The sdm R package for species distribution modelling

Babak Naimi, Elham Ebrahimi, Miguel B. Araujo

2024-06-27

Introduction

Global biodiversity change is one of the most pressing environmental issues of our time (Dornelas et al. 2023) as the planet is facing the largest biodiversity loss, known as the sixth mass extinction (Kolbert 2014), with the potential loss of half the species on the planet (Hughes 2023) and the third most important risk threatening humanity over the next 10 years (World Economic Forum, 2022). In response to such threats, global environmental authorities and the Convention on Biological Diversity (CBD) in the fifteenth meeting of the Conference of the Parties (COP15) launched a global commitment, the Kunming-Montreal Global Biodiversity Framework (KM-GBF), sets ambitious goals and targets to halt and reverse biodiversity loss by 2030 and 2050, aiming to restore ecosystems, conserve species, and ensure sustainable use of natural resources.

Quantitative analyses and sophisticated modelling tools play a significant role to inform biodiversity conservation strategies and policy decisions. Species Distribution Models (SDMs), the most common and widely used approach for biodiversity modelling (Araújo et al. 2019), have emerged as pivotal tools in biodiversity studies, offering robust methods to predict the geographic distribution of species across landscapes. These models, also known as bioclimatic envelope models, ecological niche models and habitat suitability models, explore the relationships between species observations (usually in the form of presence-only, presence-absence, or abundance) and environmental variables to provide insights into the potential distribution of species across geographic space and time (Ceballos, Ehrlich, and Dirzo 2017; Dirzo et al. 2014; Guisan and Thuiller 2005; Guisan and Zimmermann 2000). The application of SDMs spans a wide range of tasks, such as forecasting the effects of climate change on biodiversity, identifying high biodiversity value areas, selecting sites for conservation, habitat restoration, and species translocation, as well as quantifying the effects of anthropogenic factors on biodiversity, among many other applications.

Developing SDMs involves a systematic workflow that integrates ecological data, statistical and/or machine learning methods, and computational tools to predict species distributions. The sdm R package (Naimi and Araújo 2016) provides an extensible and comprehensive platform for developing SDMs using different types of species data including presence-absence, presence-only, and abundance data. The package facilitates implementing the entire species distribution modelling workflow including data integration, modelling and evaluating their performance using multiple accuracy metrics, and using the models for certain applications (Fig. 2). This package supports a variety of modelling algorithms, including machine learning techniques like Random Forest, Boosted Regression Trees (BRT), and MaxEnt, as well as classical statistical methods such as Generalized Linear Models (GLM) and Generalized Additive Models (GAM). By offering a flexible and user-friendly interface, the sdm R package supports multiple common formats of spatial and temporal data, facilitates developing SDMs by simultaneously incorporating multiple algorithms and for multiple species, the comparison of multiple models and generating their ensembles, and provides tools for interpretation of models, all enhancing the robustness and accuracy of predictions.

This book chapter demonstrates the entire workflow of species distribution modelling and practical steps using the sdm R package. We provide a comprehensive guide, from data preparation and selection of environmental variables to model building, evaluation, and prediction. Emphasizing practical application,

this chapter walks through the utilization of various modeling algorithms supported by the sdm package, including techniques for optimizing model performance and interpreting results. By illustrating the usage of the package for a real-world example as a case study, we highlight the versatility and effectiveness of SDMs in biodiversity research and conservation planning.

An overview of the SDM workflow in the sdm R package:

The 'sdm' R package is a comprehensive framework that provides a set of user-friendly functions for implementing the entire species distribution modelling workflow to develop data-driven SDMs, begins with providing data for both species (response or dependent variable) and environment (predictor or explanatory variables; also called covariates). The workflow consists of three major steps to build SDMs including data preparation, modelling, and prediction. The sdm package offers a user-friendly function for each step in the workflow with arguments controlling the tasks given a user's inputs.

(a) *Data preparation*: involves controlling and integrating species and environmental data, identifying and cleaning issues with data such as mismatch in coordinate reference systems, duplication of records, missing values, and testing collinearity among predictor variables. Different types of species data are supported in the sdm package including **presence-only**, **presence-absence**, and **abundance** data. Since presence-only is the most widely used data type, a case study in this book chapter is demonstrated based on using this data type. In case of presence-only, pseudo-absence or background records can be generated at this stage. The main function for this stage in the sdm package is `sdmData` which supports both tabular (e.g., `data.frame`) and spatial data formats, and can do all the above-mentioned tasks for cleaning and preparing data.

(b) *Modelling*: utilizes various (20+) statistical and machine learning algorithms to construct SDMs based on the prepared clean data in the previous step (a). Once the models are built (also called training of the models), they undergo rigorous evaluation to assess their performance and predictive accuracy. This step often includes resampling procedure for dividing data into training and test partitions using multiple methods such as sub-sampling, cross-validation, and bootstrapping (Naimi and Araújo, 2016), which employ widely used evaluation metrics, like the Area Under the Curve (AUC) and True Skill Statistic (TSS), to ensure robust model validation (Allouche et al., 2006; A H Fielding and Bell, 1997). A single function, sdm, in the package integrates all modelling and evaluation tasks defined in this phase that can be simultaneously applied for multiple species, multiple modelling methods, and multiple replications of data.

(c) *Prediction*: After a successful evaluation, the models are used for predicting (or projecting or hindcasting) species distributions across geographic areas and time periods under current and future environmental scenarios. The predict function in the sdm package can generate the predicted values (e.g., probability of occurrence) based on each trained model given the predictor variables as inputs. When multiple modelling methods are employed, the predictions of individual models can be combined into a single consensus prediction using ensemble prediction approach (Araújo and New, 2007). The ensemble function in the sdm package offers several methods to combine individual models such as weighted and unweighted mean, median, mean of predicted presence-absence, two step mean, etc. (check the help page of the function in the sdm package for more information).

(d) *Interpretation*: Finally, the workflow may include post-processing steps for model interpretation and visualization. This can involve mapping predicted distributions, analyzing variable importance, and generating graphical representations of model outputs which facilitate understanding and communication of results to address specific applied questions. The sdm R package streamlines these processes, providing researchers and practitioners with powerful tools to develop, evaluate, and apply SDMs effectively to inform various applications, such as identifying priority areas for conservation, assessing the impacts of climate change, and guiding habitat restoration efforts.

