Introduction

Defining where species are found is challenging because biodiversity is complex, and human sampling errors make it even harder. This short project aims to tackle a common sampling mistake by using **virtual species** distributions and **ecological niches**.

The use of empirical data when assessing an error is problematic because each dataset has many confounding factors, which preclude generalisation. A valuable alternative is the simulation of virtual species distributions, because underlying mechanisms that generate such distribution patterns are known.

A sampling bias in a species often leads to a sample of occurrences that doesn't represent its ecological niche, which is the role and conditions in which that species survives. Comparing the realized ecological niche of the species to that derived from the 'biased' subsample gives us an idea of the discrepancy.

Input data

Region of interest

The upload of the area of interest in the form of a **shapefile** is required. It is important to be in the correct folder, within which all the extensions must be present: .shp, .dbf, .shx, .prj.

Our area of interest is the province of Parma, located in Emilia Romagna, in northern Italy

```r
# Loading required package
library(sf)

# Set working directory
setwd("C:/chelsa")

# Upload shapefile
aoi <- st_read("parma.shp")
aoi <- aoi$geometry
plot(aoi)
```

Data downloading from CHELSA

The input data for virtual species is environmental spatial data (**raster data**).

CHELSA (Climatologies at high resolution for the earth's land surface areas) is a very high resolution (30 arc sec, ~1km) global downscaled climate data set: it is built to provide free access to high resolution climate data for research and application, and is constantly updated and refined.

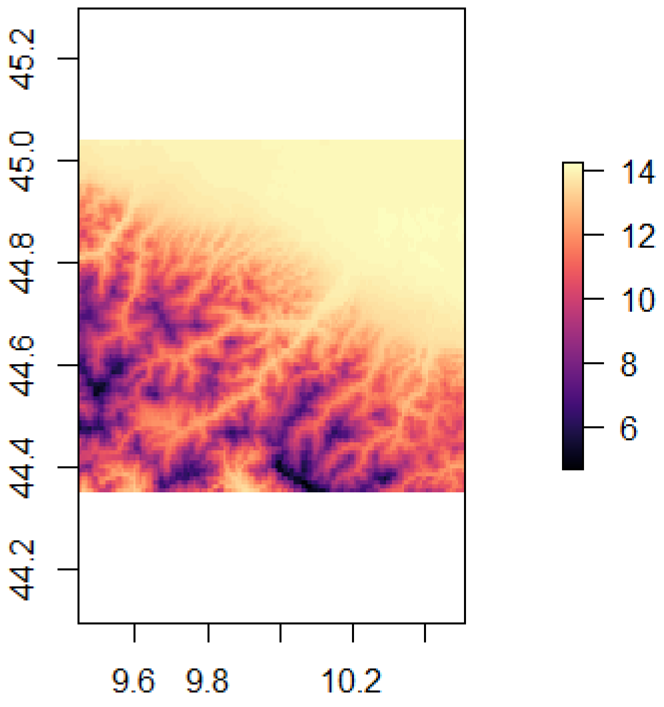
Bioclimatic variables are derived variables developed for species distribution modeling and related ecological applications: https://chelsa-climate.org/bioclim

You can directly download CHELSA data into R. The download creates a series of folders: it is necessary to move to the correct one to work on the downloaded files.

```r
# Loading required package
library(ClimDatDownloadR)

# Alternatively, a direct link with GitHub can be created
if(!require(devtools)) install.packages("devtools")
library(devtools)
devtools::install_github("HelgeJentsch/ClimDatDownloadR")

# Download bioclimatic variables from CHELSA
Chelsa.Clim.download(

  # Starting from the workind directory, specify the path
  save.location = "strade_parma",

  # 'bio' contains all the bioclimatic variables
  parameter = "bio",

  # Some variables are chosen from the 19 available
  bio.var = c(1, 7, 13, 12, 14),

  # Version
  version.var = "2.1",

  # Cropping along the area of interest
  clipping = TRUE,
  clip.shapefile = aoi,

  # Insert the coordinates of the area of interest (bounding box)
  clip.extent = c(9.439404, 44.34708, 10.50532, 45.04535),

  # Buffer, if needed
  # buffer = 3,

  # Other commands
  convert.files.to.asc = FALSE,
  stacking.data = TRUE,
  combine.raw.zip = FALSE,
  delete.raw.data = FALSE,
  save.bib.file = TRUE
)
```

The required format is a `RasterStack` containing all the environmental variables with which you want to generate a virtual species distribution.

```
# Loading required packages
library(raster)
library(viridis)

# String containing the names of raster files
rastlist <- list.files(path
="strade_parma/bio/ChelsaV2.1Climatologies/clipped_2024-02-28_09-42-52",
pattern = "CHELSA", full.names = TRUE)

# Using the list of names, all the files are imported into a single raster
package
mydata <- stack(rastlist)

# Change data names
names(mydata) <- c("mean annual T", "annual range air T", "annual precip",
"amount of prec. wettest month", "amount prec. driest month")

# bio1
plot(mydata[[1]], col =  magma(500, alpha = 1, begin = 0, end = 1,
direction = 1))
```



bio1 is the mean annual air temperature

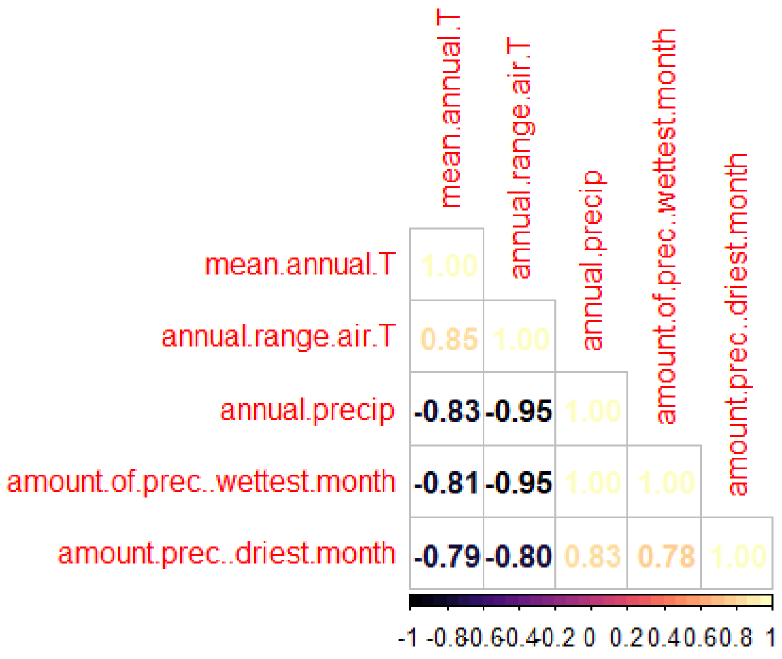
Correlation Matrix

In studying species distribution based on climate variables, conducting a correlation analysis is crucial.

