.

The returned object (gbif_data) is a list of 4 elements (i.e., the number of our species vector), thus, to select the focal species just use [[**position of the species in the vector**]] to get access to the species. For example, I selected *Quercus minima*, so, to get access to this species I just need to type **gbif_data**[[**2**]], let's see:

```
#gbif_data
# if, for any species, "Records found" is larger than "Records returned", you need to increase the 'lim

# check how the data are organized:
names(gbif_data)
names(gbif_data[[2]]$meta)
names(gbif_data[[2]]$data)
```

Wow, there are a bunch of columns, let's select some columns that are relevant for our purposes.

```
# get the columns that matter for mapping and cleaning the occurrence data:
occ_quemin <- gbif_data[[2]]$data[, c("decimalLongitude", "decimalLatitude",
                                      "scientificName", "occurrenceStatus",
                                      "coordinateUncertaintyInMeters",
```

```
                                        "institutionCode", "references")]
head(occ_quemin)
```

```
dim(occ_quemin)
```

Now let's clean a little bit the data. First, removing all NA in the occurrence data.

```
occ_quemin <- subset(occ_quemin, !is.na(decimalLongitude) & !is.na(decimalLatitude))
```

Let's do some further data cleaning with functions of the {**scrubr**} package (but note this cleaning is not exhaustive!). This will clean some improbable coordinates in the data.

```
occ_quemin <- coord_incomplete(coord_imprecise(coord_impossible(coord_unlikely(occ_quemin))))
```

```
dim(occ_quemin)
## 450    7
# show some values
#occ_quemin[1:4, c(1:5, 7:10)]
```

Now transform our data to a spatial object, this will help us to visualize better our data.

```
quemin_spt <- SpatialPointsDataFrame(coords = occ_quemin[, 1:2],
                                     data = data.frame(occ_quemin),
                                     proj4string = CRS("+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs8
```

Let's plot the distribution of *Quercus minima* occurrences.

```
plot(quemin_spt, col = "darkgreen", pch = 16)
plot(countriesCoarse, add = TRUE, lwd = 2)
```

Seems the data is correct but there are two points that are extreme at the East and West, respectively. Let's first clean those two extreme points.

```
occ_quemin <- occ_quemin %>%
  filter(decimalLongitude < -77) %>%
  filter(decimalLongitude > -90)
```

sometimes the data aggregation can cause some troubles in model predictions. To solve this issue we will apply a spatial thinning on the species occurrences, to do so, we will use the package {**spThin**}. Given that our species has a broad distribution let's sett a distance threshold of 5 km between occurrence points.

```
thinning <- thin(
      loc.data =  occ_quemin,
      verbose = FALSE,
      long.col = "decimalLongitude",
      lat.col = "decimalLatitude",
      spec.col = "scientificName",
      thin.par = 5, # points have at least a minimum distance of 5 km from each other
      reps = 1, # you can repeat this as many as you want.
      locs.thinned.list.return = TRUE,
      write.files = FALSE,
      out.dir = "Data/occ/")

thinning <- as.data.frame(thinning)
thinning$Species <- "Quercus_minima"
```

Explore the results of the thinning process. We can see that the number of occurrences of *Quercus minima* decreased from 448 occurrences to 242 occurrences. We will use the thinned data for further analyses.

```
dim(thinning)
head(thinning)
```

Transform the thinned occurrences to a spatial object

```
thinned_spt <- SpatialPointsDataFrame(coords = thinning[, 1:2],
                                      data = thinning,
                                      proj4string = CRS("+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84
```

Now visualize the both spatial objects.

```
plot(quemin_spt, col = "red", pch = 15)
plot(thinned_spt, col = "darkgreen", pch = 16, add = TRUE)
plot(countriesCoarse, add = TRUE, lwd = 2)
```

Now let's save the data processed and keep **thinned_spt** for further analyses.

```
save(occ_quemin, quemin_spt, thinning, thinned_spt,
     file = "Data/occ/quemin_OCC_processed.RData")

rm(gbif_data, occ_quemin, quemin_spt, thinning)
```

Until here we have downloaded and cleaned species occurrence data, now we will prepare the environmental data that is going to be used as predictors for constructing the **next generation ENM/SDM** for our species.