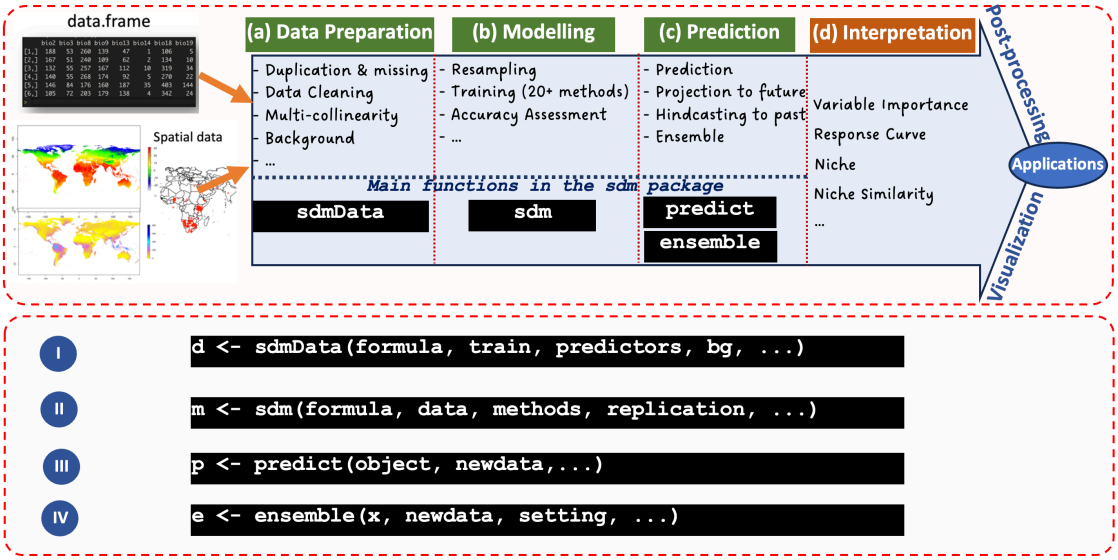


Figure 1: A schematic representation of the species distribution modeling workflow with the main functions, followed by their basic usages, in the *'sdm'* R package.

Case study: Species Distribution Modelling of *Snow leopard*

To demonstrate the SDM workflow, this section is dedicated to assess and explore **impacts of climate change on geographical distribution of Snow leopard** (scientific name: *Panthera uncia*). We use the sdm R package, along with some other packages, to conduct the study, and here, a step by step guideline is provided to discuss the practical solution for implementing the workflow and generating results in the forms of maps, tables, and graphs.

Figure 2: Snow leopard (Panthera uncia).

The solution to conduct the study has been provided following the main steps in the workflow defined in Figure 2. The solution is provided through the following sections:

(i) Downloading data: Here, we use R and the geodata package to download species data from GBIF, and climate data (for both the current and future times) from Worldclim.

(ii) Preparing data: Data are checked for some issues, and get ready as spatial datasets for conducting SDMs.

(iii) Developing models using the sdm R package: The main workflow in the sdm package (Naimi and Araújo 2016) to develop and evaluate SDMs are demonstrated.

(iv) Predict/Project the map of habitat suitability in current and future times: We predict the current

distribution of the species (habitat suitability), and project its distribution into a future climatic scenario (for the year 2100).

(v) Assess range shift in response to climate change: Given the habitat suitability maps in both the current and future times, we can assess the magnitude and distribution of changes, and quantitatively measure range shifts due to climate change.

(i) Downloading Species and Climate Data

In this section, we will use R to download occurrence data for the Snow Leopard (*Panthera uncia*) from the Global Biodiversity Information Facility (GBIF) and climate data from the WorldClim database. These datasets will serve as the foundation for developing species distribution models and assessing the potential impacts of climate change on the Snow Leopard's habitat.

Species Occurrence Data: We will use the sp_occurrence function in the geodata package to download Snow Leopard occurrences from GBIF. Occurrences are the locations where the species has been observed (mostly cover presence-only records), providing spatial points essential for building SDMs. Some descriptions are provided as comments within the code:

```
library(geodata)
```

```
## Loading required package: terra
```

```
## terra 1.7.78
```

```
# let's first check whether any records available on GBIF without downloading:

sp_occurrence("Panthera","uncia", download = F)
```

```
## Loading required namespace: jsonlite
```

```
## [1] 196
```

```
# we have 196 records available, let's download them as a data.frame:

sp <-  sp_occurrence("Panthera","uncia")
```

```
## 196 records found
```

```
## 0-196
## 196 records downloaded
```

```
#-------
# the sp data.frame has over 100 columns, corresponding to different information
# Some of the columns can be used to filter and exclude some records.
# For example, the column "basisOfRecord" is a Darwin Core term that refers to
# the specific nature of the record and can refer to one of 6 classes

# Let's check our records to see their frequency for different classes:

table(sp$basisOfRecord)
```

```
##
##  HUMAN_OBSERVATION  MATERIAL_CITATION    MATERIAL_SAMPLE PRESERVED_SPECIMEN
##                115                   8                   4                  69
```

```r
# You may find the definition from the GBIF website:
# https://docs.gbif.org/course-data-use/en/basis-of-record.html

#------
# You can also check the years of records in our dataset:

table(sp$year)
```

```
##
## 1899 1979 1998 2000 2004 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017
##    1    1    1    1    2    2    2    1    1    9    6   17    9   14   15   18
## 2018 2019 2020 2021 2022 2023 2024
##   16   25    6    5    6   23   11
```

```r
# let's also check occurrenceStatuse column:

table(sp$occurrenceStatus) # we are sure now all records are presence!
```

```
##
## PRESENT
##     196
```

```r
# if it was not all "PRESENT", we could keep only "PRESENT" to use data as
# a presence-only dataset
#---------------------------
###### Filtering some records:

# Given the information avaialble in "basisOfRecords" and "year" columns, we
# want to filter data and keep appropriate records. We may use other columns
# like coordinateUncertaintyInMeters; or we may use some specific packages
# designed to clean GBIF data (e.g., CoordinateCleaner). But here, we do our
# filtering based on the "basisOfRecords" and "year" columns.

# Based on the information we checked above, we keep records with
# basisOfRecord == "HUMAN_OBSERVATIO" & year > 1970, we use the which function:

w <- which(sp$basisOfRecord == "HUMAN_OBSERVATION" & sp$year > 1970)

# w keeps the record number (row number in sp data.frame) that we need to keep

sp <- sp[w,] # only keep records specified in w

#------
## Get rid of all columns except coordinate columns (lon & lat)
# from all columns, we only keep 2 columns (lon, lat)

sp <- sp[,c('lon','lat')]

# we add another column (its name can simply be "species") to which we assign
# 1 to all records in the column as they are presence-only:

sp$species <- 1

# let's check the head of the table (data.frame) to make sure it is fine:
```