This analysis helps identify which climate variables are strongly associated with each other and with species distribution patterns. Avoiding highly correlated variables is important because they provide redundant information, potentially leading to multicollinearity issues in statistical models.

By focusing on uncorrelated variables, models become more interpretable and accurately capture the relationships between climate and species distribution. This enhances predictive accuracy and conserves computational resources by excluding unnecessary variables.

```r
# Loading required package
library(corrplot)

# Subsample 10% of pixels and calculate pairwise correlations
r1 <- mydata$mean.annual.T
cor <- cor(sampleRandom(mydata, size= ncell(r1) * 0.30 ), method =
"pearson")

# Plot correlation matrix
df <- corrplot(cor, method = "number", col =  magma(30, alpha = 1, begin =
0, end = 1, direction = 1), type = "lower", tl.pos = 'ld')
```



Correlation matrix

Random species generation

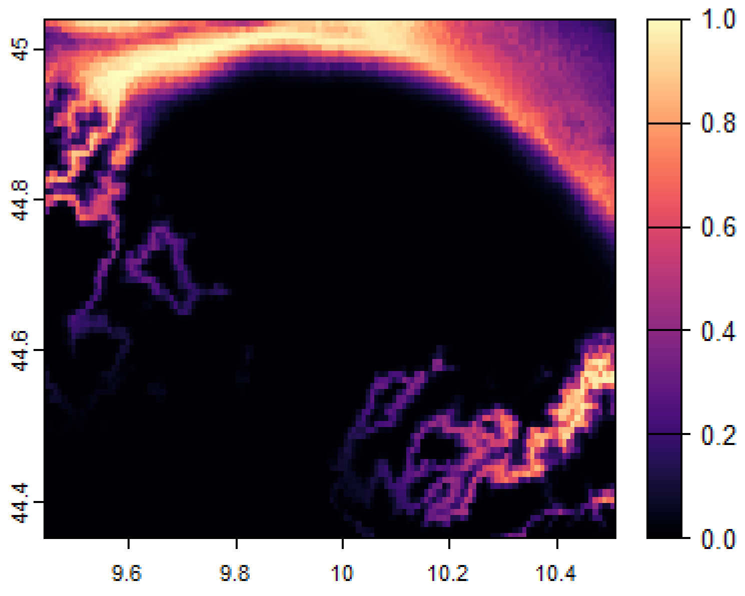
Generating random species from known environmental data allows controlling the factors that can influence the distribution of real data. To create a series of occurrence points for a species, it is necessary to go through 3 steps

First step: suitability

By intersecting bioclimatic data, the first output obtained is a suitability map. In creating this map, various parameters can be set, allowing for closer approximation to a real-world scenario.

- `species.type`. There are two main possibilities to combine response functions to calculate the environmental suitability: **additive** (the response functions are added) and **multiplicative** (the response functions are multiplied, default).
- `approach` By default, the function `generateRandomSp` uses a **PCA** approach if you have 6 or more variables, and a '**response functions**' approach if you have less than 6 variables.
- `relations`. Four relations are possible for a random generation of virtual species: **gaussian**, **linear**, **logistic** and **quadratic** relations. By default, all the relation types are used. You can choose to use any combination inside the argument.

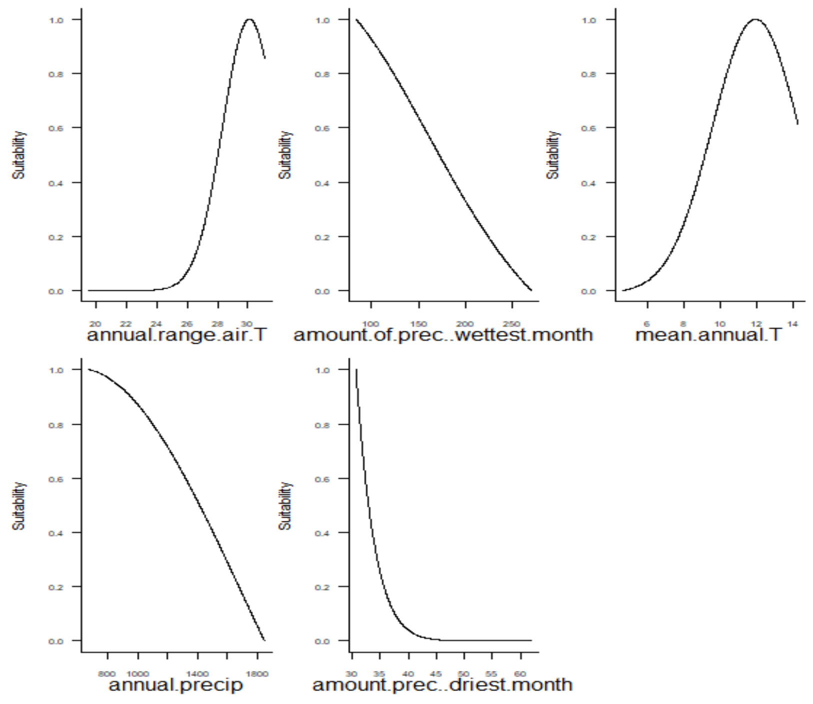
```r
# Loading required packages
library(virtualspecies)
library(ggplot2)
library(tidyverse)

# Suitability map generation
random.sp <- generateRandomSp(raster.stack = mydata,
                              convert.to.PA = FALSE,

                              # How to combine response functions
                              species.type = "multiplicative",

                              # Random approach between PCA and response
function

                              approach = "random",

                              # Response function
                              relations = "gaussian",

                              # Realistic species
                              realistic.sp = TRUE,
                              plot = FALSE)

plot(random.sp$suitab.raster, col = magma(500, alpha = 1, begin = 0, end =
1, direction = 1))
```