Prepare environmental data

Now we will prepare the environmental data necessary to construct our models. You can download the cleaned data for precipitation and temperature **HERE** or **HERE**. Note that you can download raw temperature data from HERE and raw precipitation data from HERE

Using data from precipitation and temperature we can derive 19 bioclimatic variables similar to those of the **Worldclim**. For simplicity, we call these remote sensing derived variables as **Earth Observation Bioclimatic Variables** or simply **EO-Bioclim** (Pinto-Ledezma & Cavender-Bares 2021).

Preparing environmental data is time consuming, for example, the time to process 19 EO-Bioclimatic variables for the United States will take around 45 minutes. So, we will use data that is already processed.

If you want to take the risk, you can use the next lines of code to process the EO-Bioclim variables

```
source("Code/rfunctions/makeBioVars.R")

#elev <- raster("Data/raster/Topography/US_elevation.tif")

lst <- list.files("Data/raster/MOD11C3v6.0-CHIRPSv2.0_MONTHLY_03m/", pattern = "tif$")

#Prec <- lst[1:12]
#Tmax <- lst[13:24]
#Tmin <- lst[25:36]

#Prec <- stack(paste0("Data/raster/MOD11C3v6.0-CHIRPSv2.0_MONTHLY_03m/", Prec))

#Tmax <- stack(paste0("Data/raster/MOD11C3v6.0-CHIRPSv2.0_MONTHLY_03m/", Tmax))

#Tmin <- stack(paste0("Data/raster/MOD11C3v6.0-CHIRPSv2.0_MONTHLY_03m/", Tmin))
```

```
#makeEOBios(maxTemp = Tmax, minTemp = Tmin, Precip = Prec, elevation = elev)
```

```
eobios <- list.files("Data/raster/EOBioclim/", pattern = "tif$")

EObios <- stack(paste0("Data/raster/EOBioclim/", eobios))
```

Let's explore some details of the bioclimatic data

```
names(EObios)

str(EObios[[1]])
```

Now let's plot some EO-bioclimatic variables, let's say BIO1 and BIO12 that are mean annual temperature and annual precipitation, respectively.

```
plot(EObios$EO_bio1)
```

```
plot(EObios$EO_bio12)
```

We can also load additional remote sensing biophysical variables, such as the dynamic habitat index (DHI) based on Leaf Area Index (LAI) composites (cumulative, minimum and seasonality) and elevation data from Shuttle Radar Topography Mission (SRTM).

Why we use these additional variables? Well, LAI has a strong relationship with climate and captures the dynamics of the growing season, including vegetation seasonality—important for characterizing plant species geographical ranges. SRTM has been used for mapping and monitoring the earth's surface and is an important variable for predicting species distributions and community composition.

```
lai <- list.files("Data/raster/LAI/", pattern = "tif$")

LAI <- stack(paste0("Data/raster/LAI/", lai))
```

Note that LAI layers are at global scale, so we need to crop them at the same extent as the EO-bioclimatic variables

```
LAI <- crop(LAI, extent(EObios))
```

Let's explore the output for LAI seasonality

```
plot(LAI$LAI_e_bios_Seasonality)
```

We can do the same for elevation.

```
elev <-  raster("Data/raster/Topography/elevation_1KMmd_SRTM.tif")
elev <- crop(elev, extent(EObios))
```

And plot the output

```
plot(elev)
```

It is good practice to check if the extent and pixel size are the same for all the variables, so let's take a look.

```
compareRaster(EObios, elev)
```

```
compareRaster(LAI, elev)
compareRaster(LAI, EObios)
```

LAI returned a different extent when compared to the other variables. To solve this issue we need to resample LAI as a function of elevation or EObios.

```
LAI <- resample(x = LAI, y = EObios[[1]])
```

Compare LAI layers once more.

```
compareRaster(LAI, elev)
compareRaster(LAI, EObios)
```

BAM!, now we are ready to start constructing our **next generation ENM/SDM**

But before, let's visualize the environmental data and overlap the occurrences of *Quercus minima*.

```
plot(EObios[[1]]) # mean annual temperature
plot(thinned_spt, col = "red", pch = 16, add = TRUE) # add occurrence records
plot(countriesCoarse, lwd = 2, lty = 2, add = TRUE) # add country borders
```

```
plot(LAI[[3]]) # mean annual temperature
plot(thinned_spt, col = "red", pch = 16, add = TRUE) # add occurrence records
plot(countriesCoarse, lwd = 2, lty = 2, add = TRUE) # add country borders
```

```
plot(elev) # mean annual temperature
plot(thinned_spt, col = "red", pch = 16, add = TRUE) # add occurrence records
plot(countriesCoarse, lwd = 2, lty = 2, add = TRUE) # add country borders
```

Cool!