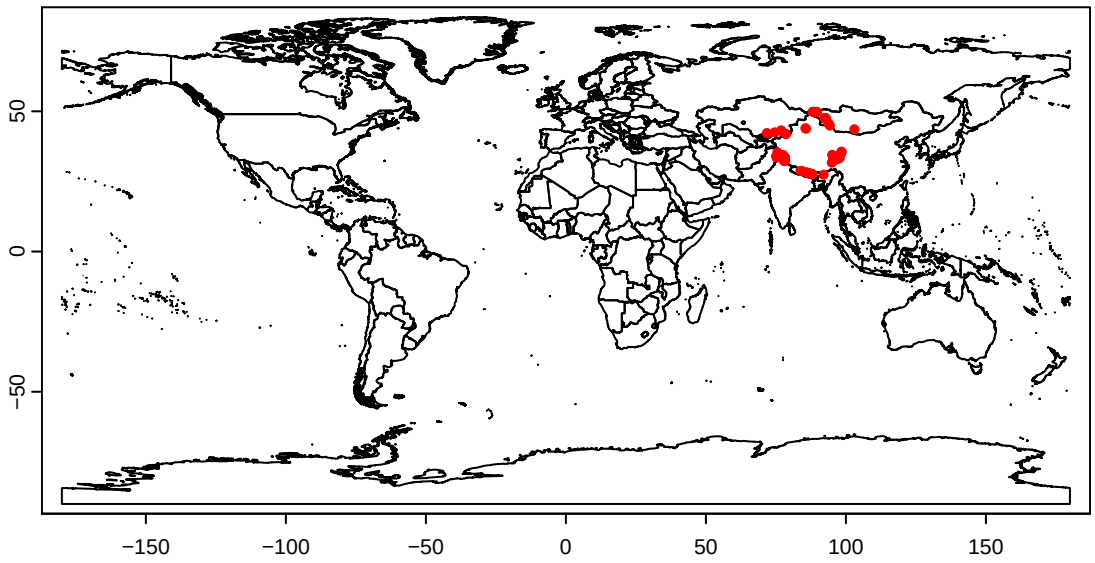
```r
head(sp)
```

```
##        lon      lat species
## 1 74.60183 42.50146       1
## 2 98.36646 35.58493       1
## 3 77.90942 32.34409       1
## 4 98.46910 35.55448       1
## 5 77.62451 33.91132       1
## 6 76.86968 34.31075       1
```

```r
# now, we convert the sp data.frame to Spatial Points using the vect function in
# the terra package (generates SpatVector object):

sp <- vect(sp, c('lon','lat'))

sp # sp, now, is a SpatVector object contining spatial points of species records
```

```
##  class       : SpatVector
##  geometry    : points
##  dimensions  : 115, 1  (geometries, attributes)
##  extent      : 71.76065, 103.0307, 27.41172, 49.8562  (xmin, xmax, ymin, ymax)
##  coord. ref. :
##  names       : species
##  type        :   <num>
##  values      :       1
##                      1
##                      1
```

```r
# let's generate a map to see the locations of the species records:

# We had the world map but if you don't, you can find it on Internet!
wld <- vect('world.shp')


plot(wld)

points(sp,col='red')
```



Downloading Climate Data:

We are going to use climate data with 19 bioclimatic variables (represent different aspects of climate condition including mean annual temperature and precipitation, their seasonality and extreme conditions) available in the WorldClim dataset. The bioclim variables are extracted from monthly weather data (monthly minimum and maximum temperature and mean precipitation). The variables are described here: https://www.worldclim.org/data/bioclim.html

Given that the purpose of the instructions provided here is to demonstrate how the SDM workflow can be implemented, we will use a coarse resolution (10 minutes; approximately 20 km) dataset to make the computation fast. However, in practice, the resolution should be selected based on the scale of the study to ensure accurate results. The WorldClim dataset is available on the worldclim.org website.

We can download the dataset for both current and future time periods. The future climate conditions are estimated using General Circulation Models (GCMs), which simulate the Earth's climate based on various physical processes and interactions within the atmosphere, ocean, and land surface. These models consider different scenarios known as Shared Socio-economic Pathways (SSPs), which reflect various potential future conditions based on socio-economic trends and policies. SSPs provide a comprehensive framework to analyze the impacts of different socio-economic developments on climate change and to explore mitigation and adaptation strategies.

The climate data for the current time (baseline), represent mean climate condition generated from weather data recorded between 1980-2000. To download the climate data for future times, we need to select the appropriate GCMs (we may select several or even all GCMs for a real study) and SSP scenarios that align with the objectives of our study. For example, we might choose SSP2-4.5, which represents a moderate scenario where socio-economic trends lead to stabilization in global emissions by mid-century. Alternatively, we could select SSP5-8.5, which represents a high-emission scenario. Under SSP5-8.5, greenhouse gas emissions continue to rise throughout the century, leading to significant increases in temperature and more extreme climate conditions (also called business as usual or worst-case scenario). This scenario is useful for understanding the potential impacts of unmitigated climate change and helps in planning for worst-case outcomes.