Suitability map

You can visualize how each variable contributed to generating the suitability map using the `plotResponse` command

```
plotResponse(random.sp)
```



Response functions

Second step: presence/absence

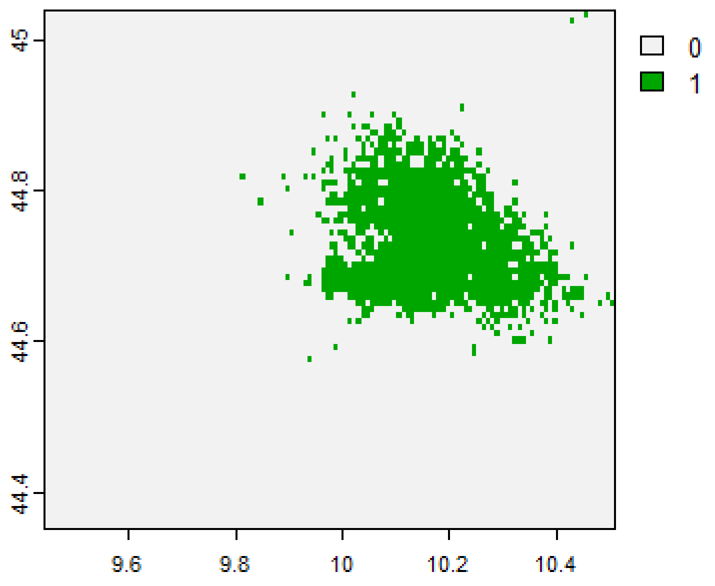
In the second step, the suitability map is converted into a binary **presence/absence map** through a probability function (logistic curve) that associates the suitability value with the probability of finding the virtual species for each pixel.

This subset of the environmental niche that is actually occupied by the species corresponds to the **realized niche** (*Hutchinson, 1957*).

Customizing the function is also possible in this step by the parameters `alpha` and `beta` which determine the shape of the logistic curve.

- `alpha` drives the 'slope' of the curve.
- `beta` controls the inflexion point. A higher beta will decrease the probability of finding suitable conditions (smaller distribution range).
- `species.prevalence`. The species prevalence is the number of places (here, pixels) actually occupied by the species out of the total number of places (pixels) available.

```
# Presence/Absence: requires defining the parameters alpha, beta, and
species prevalence
new.pres <-convertToPA(random.sp,
                       beta = "random",
                       alpha = -0.05, plot = FALSE,
                       species.prevalence = 0.1)

plot(new.pres$pa.raster)
```



Realized niche

Third step: occurrences

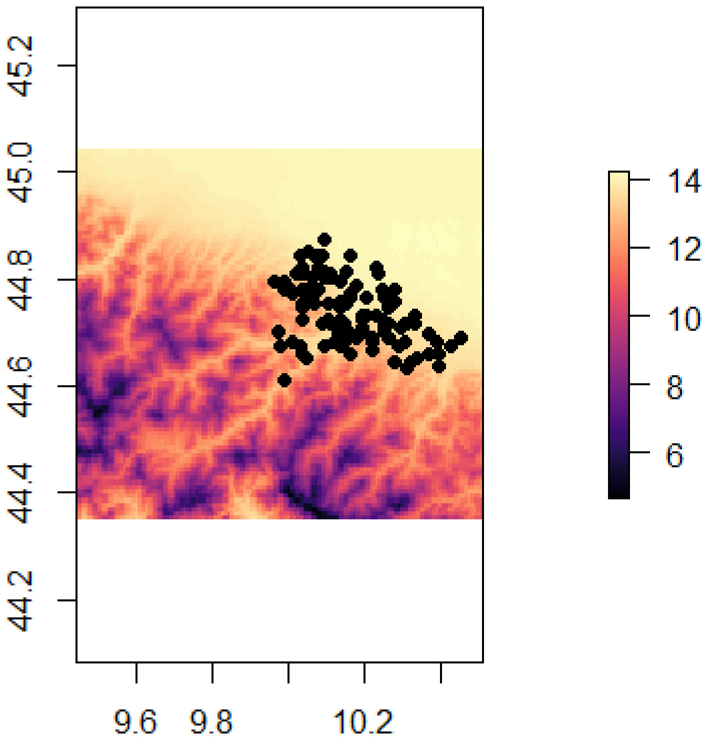
The third step consists of generating, within the presence/absence raster, a series of occurrence points. This step involves many parameters that are important to test.

- `n`: number of occurrences
- `type`. The type of occurrences can be 'presence-absence' or 'presence only'. In the case of choosing the second type, points will be generated within the realized niche, so the maximum number of points allowed is equal to the quantity of pixels present in the presence-absence raster with a value of 1. You can easily find this number (ncell) with the following command:

```
 > new.pres$pa.raster %>% raster()
class      : RasterLayer
dimensions : 83, 128, 10624  (nrow, ncol, ncell)
...
```

- `sample.prevalence`: the sample prevalence is the proportion of samples in which the species has been found.
- `error.probability`: the probability of an erroneous detection of our virtual species. It's a human error and could depend, for example, on the sampler's ability to correctly recognize a species: the error is common in the case of plants. The error probability is useless in a ‘presence only’ sampling scheme, because the sampling strictly occurs within the boundaries of the species distribution range. Nevertheless, if you still want to introduce errors, then make a sampling scheme ‘presence-absence’, and use only the ‘presence’ points obtained.
- `detection.probability`: the detection probability of our virtual species, ranging from 0 (species cannot be detected) to 1 (species is always detected). This factor, instead, depends on the species: there are species more likely to be sampled.
- `correct.by.suitability` a parameter you can set to complexify the detection probability by making it dependent on the environmental suitability. In this case, cells will be weighted by the environmental suitability: less suitable cells will have a lesser chance of detecting the species.

```
  presence.points <- sampleOccurrences(new.pres,
                                    n = 100,
                                    type = "presence only",
                                    sample.prevalence = 0.9,
                                    error.probability = 0,
                                    detection.probability = 1,
                                    correct.by.suitability = TRUE,
                                    plot = FALSE)

# Plot
plot(mydata[[1]], col =  magma(500, alpha = 1, begin = 0, end = 1,
direction = 1))
points(presence.points$sample.points, col = "black", pch = 19)
```