Accessible area

Before any further analysis we need to define the accessible are for our species in the geographical space, remember our species is mostly distributed in the United States with some occurrences in southeast Canada. To define the accessible area let's use a bounding box around the known species occurrences plus ~200 km beyond that bound. This will give us an approximation of the species dispersal within the geographical domain, i.e., North America. This bounding box represent the component **M** in the **BAM** diagram.

```
### Species specific accessible area
NAs <- subset(countriesCoarse, continent == "North America") # North American Extent

bb <- bbox(thinned_spt) # bounding box

e <- extent(c(bb[1]-2, bb[3]+2, bb[2]-2, bb[4]+2)) # bounding box + 200 km

p <- as(e, 'SpatialPolygons') # transform to polygon
crs(p) <- crs(EObios) # use the geographical reference of the bioclimatic variables

crs(NAs) <- crs(EObios)

out <- gIntersection(NAs, p, byid = FALSE) # use NAs to eliminate areas on the sea
```

Now let's plot the accessible area.

```
plot(EObios[[1]])
plot(p, add = TRUE, lty = 2)
plot(out, add = TRUE, lwd = 2)
#enviSPP <- raster::crop(envi, out)
```

Nice, we can see that the accessible area for our species includes most of Central and East United States and southeastern Canada.

Now crop the environmental data to the extent of the accessible area.

```
EObios_spp <- raster::crop(EObios, out)
EObios_spp <- raster::mask(EObios_spp, out)

plot(EObios_spp[[1]])
plot(thinned_spt, add = TRUE, col = "red", pch = 16)
plot(NAs, add = TRUE, lty = 2)
```

Now, LAI.

```
LAI_spp <- raster::crop(LAI, out)
LAI_spp <- raster::mask(LAI_spp, out)

plot(LAI_spp[[3]])
plot(thinned_spt, add = TRUE, col = "red", pch = 16)
plot(NAs, add = TRUE, lty = 2)
```

And finally elevation

```
elev_spp <- raster::crop(elev, out)
elev_spp <- raster::mask(elev_spp, out)

plot(elev_spp[[1]])
plot(thinned_spt, add = TRUE, col = "red", pch = 16)
plot(NAs, add = TRUE, lty = 2)
```

We will use this accessible area as extent for all the posterior analyses.

Pseudoabsences

To model the environmental niches (**ghosts**) and to project the species distributions (**shadows**), data on species absences is necessary, however we only have presence data. What should we do if we don't have absences? That's the question! There is no a straightforward answer for that question, but the most common procedure to get absence data is to generate random points (AKA **pseudoabsences**) on the geographical domain (i.e., the accessible area or **M**). There is a lot of discussion on how many pseudoabsences we need to use for ENM/SDM, here we will use a conservative number, i.e., twice the number presences.

```
set.seed(12345) # Random Number Generation to obtain the same result
# Generate the data
absence <- randomPoints(mask = EObios_spp[[1]],
                        n = round(nrow(thinned_spt)*2, 0), # number of pseudoabsences
                        p = thinned_spt, ext = extent(EObios_spp))
```

Now let's combine the presence and pseudoabsence data

```
presence <- data.frame(coordinates(thinned_spt)) # presence data
absence <- data.frame(absence) # pseudoabsence data
names(absence) <- names(presence)

presence$Occurrence <- 1 # presence data
absence$Occurrence <- 0 # pseudoabsence data

quemin <- rbind(presence, absence) # combine both information
quemin$Species <- "Quercus_minima"
```

Finally transform the presence-pseudoabsence data to a spatial object and visualize it!

```
coordinates(quemin) <- ~ Longitude + Latitude
crs(quemin) <- crs(EObios_spp)

quemin
```

```
plot(EObios_spp[[1]])
plot(quemin[quemin$Occurrence == 1, ], col = "blue", add = TRUE, pch = 16)
points(quemin[quemin$Occurrence == 0, ], col = "red", pch = 16)
```

We can save the processed data and clean the environment a little bit.

```
save(presence, absence, quemin, file = "Data/occ/quemin_PresAbs.RData")

save(bb, e, NAs, out, p, file = "Data/occ/accessible_area_quemin.RData")

rm(absence, presence, EObios, e, out, p, bb)
```

Now, we need to decide which variables are necessary to model the niche (Grinnellian niche or Fundamental niche or the Ghost) for our oak species. Here we can rely the selection to the specialist or use statistical tools or both.