For this exercise, let's download climate data for both the current time (baseline) and future time (year 2100; it is generated based on 20 years of data between 2081-2100) for one of GCMs and SSP5-8.5. We can use the worldclim_global function to download climate data for the current time, and cmip6_world to download for the future. Check the help page of the functions for more details about the usage of these functions:

```r
# the following function, download bioclim layers with the resolution of 10 min.

#########################################################
##-----> Downloadin climate for the Baseline (current time):
#########################################################

bio <- worldclim_global(var="bio",res=10, path = getwd())

bio
```

```
## class       : SpatRaster
## dimensions  : 1080, 2160, 19  (nrow, ncol, nlyr)
## resolution  : 0.1666667, 0.1666667  (x, y)
## extent      : -180, 180, -90, 90  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## sources     : wc2.1_10m_bio_1.tif
##               wc2.1_10m_bio_2.tif
##               wc2.1_10m_bio_3.tif
##               ... and 16 more source(s)
## names     : wc2.1~bio_1, wc2.1~bio_2, wc2.1~bio_3, wc2.1~bio_4, wc2.1~bio_5, wc2.1~bio_6,
## min values   :    -54.72435,      1.00000,     9.131122,         0.000,    -
29.68600,   -72.50025, ...
## max values :   30.98764,   21.14754, 100.000000,   2363.846,   48.08275,   26.30000, ...
```

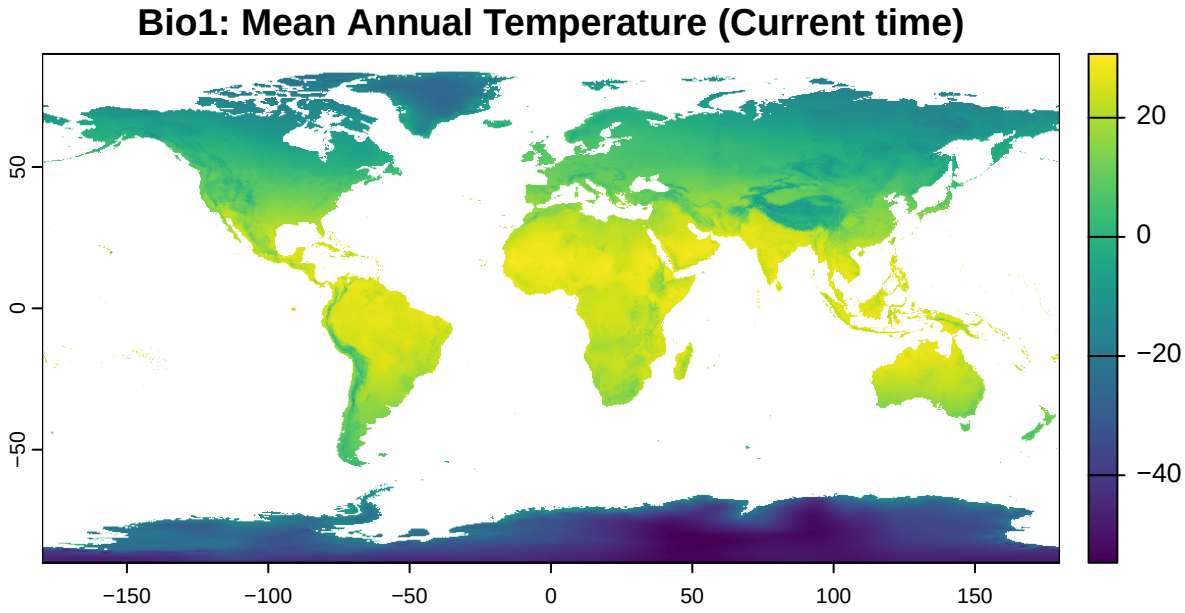
```r
# let's check the names of layers in the bio object
names(bio)
```

```
##  [1] "wc2.1_10m_bio_1"  "wc2.1_10m_bio_2"  "wc2.1_10m_bio_3"  "wc2.1_10m_bio_4"
##  [5] "wc2.1_10m_bio_5"  "wc2.1_10m_bio_6"  "wc2.1_10m_bio_7"  "wc2.1_10m_bio_8"
##  [9] "wc2.1_10m_bio_9"  "wc2.1_10m_bio_10" "wc2.1_10m_bio_11" "wc2.1_10m_bio_12"
## [13] "wc2.1_10m_bio_13" "wc2.1_10m_bio_14" "wc2.1_10m_bio_15" "wc2.1_10m_bio_16"
## [17] "wc2.1_10m_bio_17" "wc2.1_10m_bio_18" "wc2.1_10m_bio_19"
```

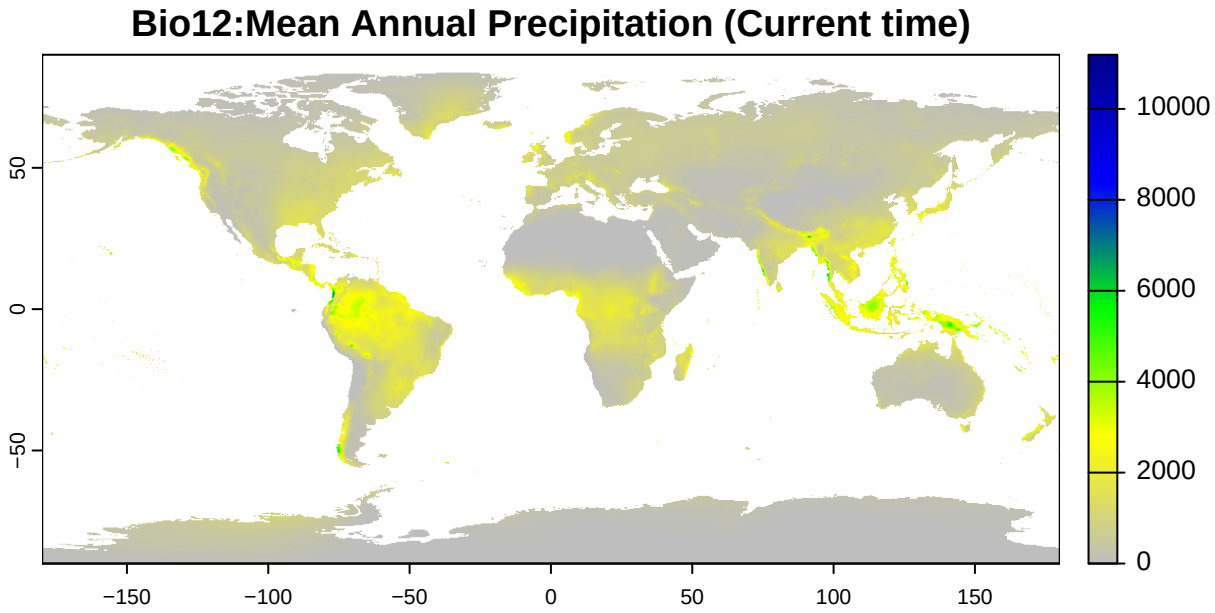
```r
# As you can see, the bio object contains 19 bioclim variables representing
# climate conditions for the current time. The names are long, have information
# about the version of the WorldClim dataset (WC_2.1) and resolution.

# Let's make the names simpler (i.e., bio1, bio2, ..., bio19)

# We can simply assign the new names:

names(bio) <- paste0('bio',1:19) # Do you know why we used the paste0 function?

# now, let's check again the names of our data object:
names(bio)
```

```
##  [1] "bio1"  "bio2"  "bio3"  "bio4"  "bio5"  "bio6"  "bio7"  "bio8"  "bio9"
## [10] "bio10" "bio11" "bio12" "bio13" "bio14" "bio15" "bio16" "bio17" "bio18"
## [19] "bio19"
```

```r
# Let's visualise the first layer (Mean Annual Temperature):


plot(bio[[1]],main='Bio1: Mean Annual Temperature (Current time)')
```




```
# and 12th layer (Mean Annual Precipitation)

# for this one, let's define an appropriate color palette:

cl <- colorRampPalette(c('gray','yellow','green','blue','darkblue'))


plot(bio[[12]],col=cl(200),main='Bio12:Mean Annual Precipitation (Current time)')
```




```
############################################################
#----> Downloading bioclim variables for the Future time:
############################################################

# To download bioclim for a future time, we need the following information:
```