Occurrences of the random species

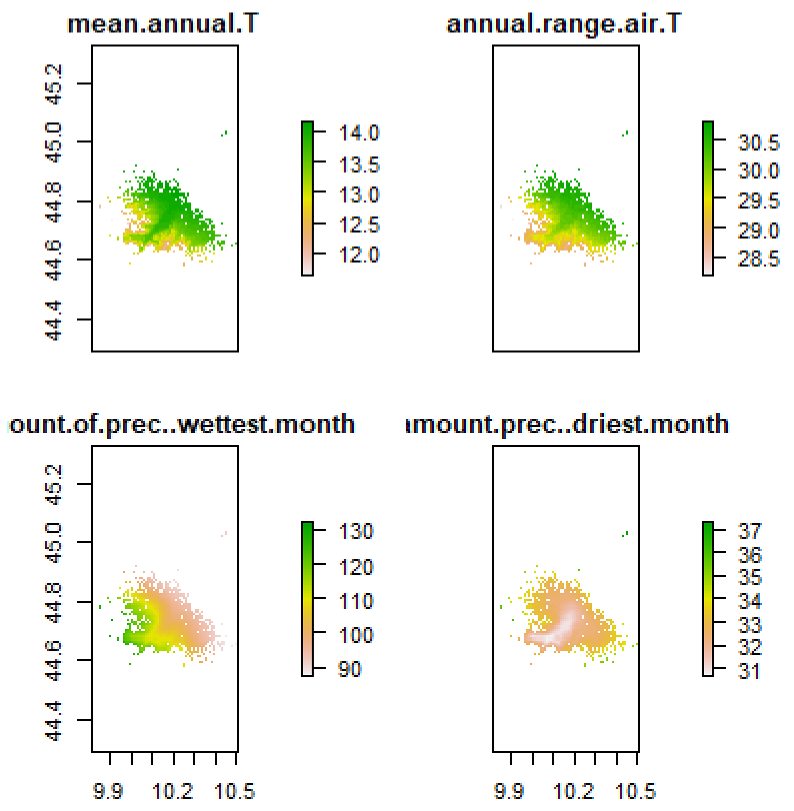
Preliminary Steps for Niche Analysis

The following are a series of necessary steps to associate the corresponding bioclimatic variables with the realized niche (step 2) and occurrence points (step 3).

```
# Raster with presence/absence points (step 2): used to create the realized
niche
# It must be a RasterLayer object
raster01 <- new.pres$pa.raster %>% raster()

# The bioclimatic layers are extracted one by one from the stack
r1 <- mydata$mean.annual.T
r2 <- mydata$annual.range.air.T
r3 <- mydata$annual.precip
r4 <- mydata$amount.of.prec..wettest.month
r5 <- mydata$amount.prec..driest.month

# A stack is created containing the bioclimatic variables and the raster of
presence/absence (realized niche)
stack_pa <- brick(r1, r2, r3, r4, r5, raster01)

# The values are extracted from the stack, converted into a dataframe, and
only the presence pixels (1) are retained
values <- stack_pa %>% rasterToPoints() %>% as.data.frame()
filtered_pa <- values %>% filter(., lyr.1 == 1) %>% as.data.frame()
```

This way, `filtered_pa` is obtained, a dataframe containing the points of the realized niche and their corresponding bioclimatic variables.

```
# Conversion from a dataframe to a raster
raster_pa <- filtered_pa %>% .[,-8] %>% rasterFromXYZ()
plot(raster_pa)
```



Bioclimatic variables only corresponding to the realized niche

The same steps will be performed for the occurrence points of the species

```
# The raster of occurrences is transformed into a dataset, from which the
rows satisfying both conditions Real = 1 and Observed = 1 are preserved
raster_occurences <- presence.points$sample.points %>% as.data.frame() %>%
.[.$Real == 1 & .$Observed == 1, ]

# The environmental variables are associated with the occurrences using
their coordinates
stack_occ <- brick(r1, r2, r3, r4, r5)
values_occ <- stack_occ %>%  rasterToPoints() %>% as.data.frame()
filtered_occ <- merge(values_occ, raster_occurences, by = c("x", "y"))
```

`filtered_occ` contains the coordinates of occurrence points and their corresponding environmental variables

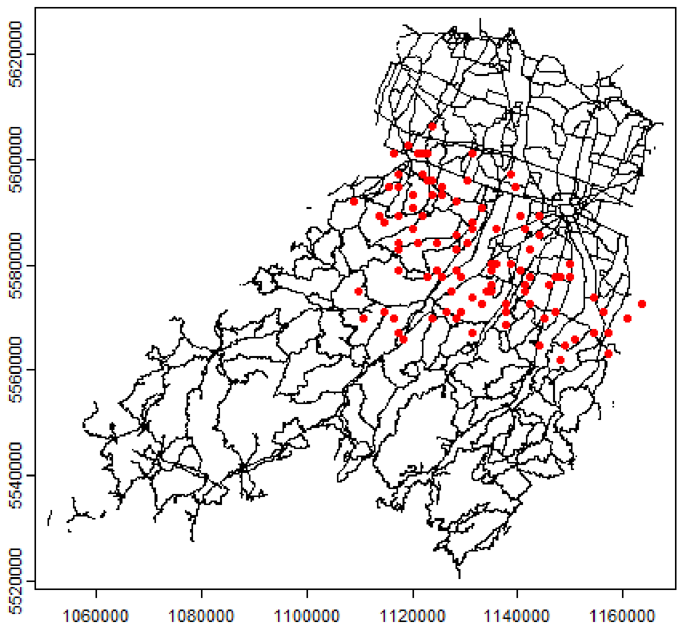
Introducing roadside bias

One of the most recognized forms of bias in distributional data is the high concentration of observations (or collection sites) along road: models based on data collected along roads should lead to inaccurate predictions when applied to larger area (*Kadmon et al., 2004*)

The main mechanism by which roadside bias can affect the accuracy of bioclimatic models is through climatic bias in the distribution of the road network.