Variable selection

Let's use the statistical way. However we can ask Jeannine about which variables are more important for the distribution of oaks.

```
quemin_EObios <- data.frame(raster::extract(EObios_spp, quemin))

quemin_EObios <- cbind(data.frame(quemin), quemin_EObios)

quemin_EObios <- quemin_EObios[complete.cases(quemin_EObios), ]
quemin_EObios <- na.omit(quemin_EObios)

head(quemin_EObios)
```

One way to select variables is to explore the correlation among the predictors. We can use a threshold of **|r| <0.7** which means that correlated variables below this threshold can be considered as no problematic **(Dormann et al. 2013)**, however you can use a more conservative threshold, such as <0.5. Let's see.

```
# We first estimate a correlation matrix from the predictors. We use Spearman rank correlation coeffici
cor_mat <- cor(quemin_EObios[, c(6:24)], method = 'spearman')
```

Let's visualize the correlation matrix.

```
corrplot.mixed(cor_mat, tl.pos = "lt", tl.cex = 0.5, number.cex = 0.5,
               addCoefasPercent = TRUE, mar = c(0, 0, 1, 0))
```

Looks like that several variables are highly correlated, but let's select the ones that are highly correlated with each other and exclude them for further analysis. However inspecting predictor by predictor can take forever, what should we do? Well we have two options:

1. Talk with the specialist about which predictors should we use, or
2. Let the computer decide

The simple solution is letting the computer decide, thus, for doing that, we will use an amazing function called '**select07()**' from the package {**mecofun**} that will select the predictors bellow the correlation threshold.

Let's install the package from its repo as it is not currently published in CRAN. Important, a message will

appear in your console asking if you would like to update an existing package, please type **3** and press **Enter**. This will order **R** to install just {**mecofun**} and also will avoid any error in the installation.

```
# Install the package from source
remotes::install_git("https://gitup.uni-potsdam.de/macroecology/mecofun.git")
```

The function will return a list of three objects: 1. AIC values for each model 2. The correlation matrix 3. A vector of the selected variables.

```
library(mecofun)

# Run select07()

covar_sel <- select07(X = quemin_EObios[, -c(1:5)], # only predictors data
                      y = quemin_EObios$Occurrence, # presence-absence data
                      threshold = 0.7) # here you can change the threshold for one

# Check out the structure of the resulting object:
str(covar_sel)
```

```
covar_sel$AIC # aic values
covar_sel$cor_mat # matrix of pairwise correlation
covar_sel$pred_sel # selected variables
```

According to the **select07()** function six predictors best fit the data and also have low correlation. Assuming that this is correct, we will use these variables for further analyses. The selected variables are: "EO_bio4", "EO_bio17", "EO_bio9", "EO_bio10", "EO_bio12", "EO_bio2".

```
preds <- covar_sel$pred_sel
preds
```

If you have some issues installing the package {**mecofun**} please use the next lines of code. The function **select07_v2** is the very same from the package {mecofun}. In this case I just extracted the function from the package in order to avoid installation issues.

```
#source("R-Functions/select07_mod.R")
source("https://raw.githubusercontent.com/jesusNPL/BiodiversityScience/master/Spring2021/R-Functions/se
# Run select07()
covar_sel <- select07_v2(X = quemin_bios[, -c(1:5)], # only predictors data
                      y = quemin_bios$Occurrence, # presence-absence data
                      threshold = 0.7) # here you can change the threshold for one
```

This extracted function will return the same outputs.

```
preds <- covar_sel$pred_sel
preds
```

Finally, let's select the bioclimatic variables and plot them

```
envi_quemin_sel <- stack(EObios_spp$EO_bio2, EObios_spp$EO_bio4,
                         EObios_spp$EO_bio9, EObios_spp$EO_bio10,
                         EObios_spp$EO_bio12, EObios_spp$EO_bio17,
                         LAI_spp$LAI_e_bios_Seasonality,
                         elev_spp)

plot(envi_quemin_sel)
```

Okay, we re ready to build the models for our species.