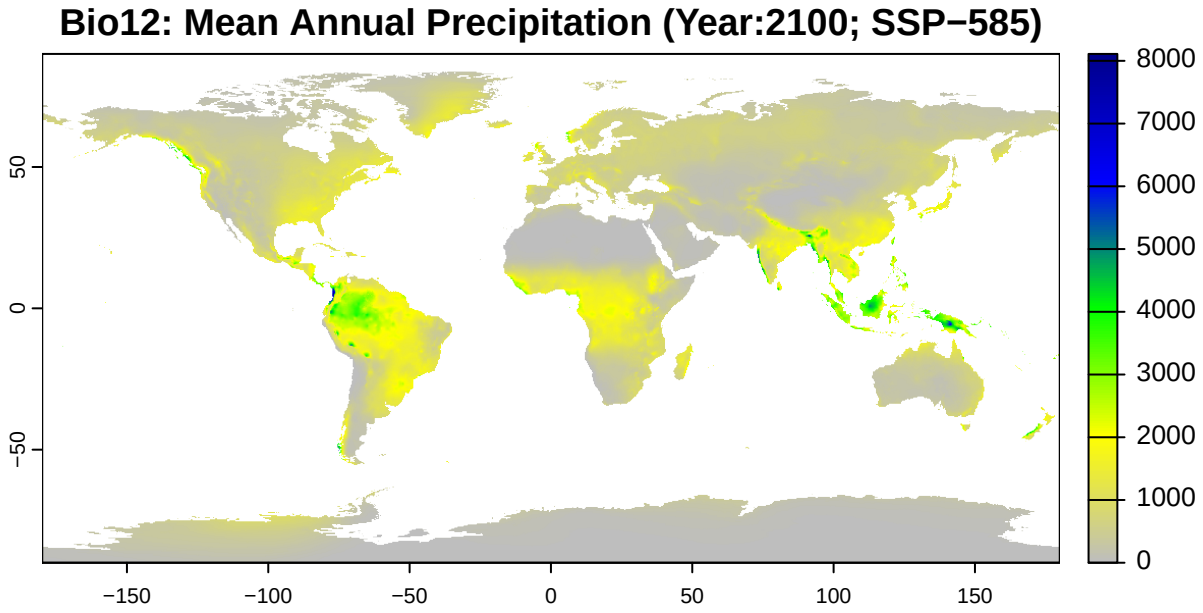
```
## GCM: Which Climate Model? (20+ GCMs are available!, here we chose: "MIROC6")

## SSP: Which Climate Scenario? (4 different scenarios are available!)
#-------> ssp: 126, 245, 370, 585 (from "optimistic"....to.... "pessimistic")

## Year: Which future time (e.g., "2081-2100")

# we can also specify variables (here, we need bioclim; "bioc"),

## res: resolution (10 arc min ~ equivalent to almost 20 km)
#-----------
# we can use cmip6_world function in the geodata package to download climate
# variables for the future times:

biof <- cmip6_world(model="MIROC6", ssp='585',
                    time = "2081-2100", var = 'bioc', res=10, path=getwd())

biof
```

```
## class       : SpatRaster
## dimensions  : 1080, 2160, 19  (nrow, ncol, nlyr)
## resolution  : 0.1666667, 0.1666667  (x, y)
## extent      : -180, 180, -90, 90  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## source      : wc2.1_10m_bioc_MIROC6_ssp585_2081-2100.tif
## names       : bio01, bio02, bio03,  bio04, bio05, bio06, ...
## min values  : -50.3,  -1.9, -10.6,   13.4,   -26, -68.4, ...
## max values  :  36.3,  22.5,  95.4, 2250.5,    55,  27.8, ...
```

```
# let's check the names of the biof object which contains bioclim variables of
# future (year: 2100) for the worst-case scenario (ssp585):

names(biof)
```

```
##  [1] "bio01" "bio02" "bio03" "bio04" "bio05" "bio06" "bio07" "bio08" "bio09"
## [10] "bio10" "bio11" "bio12" "bio13" "bio14" "bio15" "bio16" "bio17" "bio18"
## [19] "bio19"
```

```
# For our modelling practice, the names of bioclim layers in the current and
# future time should be exactly the same, so since the order of the layers in
# bio and biof is the same, we can simply assign the names of biof to be the
# same as bio:

names(biof) <- names(bio)


# let's visualise precipitation for the future time:

plot(biof[[12]],col=cl(200),
     main="Bio12: Mean Annual Precipitation (Year:2100; SSP-585)")
```