Our goal is to associate the occurrence points with the probability of being sampled, knowing that as one moves away from roads, the probability decreases: the first step involves calculating the distance of the points from the nearest road. In our case, the road network is that of the province of Parma.

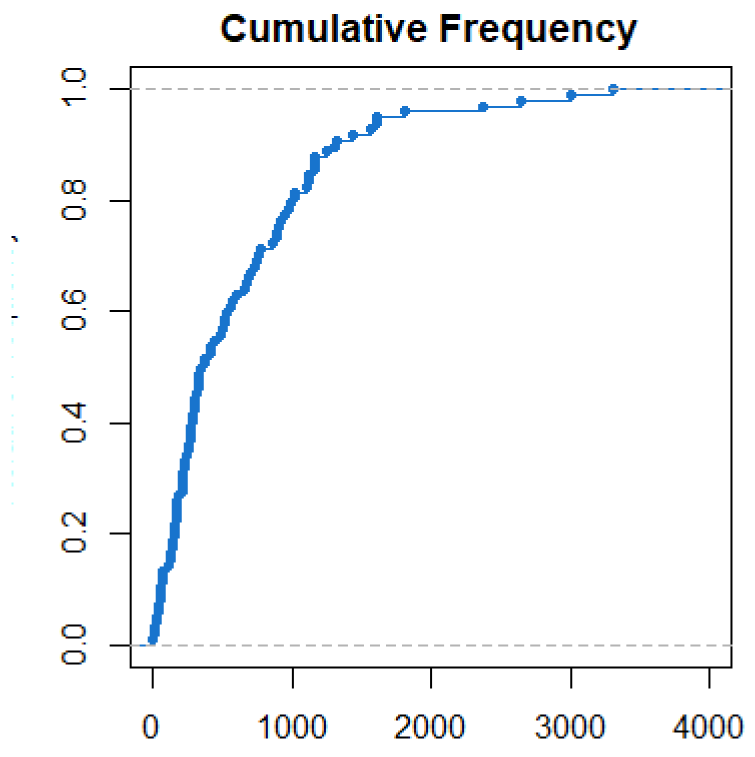
```
# Loading required package
library(terra)

# The occurrences (data.frame) are transformed into a SpatVector object
coord_occ <- terra::vect(filtered_occ, geom = c("x","y"), crs="epsg:4326")

# For calculating distances in meters, it is necessary to change the CRS
from 4326 to 3857. Keeping WGS84 will result in distances in degrees
coord_occ_dist <- terra::project(coord_occ, "EPSG:3857")

# Upload the shapefile containing the road network and convert it into a
SpatVector object
setwd("C:/chelsa")
roads <- "roads_parma.shp" %>% st_read() %>%  .$geometry %>% terra::vect()

# Here, the CRS also needs to be changed
roads <- terra::project(roads, "EPSG:3857")

# Visualization
plot(roads)
points(coord_occ_dist, col = "red")
```



```r
# With this function, the distance of each point from all roads is obtained
dist <- distance(coord_occ_dist, roads, unit ="m")

# Distance of each point from the nearest road
min_dist <- apply(dist, 1, min)
summary(min_dist)
```

With a cumulative density function, you can observe the distribution of points relative to roads: it saturates quickly

```r
# The distances are extracted and associated with the dataset of
occurrences
coord_occ$distance <- min_dist

# To create the function, distances are extracted from the dataset of
occurrences in the form of a dataframe
occ_hist <- coord_occ$distance
occ_hist <- as.data.frame(occ_hist)

# Empirical cumulative distribution function
ecdf_fun <- ecdf(occ_hist$occ_hist)

# Plot
plot(ecdf_fun, verticals = TRUE,
     main = "Cumulative Frequency",
     xlab = "Distance (m)",
     ylab = "Relative Frequency",
     xlim = c(0, 4000), pch = 20, col = "dodgerblue3")
```



Cumulative frequency

Sampling probability

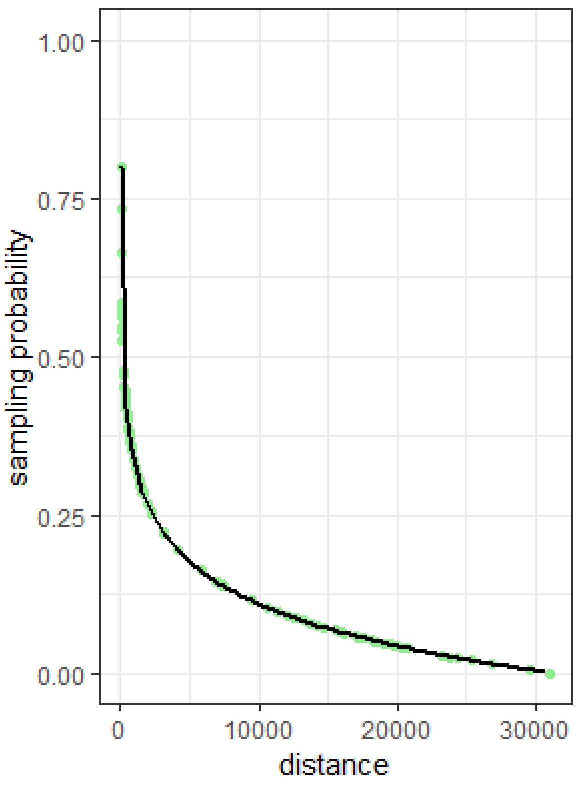
Since the probability of sampling decreases with distance from the road, we can imagine a decreasing logarithmic curve.

$$P(sampling) = 1 - \frac{log(c \cdot min\_dist)}{log(c \cdot max(min\_dist))}$$

The parameter **c**, which varies from 0 to 1, performs a transformation on the distances, modulating the slope of the curve. Values close to 0 for c decrease the distance value, also associating a good probability of finding samples even at large distances.