Building Environmental Niche Models

To model the environmental niches for *Quercus minima* we will use the R package {**sdm**}. This package is very useful and first we need to create a data object which will save us a lot of time in the future.

```
queminDATA <- sdmData(formula = Occurrence ~ EO_bio2 + EO_bio4 + EO_bio9 + EO_bio10 + EO_bio12 + EO_bio
                      train = quemin, # presence-pseudoabsence data
                      predictors = envi_quemin_sel, # selected covariables
                      crs = crs(bios_quemin_sel))
```

Let's explore a little bit the **sdm** object.

```
queminDATA
```

We can see that our object has 1 species, 8 features or predictors and the type of data is presence-absence (well is pseudoabsence) and the number of records.

Now let's model the species-environment relationship or Grinnellian niches. Under {**sdm**} we can model the species-environment relationships using several algorithms at once, but it depend completely on the power of your computer, so use it at your own risk. To get the complete list of algorithms available in the {**sdm**} package just type **getmethodNames('sdm')** in your console.

```
getmethodNames('sdm')
```

Here we will explore the predictions of six algorithms, i.e., Bioclim, Gower, GLMs GAM, SVM and Random forest.

Fit models

Bioclim model

The BIOCLIM algorithm computes the similarity of a location by comparing the values of environmental variables at any location to a percentile distribution of the values at known locations of occurrence ('training sites').

Gower model

The Domain algorithm computes the Gower distance between environmental variables at any location and those at any of the known locations of occurrence ('training sites'). For each variable the minimum distance between a site and any of the training points is taken.

Generalized Linear Models

A generalized linear model (GLM) is a generalization of ordinary least squares regression. Models are fit using maximum likelihood and by allowing the linear model to be related to the response variable via a link function and by allowing the magnitude of the variance of each measurement to be a function of its predicted value. Depending on how a GLM is specified it can be equivalent to (multiple) linear regression, logistic regression or Poisson regression.

Random forest

Random Forest (Breiman, 2001) is an extension of the classifictation and regression trees (CART; Breiman et al., 1984).

Support Vector Machines or SVM model

Support Vector Machines are an excellent tool for classification, novelty detection, and regression.

Support Vector Machines (SVMs; Vapnik, 1998) apply a simple linear method to the data but in a high-dimensional feature space non-linearly related to the input space, but in practice, it does not involve any computations in that high-dimensional space. This simplicity combined with state of the art performance on many learning problems (classification, regression, and novelty detection) has contributed to the popularity of the SVM (Karatzoglou et al., 2006).

If you have a powerful computer you can set more cores that will speed the computation time. Additionally, Here we are going to fit 6 models and evaluate them through 2 runs of subsampling, each draw 30 percent of training data as test dataset:

```
#This takes sometime (~3 minutes) please be patient!

queminSDM <- sdm(Occurrence~., data = queminDATA, # formula and data
                 methods = c("bioclim", "domain.dismo", "glm", "gam", "rf", "svm"), # algorithms
                 replication = "sub", test.percent = 30, n = 2, # training-testing subsampling
                 parallelSettings = list(ncore = 8, method = "parallel")) # parallel computation

write.sdm(queminSDM, file = "Output/queminSDM", overwrite = TRUE) # save the model for latter
```

Now let's see how well each algorithm fitted the data.

```
queminSDM
```

```
queminSDM <- read.sdm("Output/queminSDM.sdm")

getModelInfo(queminSDM)

getEvaluation(queminSDM)
```

We can see that all algorithms worked relatively well. The evaluation table shows four metrics of model evaluation, i.e., AUC, COR, TSS and Deviance, each metric has its own properties but I personally like the true-skill statistics (TSS) metric, because it evaluates the matches and mismatches between observations and predictions. So by looking at the column of TSS we can see that the algorithm that best fitted the occurrence data is **svm** followed by **random forest**, and **gam** and **glm**, respectively. Interestingly, these models also have the highest area under the operating curve (AUC) metric values.