(ii) Data preparation

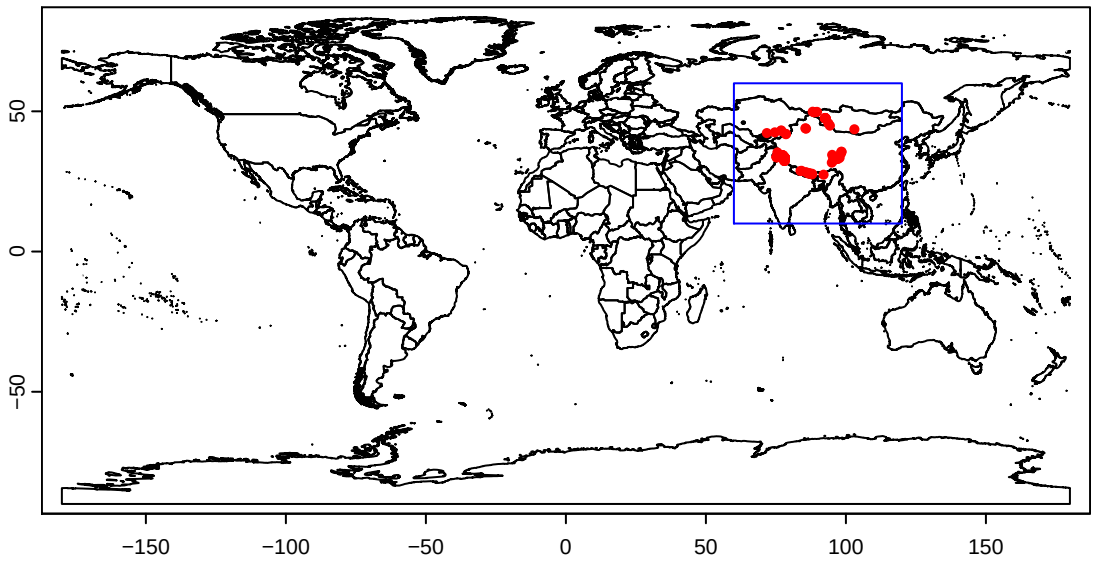
After downloading data, we need to take some steps to prepare data for modelling.

While statistical methods can be employed to identify outliers of the species data, a visual inspection of the map with species locations suggests that the points in North America may be outliers. This suspicion can be further validated by comparing these points with the species' range map provided by the IUCN. Therefore, we can exclude the outlier records. However, if the study area to train the model is selected and the outliers are located outside of the study area, the outliers will not be considered in the modelling procedure (i.e., no need to exclude them).

Selecting study area:

Although the purpose of our case study is to generate **global** potential distribution of species, we may select a realistic study area which can be an area potentially accessible to the species. Since the background records are generated within the study area, it is important to select a reasonable area to train SDMs. We may use a biogeographical map to overlap presence records and select the polygons intersected which species presence locations and conider them as the study area. Alternatively, we can approximately specify the area by choosing its upper left and lower right which give us the minimum and maximum x and y coordinates to make an extent object using the terra package.

```
library(terra)

plot(wld)

bnd <- ext(c(60,120,10,60)) # spatial extent with c(X_min, X_max, Y_min, Y_max)

points(sp,col='red')

plot(bnd,add=T,border='blue')
```



Cropping Climate Data to the extent defined as the study area:

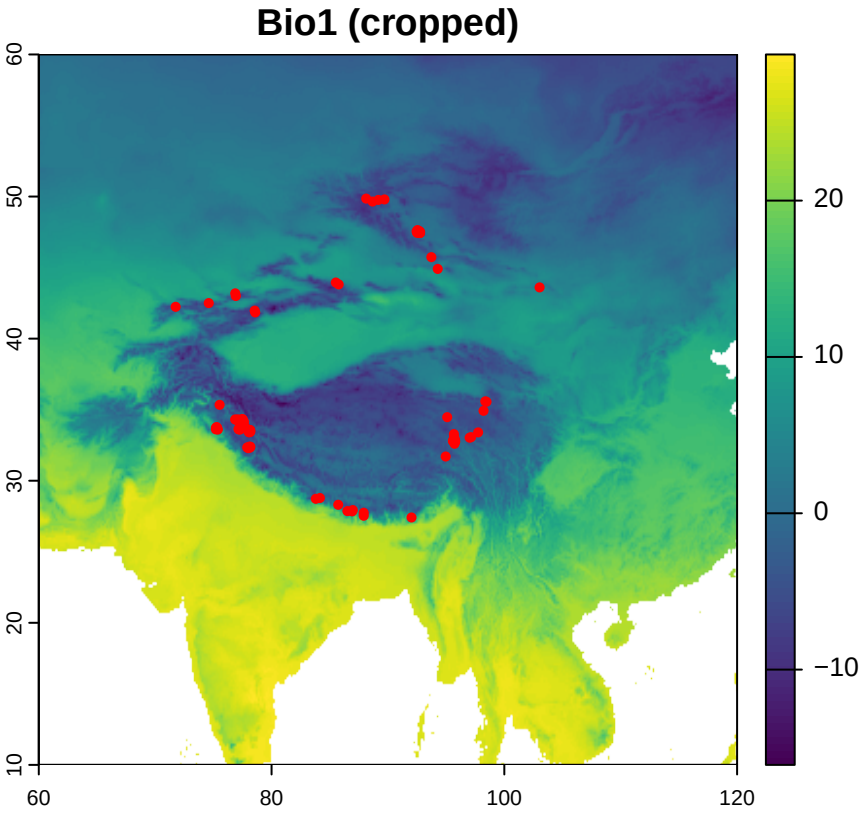
As mentioned earlier, when the species records are presence-only, we need to consider a realistic area (areas potentially accessible for the species) to calibrate (train) SDMs where the background records are drawn. Here, the specified extent (the rectangle) is used as the study area to train SDMs but we will use the models to predict/project across the world.

The following code crops the bioclim variables of the current time within the specified extent (bnd object). We don't crop the future dataset (biof) as the model training uses on the variables in the current time:

```
# the extent specified earlier:
bnd
```

```
## SpatExtent : 60, 120, 10, 60 (xmin, xmax, ymin, ymax)
```

```
bioc <- crop(bio, bnd) # crop bio object within the bnd extent

# let's visualise a layer from "bioc" and add the species occurrences to the map:

plot(bioc[[1]],main='Bio1 (cropped)')

points(sp,col='red')
```



Multi-collinearity:

When we have two or more numerical predictor variables that are strongly correlated, the data are subjected to the issue of multi-collinearity (also called collinearity) (Dormann et al. 2013). Some modelling methods, such as GLMs, may be sensitive to the collinearity issue as their parameterisation can be affected. Even if a modelling method is not sensitive, excluding the collinear variables is a good practice given the parsimony rule to avoid redundant information in the model which makes it un-necessarily complex.