In this example, for simplicity, c will be set equal to 1.

```r
# Sampling probability formula
c <- 1
sampling_prob <- 1-(((log(c*min_dist))/(log(max(c*min_dist)))))

# Sampling probability curve
hist_prob <- bind_cols(sampling_prob = sampling_prob, distance = min_dist)
%>%
  ggplot(aes(x = distance, y = sampling_prob)) +
  ylim(0, 1) +
  geom_point(color = "lightgreen") +
  geom_smooth(formula = y ~ log(x), method = "lm", color = "black") +
  labs(x = "distance", y = "sampling probability") +
  theme_bw()

plot(hist_prob)
```

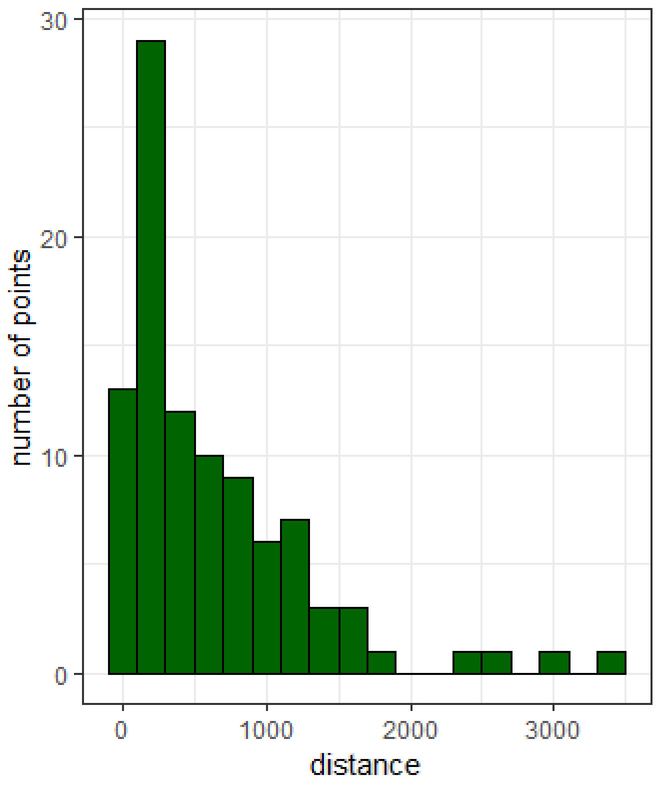


Sampling probability in relation to distance from roads

However it is constructed, the curve implies that not all points will be sampled equally: points close to the roads are sampled more, while as one moves away, with the decrease in probability, the actual number of points collected also decreases.

Let's first examine how the points are distributed with respect to distance.

```r
summary(min_dist)
#     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#    1.899  176.794  365.086  619.756  899.481 3307.472
# Istogram with number of points per distance. The 'binwidth' parameter
controls the width of the bars: each bar represents a distance of 200
meters
coord_occ %>% as.data.frame() %>% ggplot(., aes(x = distance)) +
  geom_histogram(binwidth = 200, fill = "darkgreen", color = "black") +
  labs(x = "distance", y = "number of points") +
  theme_bw()
```



Points per distance

It is useful to establish distance classes, dividing distances into intervals: each class will have its own number of points, its average sampling probability, and the quantity of points obtained when considering the roadside bias.

```
# Based on statistics, the intervals' limits are established
lim <- c(0, 200, 500, 800, 1000, 2000, 4000)

# Each point is assigned to its respective class
distance_class <- bind_cols(sampling_prob = sampling_prob,  coord_occ %>%
as.data.frame(geom="XY"))   %>%
  add_column(class = cut(.$distance, breaks = lim, labels = F))


# Points per class
count(distance_class, class)
#  class  n
#      1  26
#      2  28
#      3  15
#      4   8
#      5  16
#      6   4

# By grouping into classes, for each class, you obtain the initial number
of points, the average sampling probability, and the points actually
sampled according to this probability
groups <- distance_class %>%
  group_by(class) %>%
  summarise(plotn = n(), meanprob = mean(sampling_prob)) %>%
  add_column(plot_sampl = .$plotn * .$meanprob)

#  Union of the two dataframes through coordinates
whole_data <- merge(distance_class, groups, by="class")
```

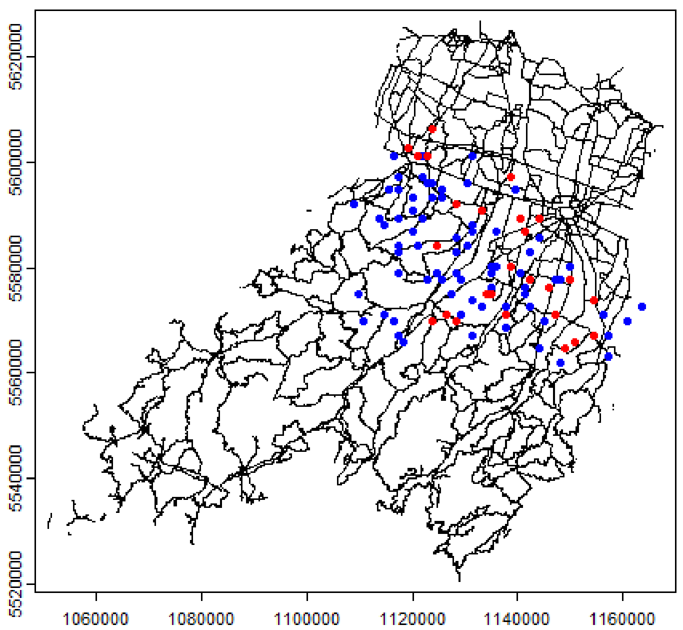
As output, a dataframe is obtained where each point is associated with:

- Bioclimatic variables
- Sampling probability
- Distance from the road
- Distance class it belongs to
- Average sampling probability of that class
- Number of points falling into that class
- Actual points obtained according to the sampling probability