Another way to evaluate the model performance is to plot the AUCs.

```
roc(queminSDM, smooth = TRUE)
```

Until here we have downloaded and processed occurrence data and fitted and evaluated six different algorithms. Now let's predict the Grinnellian niche for *Quercus minima* in the accessible area. We can do that for all six algorithms at once but it will take a very long time, so let's do it algorithm by algorithm. In addition, you can use the fitted object (i.e., queminSDM) to make predictions to other regions (e.g., areas with similar environmental conditions or invadable areas) or to project to other periods of time (i.e., to the future or the past).

```
quemin_ENM_bioclim <- predict(queminSDM, newdata = envi_quemin_sel,
                              method = "bioclim", mean = TRUE,
                              parallelSettings = list(ncore = 8, method = "parallel"))
quemin_ENM_bioclim
```

Now plot the resulting prediction under the **bioclim** algorithm. This map is showing the probability of occurrence of *Quercus minima* under the algorithm **bioclim**, where values closest to 1 indicate higher probability and values near 0 low probability of occurrence. Before visualizing the prediction let's re-scale the prediction to make it more interpretable.

```
quemin_ENM_bioclim <- sdmvspecies::rescale(quemin_ENM_bioclim$sp_1.m_bioclim.re_subs)

plot(quemin_ENM_bioclim)
```

Let's repeat the process for the best four algorithms. However, consider that the time to make predictions increase with model complexity, for example, the simple model **bioclim** only takes ~2 minutes (using eight cores), whereas **svm** ~5 minutes. So run model predictions under your own risk.

```
quemin_ENM_glm <- predict(queminSDM, newdata = envi_quemin_sel,
                          method = "glm", mean = TRUE,
                        parallelSettings = list(ncore = 8, method = "parallel"))

quemin_ENM_gam <- predict(queminSDM, newdata = envi_quemin_sel,
                          method = "gam", mean = TRUE,
                        parallelSettings = list(ncore = 8, method = "parallel"))

quemin_ENM_rf <- predict(queminSDM, newdata = envi_quemin_sel,
                          method = "rf", mean = TRUE,
                       parallelSettings = list(ncore = 8, method = "parallel"))

quemin_ENM_svm <- predict(queminSDM, newdata = envi_quemin_sel,
                          method = "svm", mean = TRUE,
                        parallelSettings = list(ncore = 8, method = "parallel"))

# Please, do not pay attention to the warnings...
```

```
quemin_ENM_glm <- sdmvspecies::rescale(quemin_ENM_glm$sp_1.m_glm.re_subs)
quemin_ENM_gam <- sdmvspecies::rescale(quemin_ENM_gam$sp_1.m_gam.re_subs)
quemin_ENM_rf <- sdmvspecies::rescale(quemin_ENM_rf$sp_1.m_rf.re_subs)
quemin_ENM_svm <- sdmvspecies::rescale(quemin_ENM_svm$sp_1.m_svm.re_subs)
```

Now plot the four best predictions

```
quemin_ENM_four <- raster::stack(quemin_ENM_glm, quemin_ENM_gam,
                                 quemin_ENM_svm, quemin_ENM_rf)

names(quemin_ENM_four) <- c("GLM", "GAM", "SVM", "RF")
```

```
plot(quemin_ENM_four)
```

As you can see the four algorithms returned different predictions, for example **GLM** return a broad prediction while **SVM** a narrow prediction. Is there a **silver-bullet** (i.e., perfect algorithm) for model predictions? It seems that it depends. If you are curious about this issue this **paper** led by Huijie Qiao can help you.

One potential solution to avoid this issue is to create an **ensemble** model. Basically an ensemble return a prediction by combining results of different modeling algorithms (Araújo & New, 2007). These ensemble predictions are also know as consensus predictions. Here rather than using the six models we will use the best four models (i.e., svm, rf, gam, and glm) and we will use the TSS metric as weights. Note that this will take around 10 minutes to run using 12 cores.

```
  ### Ensemble prediction - ensemble based on TSS statistics
quemin_ENM_ensemble <- ensemble(queminSDM, newdata = envi_quemin_sel,
                                method = c("rf", "gam", "svm", "glm"),
                                setting = list(method = "weighted", stat = "TSS"),
                                parallelSettings = list(ncore = 12, method = "parallel"))
```

Don't forget to re-scale the model prediction and visualize it.

```
quemin_ENM_ensemble <- sdmvspecies::rescale(quemin_ENM_ensemble)
```

Plot the ensemble prediction

```
plot(quemin_ENM_ensemble)
```

How do you feel about that? Does the ensemble prediction outperform single algorithm predictions?