Several approaches are available to deal with collinearity issue. One approach is to transform the environmental variables into a new set of variables using Principle Component Analysis (PCA), an ordination method for reducing data dimensions and generate new, uncorrelated variables. This can be done using the `pca` function, or by specifying the `pca` term in the formula in the `sdmData` function.

Another approach is to identify and exclude variables that show signs of collinearity. To identify collinear variables, a simple pairwise correlation test can be conducted between each pair of variables (a correlation greater than 0.7 is indicative of collinearity). Alternatively, the variance inflation factor (VIF) can be measured for each variable, representing how much of a variable's variance can be explained by other variables (a VIF greater than 10 is a sign of collinearity).

A popular hybrid approach was introduced by Naimi et al. (2014) to measure VIF or combination of VIF and correlation coefficient through a stepwise procedure (Naimi et al. 2014). Two functions implemented in the usdm package can be used here including `vifstep`, and `vifcor`. To deal with collinearity, one of these two methods should be used. `vifstep` checks the VIF metric for all numerical predictor variables and identifies the variable with the maximum VIF for exclusion if it exceeds the specified threshold (default=10). This procedure is repeated step by step until all the remained variables have VIF values below the threshold.

Alternatively, `vifcor` checks the correlation coefficient for all possible pairs of variables, and identifies the pair with the maximum correlation. If it exceeds the specified threshold (e.g., 0.7), one of the two variables in the pair is excluded. To make decision that which one should be excluded, VIF is calculated for both

variables in the pair and the one with the greater VIF is excluded. The procedure is repeated step by step until none of the remaining pairs have a correlation greater than the threshold (Naimi et al. 2014).

Here, we test the collinearity issue among the 19 bioclim variables in the current time using the `vifstep` function, and exclude collinear variables from all datasets (current and future times):

```
library(usdm) # contains the functions to deal with collinearity

v <- vifstep(bioc, th=10) # checks collinearity with vifstep and threshold of 10

# here is the output of vifstep, reporting which variables should be excluded:
v
```

```
## 10 variables from the 19 input variables have collinearity problem:
##
## bio6 bio11 bio10 bio7 bio16 bio1 bio12 bio17 bio4 bio5
##
## After excluding the collinear variables, the linear correlation coefficients ranges betwe
## min correlation ( bio14 ~ bio9 ):  -0.02655598
## max correlation ( bio18 ~ bio13 ):  0.7057731
##
## ---------- VIFs of the remained variables --------
##   Variables      VIF
## 1      bio2 2.560408
## 2      bio3 2.741559
## 3      bio8 1.445355
## 4      bio9 2.103015
## 5     bio13 4.235066
## 6     bio14 2.387933
## 7     bio15 2.287600
## 8     bio18 2.798137
## 9     bio19 1.408472
```

```
# Now, let's exclude the collinear varibales from all data objects:
# ... exclude collinear variables from bioc (cropped bio) based on "v"
bioc <- exclude(bioc,v)

# now, you can see that bioc has the selected variables:

bioc
```

```
## class       : SpatRaster
## dimensions  : 300, 360, 9  (nrow, ncol, nlyr)
## resolution  : 0.1666667, 0.1666667  (x, y)
## extent      : 60, 120, 10, 60  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## source(s)   : memory
## names       :     bio2,     bio3,      bio8,      bio9, bio13, bio14, ...
## min values  :  3.83750, 14.90886, -15.74217, -32.20467,     3,     0, ...
## max values  : 18.35242, 76.99110,  36.88192,  34.43004,  2381,    77, ...
```

```
# let's do it for objects the objects of global scale (bio, biof):
bio <- exclude(bio, v)

biof <- exclude(biof, v)
```

```
# If you want to save the new data objects, you can use the writeRaster function
# Example:....> bio <- writeRaster(bio, filename="bioclim_selected_current.tif")

#-----------
# as you can see, the selected variables are in bio (and in biof and bioc):

bio
```

```
## class       : SpatRaster
## dimensions  : 1080, 2160, 9  (nrow, ncol, nlyr)
## resolution  : 0.1666667, 0.1666667  (x, y)
## extent      : -180, 180, -90, 90  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## sources     : wc2.1_10m_bio_2.tif
##               wc2.1_10m_bio_3.tif
##               wc2.1_10m_bio_8.tif
##               ... and 6 more source(s)
## names       :     bio2,       bio3,      bio8,      bio9, bio13, bio14, ...
## min values  :  1.00000,   9.131122, -66.29942, -54.26688,     0,     0, ...
## max values  : 21.14754, 100.000000,  37.70479,  37.43358,  2381,   484, ...
```

(iii) Developing models using the sdm R package:

Now, we have both species (response) and climate (predictors) variables ready to implement the modelling workflow using the sdm package. The package can be installed normally from CRAN or from Github using: `devtools::install_github('babaknaimi/sdm')`. **To get all modelling methods available for training SDMs, some additional packages also need to be installed. You may use the `installAll()` function one time after installing sdm to get all required packages installed on your machine.**

In the sdm package, two user-friendly functions should be used to develop the models. First, an sdmdata object should be created using the `sdmData` function, then, models are trained and evaluated using the `sdm` function.

The usage of the function:

```
sdmData(formula,train,predictors,test,bg,filename,crs,impute,metadata,...)
```

The `sdmData` function requires both species and predictor variables and several issues with data (e.g., duplications, missing values) and inconsistencies (e.g., mis-match in coordinate reference systems) are checked and fixed by this function. It has a `formula` interface (check the following box) which provides flexibility for users to control data structure and formats, and apply several functions on data.

BOX 1: Formula interface in the sdmData function

The "formula" interface in R is a powerful and versatile tool that simplifies the specification of statistical models. Central to many modeling functions in R, the formula interface allows users to define the relationship between the dependent and independent variables in a clear and concise manner. This interface uses a symbolic syntax to represent models, making it intuitive and accessible for users to specify complex models without delving into the underlying mathematical details.