Dataset resampled based on probability

Resampling the original dataset through probability, which acts as a filter, results in a reduced dataset.

```
# Points filtered by probability
by_prob <- whole_data %>%
  group_by(class) %>%
  group_split() %>%
  map(function(z){
    set.seed(1234)
    z %>%
      sample_frac(size = mean(z$sampling_prob))
  }) %>%
  do.call(bind_rows, .)

# In each group, multiplying the initial number of points by the average
sampling probability gives a smaller number of points per class
count(by_prob, class)

#   class   n
#     1    12
#     2     8
#     3     3
#     4     1
#     5     2

# In red are the original points, in blue are those actually sampled
plot(roads)
points(whole_data %>% terra::vect(., geom = c("x","y"), crs="epsg:4326")
%>%
         terra::project(., "EPSG:3857"), col="blue")
points(by_prob %>% terra::vect(., geom = c("x","y"), crs="epsg:4326") %>%
         terra::project(., "EPSG:3857"), col="red")
```



More points are sampled near the roads

Niche analysis

Since data on species distribution are often biased and incomplete, distribution maps derived directly from such data may reflect patterns of sampling efforts rather than true patterns of species distribution (*Kadmon et al., 2004*).

One possible approach to overcome this problem is to apply **ecological niche models** as a tool for mapping distribution ranges. According to this idea, the data available on the distribution of the relevant taxon are used to identify its distribution in an ecological, rather than geographical space.

Species that prefer or tolerate similar environmental preferences may co-occur in the absence of competition, when competition is stable, or when a community is not in equilibrium.

How much does resource use by species A overlap with that of species B? Recent concern over the effects of global change on species distributions has emphasized the need to quantify differences among species in their environmental requirements in a geographical context and at an extent comparable to that of species ranges (*Broennimann et al., 2011*)

The metric **D** (Schoener, 1970; reviewed in Warren et al., 2008) can be used to calculate the niche overlap between two different entities. This metric varies between 0 (no overlap) and 1 (complete overlap)

In our case, the theory and metrics that quantify niche overlap can be used to evaluate the deviation of the realized niche of the species from that derived from the subset of points actually sampled: the species is the same. The second niche will necessarily be contained within the first.

In R, this calculation is possible thanks to the ecospat package. (Di Cola et al, 2016) The aim of the ecospat package is to make available novel tools and methods to support spatial analyses and modeling of species niches and distributions in a coherent workflow. The package is written in the R language (R Development Core Team) and contains several features, unique in their implementation, that are complementary to other existing R packages. Pre-modeling analyses include species niche overlap, quantifications and comparisons between distinct ranges or time periods, measures of phylogenetic diversity, and other data exploration functionalities.

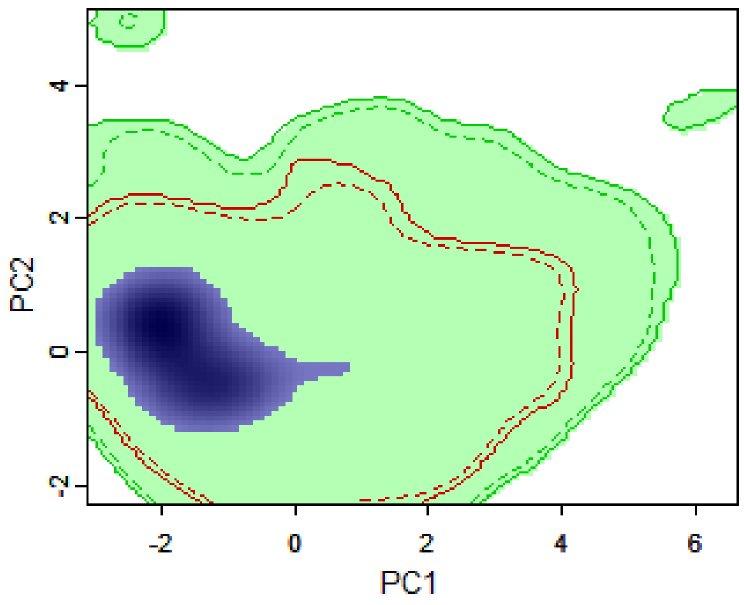
Let's take the dataset containing the coordinates of the points sampled according to probability. For instance, let's choose to visualize the ecological niche generated by the points belonging to distance class 3, from 500 meters to 800 meters: how much does the niche of this subset differ from the realized niche?

```r
# Filtering points of class 2
points_prob <- by_prob %>% filter(class==2) %>% terra::vect(., geom =
c("x","y"), crs="epsg:4326")

# Number of points
nrow(points_prob)
# 53

# Extract the coordinates of the points belonging to the chosen class
coordinate_prob <- points_prob %>%  st_as_sf() %>% sf::st_coordinates()
coordinate_prob

# Create a dataframe
coordinate_prob <- data.frame(x = coordinate_prob[, "X"], y =
coordinate_prob[, "Y"])

# Let's merge the two dataframes (original occurrences and points belonging
to the chosen class) based on the 'x' and 'y' columns
new_points <- merge(filtered_occ, coordinate_prob, by = c("x", "y"))

# From dataframe to raster
raster_occ <- new_points %>% .[,-(8:9)] %>% rasterFromXYZ()
```