You might also want to save the predictions.

```
dir.create("Output/ENMs")

writeRaster(quemin_ENM_four, filename = "Output/ENMs/quemin_ENM",
            suffix = names(quemin_ENM_four), format = "GTiff",
            bylayer = TRUE, overwrite = TRUE)

writeRaster(quemin_ENM_ensemble, filename = "Output/ENMs/quemin_ENM_Ensemble",
            format = "GTiff", bylayer = TRUE, overwrite = TRUE)
```

Let's try making a nicer map.

```
library(rasterVis)
library(RColorBrewer)

mapTheme <- rasterTheme(region = rev(brewer.pal(11, "Spectral")),
  layout.widths = list(right.padding = 10),
  axis.line = list(col = "transparent"),
  tick = list(col = 'transparent'))

levelplot(quemin_ENM_ensemble,
  maxpixels = 1e10,
  margin = FALSE,
  par.settings = mapTheme,
  scales = list(x = list(draw = FALSE),
                y = list(draw = FALSE)),
  zlim = c(0, 1))

grid::grid.text('Quercus minima \n Probability of presence',
  rot = 90,
  y = unit(0.5, "npc"),
  x = unit(0.925, "npc"),
  gp = grid::gpar(fontsize = 15))
```

Finally, let's produce "shadows" from the "ghosts".

Building Species Distributions

Species geographical distributions

First we need to set a binarization threshold for the model predictions, the threshold (cut-off) is used to transform model predictions (probabilities, distances, or similar values) to a binary score (presence or absence).

Here we will find the threshold for the **ensemble** prediction, but you can do for each algorithm separately. The threshold we will use intents to maximizes the **sensitivity** (ability of a test to correctly identify presences or true positive rate) plus **specificity** (ability of a test to correctly identify absences or true negative rate) in the predictions.

```
dt <- data.frame(as.data.frame(queminDATA), coordinates(queminDATA))
head(dt)

prediction <- raster::extract(quemin_ENM_ensemble, dt[, c("Longitude", "Latitude")])

evaluation <- sdm::evaluates(dt$Occurrence, prediction) # observed versus expected

threshold_sel <- evaluation@threshold_based$threshold[2]

round(threshold_sel, 2)
```

The threshold that maximizes the sensitivity and specificity is **0.36**, we will use that value to produce "shadows" from the "ghosts".

```
quemin_SDM_ensemble <- quemin_ENM_ensemble

quemin_SDM_ensemble[] <- ifelse(quemin_SDM_ensemble[] >= threshold_sel, 1, 0)
```

Plot the distribution of *Quercus minima*

```
plot(quemin_SDM_ensemble)
plot(countriesCoarse, add = TRUE)
```

That's it, we modeled environmental niches (**ghosts**) to produce geographical distributions (**shadows**).

You might also want to save this final raster file in your hard drive.

```
dir.create("Output/SDMs")

writeRaster(quemin_SDM_ensemble, filename = "Output/SDMs/quemin_SDM_Ensemble",
            format = "GTiff", bylayer = TRUE, overwrite = TRUE)
```

References

Araújo M. B. and New, M. (2007). Ensemble forecasting of species distributions. Trends Ecol Evol 22(1):42–47.

Breiman, L., J. Friedman, C.J. Stone and R.A. Olshen, (1984). Classification and Regression Trees. Chapman & Hall/CRC.

Breiman, L. (2001). Random Forests. Machine Learning 45: 5-32. doi:10.1023/a:1010933404324

Dormann, C. F., Elith, J., Bacher, S., Buchmann, C., Carl, G., Carré, G., ... Lautenbach, S. (2012). Collinearity: a review of methods to deal with it and a simulation study evaluating their performance. Ecography, 36(1), 27–46. doi:10.1111/j.1600-0587.2012.07348.x

Karatzoglou, A., D. Meyer & K. Hornik, 2006. Support Vector Machines in R. Journal of statistical software 15(9).

Pinto-Ledezma, J.N. & J. Cavender-Bares. 2021. Predicting species distributions and community composition using satellite remote sensing predictors. Scientific Reports, 11: 16448.

Vapnik, V., 1998. Statistical Learning Theory. Wiley, New York.