At this point, there are occurrences and the climatic variables associated with them: or the niche quantification, a matrix with the background environmental variables from both ranges, as well as the global environment are needed.
Afterwards, a Principal Component Analysis (PCA) of the environmental data is carried out.

```r
# Loading required packages
library(ecospat)
library(ade4)

# With the getValues function, bioclimatic data is obtained in the form of
a dataframe: the function is applied to both the realized niche and the
subset
env_pa <- getValues(raster_pa)
env_occ <- getValues(raster_occ)

# Remove missing values
env_occ <- env_occ[complete.cases(env_occ), ]
env_pa <- env_pa[complete.cases(env_pa), ]

# Produce global environmental background data
globalEnvM <- rbind(env_pa, env_occ)

# PCA on the global data
pca.clim <- dudi.pca(globalEnvM, center = TRUE,
                     scale = TRUE, scannf = FALSE, nf = 2)

# Two-dimensional summary of the total environmental variability
global.scores <- pca.clim$li
```

```
# The observation data are mapped into that two-dimensional space using the
suprow function
pa.scores <-
  suprow(pca.clim,
         data.frame(filtered_pa)[, colnames(globalEnvM)])$li

occ.scores <-
  suprow(pca.clim,
         data.frame(filtered_occ)[, colnames(globalEnvM)])$li

pa.scores1 <- suprow(pca.clim, env_pa)$li
occ.scores1<- suprow(pca.clim, env_occ)$li


# Calculate the Occurrence Density Grid
pagrid <- ecospat.grid.clim.dyn(global.scores,
                                pa.scores,
                                pa.scores1)

occgrid <- ecospat.grid.clim.dyn(global.scores,
                                 occ.scores,
                                 occ.scores1)

# Plot niche category
ecospat.plot.niche.dyn(pagrid, occgrid, quant = 0.1, interest = 2,
name.axis1 = "PC1", name.axis2 = "PC2")

# Calculate niche overlap
ecospat.niche.overlap(pagrid, occgrid, cor=T)
# $D
# [1] 0.1171442

# $I
# [1] 0.33873
```



The green niche is the realized one, while the blue one comes from the subset

Cube

By adding a column containing time, specifically a random year within the climatic interval (1980-2010), we obtain a data cube: species, x, y, distance. This last factor, as we have seen, influences the sampling.

So far, the code only generates one species: to obtain a dataframe containing a chosen number of random species, it is necessary to iterate the process as follows

```
# Iteration number
num_iter <- 3

# Empty lists
dataframe_list_before <- list()
dataframe_list_after <- list()
```

```
# For loop to generate species and calculate distances for each iteration
for (i in 1:num_iter) {
  set.seed(i)
  random.sp <- generateRandomSp(raster.stack = mydata,
                               convert.to.PA = FALSE,
                               species.type = "multiplicative",
                               approach = "random",
                               relations = "gaussian",
                               realistic.sp = TRUE,
                               plot = FALSE)
  new.pres <-convertToPA(random.sp,
                        beta = "random",
                        alpha = -0.05, plot = FALSE,
                        species.prevalence = 0.1)
  presence.points <- sampleOccurrences(new.pres,
                                      n = 100,
                                      type = "presence only",
                                      sample.prevalence = 0.9,
                                      error.probability = 0,
                                      detection.probability = 1,
                                      correct.by.suitability = TRUE,
                                      plot = FALSE)
  raster01 <- new.pres$pa.raster %>% raster()
  r1 <- mydata$mean.annual.T
  r2 <- mydata$annual.range.air.T
  r3 <- mydata$annual.precip
  r4 <- mydata$amount.of.prec..wettest.month
  r5 <- mydata$amount.prec..driest.month
  stack_pa <- brick(r1, r2, r3, r4, r5, raster01)
  values <- stack_pa %>% rasterToPoints() %>% as.data.frame()
  filtered_pa <- values %>% filter(., lyr.1 == 1) %>% as.data.frame()
  raster_pa <- filtered_pa %>% .[,-8] %>% rasterFromXYZ()
  raster_occurences <- presence.points$sample.points %>% as.data.frame()
```

```r
%>% .[.$Real == 1 & .$Observed == 1, ]
  stack_occ <- brick(r1, r2, r3, r4, r5)
  values_occ <- stack_occ %>%  rasterToPoints() %>% as.data.frame()
  filtered_occ <- merge(values_occ, raster_occurences, by = c("x", "y"))
  coord_occ <- terra::vect(filtered_occ, geom = c("x","y"),
crs="epsg:4326")
  coord_occ_dist <- terra::project(coord_occ, "EPSG:3857")
  setwd("C:/chelsa")
  roads <- "strade_parma.shp" %>% st_read() %>%  .$geometry %>%
terra::vect()
  roads <- terra::project(roads, "EPSG:3857")
  dist <- distance(coord_occ_dist, roads, unit ="m")
  min_dist <- apply(dist, 1, min)
  coord_occ$distance <- min_dist
  c <- 1
  sampling_prob <- 1-(((log(c*min_dist))/(log(max(c*min_dist)))))
  lim <- c(0, 500, 1000, 2000, 5000, 10000, 15000)
  distance_class <- bind_cols(sampling_prob = sampling_prob,  coord_occ %>%
as.data.frame(geom="XY"))   %>%
    add_column(class = cut(.$distance, breaks = lim, labels = F))
  groups <- distance_class %>%
    group_by(class) %>%
    summarise(plotn = n(), meanprob = mean(sampling_prob)) %>%
    add_column(plot_sampl = .$plotn * .$meanprob)
  whole_data <- merge(distance_class, groups, by="class")
  set.seed(i)
  whole_data$time <- sample(1980:2010, nrow(whole_data), replace = TRUE)
  by_prob <- whole_data %>%
    group_by(class) %>%
    group_split() %>%
    map(function(z){
      set.seed(1234)
      z %>%
        sample_frac(size = mean(z$sampling_prob))
    }) %>%
    do.call(bind_rows, .)
  coord_occ <- as.data.frame(whole_data)
  dataframe_before <- whole_data[,-(1:9)]
  dataframe_before <- dataframe_before[,-(4:6)]
  by_prob <- as.data.frame(by_prob)
  dataframe_after <- by_prob[,-(1:9)]
  dataframe_after <- dataframe_after[,-(4:6)]
  dataframe_before$specie <- paste("specie", i)
  dataframe_after$specie <- paste("specie", i)
  dataframe_list_before[[i]] <- dataframe_before
  dataframe_list_after[[i]] <- dataframe_after

}
```

```r
# Combine all the dataframes obtained into a single dataframe
dataframe_before <- do.call(rbind, dataframe_list_before)
dataframe_after <- do.call(rbind, dataframe_list_after)
dataframe_before
dataframe_before <- dataframe_before[, c("specie", "x", "y", "time",
"distance")]
dataframe_after <- dataframe_after[, c("specie", "x", "y", "time",
"distance")]

setwd("C:/chelsa")
# Save the cube!
write.csv(dataframe_before, file = "cube_before.csv", row.names = FALSE)
write.csv(dataframe_after, file = "cube_after.csv", row.names = FALSE)
```