At its core, the formula interface uses a simple syntax: the tilde ( ) operator separates the response variable on the left from the predictor variables on the right. For example, in the formula y x1 + x2, y is the dependent variable (species), and x1 and x2 are the independent variables (predictors; environmental variables). In the sdm R package, the formula interface has been extended to support

specifying different types of data (e.g., categorical/factor variables, time, coordinates), and provide flexibility in controlling data processing and modelling by users.

The formula interface is implemented as the first argument in the two main functions in the sdm package, including `sdmData`, and `sdm` Following syntax are examples to illustrate how the formula interface in the sdm package provides flexibility which contributes to make the package user-friendly. The formula in both the `sdmData`, and `sdm` functions is used to specify the species in the left-hand side, and predictors in the right-hand side. Let's assume the name of species in our data is "species", and we used three predictors named "bio1", "bio2", and "bio3", then, the formula can be defined as:

```
sdmData(species ~ bio1 + bio2 + bio3, …)
```

We can simply use "." in the right-hand side to avoid typing all the variables names, meaning use all existing variables:

```
species ~ .
```

Categorical variables: If we have categorical variables in our dataset, we can explicitly specify them by using the term of "factor" or simply "f" in the formula (refers to factor data type that is used in R for handling categorical data). Let's assume our categorical variable is "landuse", then the formula would be:

```
species ~ . + f(landuse)
```

Spatial coordinates: When we use spatial data objects (e.g., spatial points and raster), coordinates of species locations are extracted from the dataset, but when the input dataset is a data.frame and if spatial coordinates are available (e.g., two columns of the data.frame are coordinates with the names of "lon" and "lat"), we can specify them in the formula:

```
species ~ . + f(landuse) + coords(lon+lat)
```

Time (temporal data): If data/time records are available (e.g., date of species observations), it can be specified using the term of "time":

```
species ~ . + f(landuse) + coords(lon+lat) + time(eventDate)
```

Multiple species: The sdm package supports modelling for simultaneously multiple species. When we have multiple species in the dataset, their names can be included in the lef-hand side of the formula as:

```
species1 + species2 + species3 ~ . + f(landuse)
```

If the left-hand side of the formula is left empty, the package tries to detect the available species data in the input dataset by assuming all variables having binomial records (1 and 0), are species variables (presence-absence):

```
~ . + f(landuse)
```

Selecting variables: Another formula term in the sdm package, "select", can be used in data preparation procedure to select a subset of variables using a variable selection procedure. For instance, assessing the issue of multi-collinearity in data may be resulted to exclude some problematic variables and keep a subset. The multi-collinearity assessment in the sdm package uses two different approaches, "vifcor" and "vifstep" (Naimi et al., 2014), implemented in the usdm R package. By using "select", a user can specify names of (numerical) variables for which the collinearity should be checked: The following example checks collinearity for all numeric variables using the vifstep function given the threshold of 10:

```
species ~ . + select(., method="vifstep",th=10)
```

The following example checks collinearity for all numeric variables except "bio1" using the vifcor function given the threshold of 0.7:

```
species~ . + select(. - bio1, method="vifcor", th=0.7)
```

Scaling the variables: Given that the range (gradient) of predictor variables is usually different and not directly comparable to each other, transforming them to a similar range (scaling) can significantly enhance model convergence and performance. Scaling ensures that each predictor variable contributes equally to the model, preventing variables with larger ranges from disproportionately influencing the results. This is especially important for modeling algorithms sensitive to the scale of input data, can also facilitate the optimization process, leading to faster and more stable convergence of the model parameters.

In The sdm package, it is straightforward and easy to use the scale term in the formula to standardize predictor variables before model fitting. By using the scale function, users can ensure that their models are robust, efficient, and interpretable, as this step is a best practice in the species distribution modeling workflow, reinforcing the reliability and effectiveness of the models developed using the sdm package. Two scaling methods are available in the package, "minmax" (default) transforms the data to a range between 0 and 1, and "center" scales each variable by subtracting its mean, divided by its standard deviation:

```
species ~ . + scale(.) # scaling of all variables using the default method
("minmax")

species ~ . + scale(bio1 + bio2) # scaling of only bio1 and bio2 variables

species ~ . + scale(., method="center") # scaling all using the method of
"center"
```

Principle component analysis (PCA): PCA is a valuable technique for reducing the dimensionality of data and addressing issues of multi-collinearity among predictor variables. In the context of species distribution modeling, PCA transforms the original numerical variables into a set of uncorrelated components, which can then be used as predictors in the model. This transformation helps in simplifying the model by retaining only the most significant components, thereby enhancing computational efficiency and model interpretability. The sdm R package facilitates the incorporation of PCA directly within the formula interface. By specifying a PCA term in the formula, users can automatically transform the numerical variables into principal components, selecting either the first n components or those that explain a specified portion of the total variance (e.g., 80

The following line of code re-scales and transforms all numeric variables using PCA and selects the first two generated components to be used as predictor variables in the SDM workflow:

```
species ~ pca(. , n=2 , scale = TRUE)
```

Alternatively if n = "auto", the function identifies the best number of components which is the default behaviour; i.e., the following line of code is equivalent to: species ~ pca(.) :

```
species ~ pca(. , n= "auto", scale = TRUE)
```

Or you may specify the minimum percentage of variability explained by the selected components:

```
species ~ pca(. , n=0.8 , scale = TRUE)
```

By using the formula interface, users can leverage R's extensive statistical modeling capabilities with ease. This interface not only promotes consistency across different modeling functions but also enhances readability and reproducibility of the code. As such, the formula interface is an essential feature of R that empowers users to build, analyze, and interpret a wide array of statistical models efficiently.

The sdmdata object, generated by the `sdmData` function, is like a database containing all species and environmental records as well as additional information (spatial coordinates, temporal dimension, grouping factors, coordinate reference system, metadata, etc.).

Let's create the `sdmdata` object given the species data (`SpatVector` object: `sp`) and the climate variables (`SpatRaster` object: `bioc`). In the species data (`sp`), we had the presence-only values (1) assigned to a column named `species`.

Backgrounds (pseudo-absences): Since the species dataset is presence-only, we need to generate background (pseudo-absence) records. Several methods are available to generate backgrounds (you may check the help page of the `background` function in the sdm package). For this purpose, we use the method of `gRandom` which generates background randomly in the geographic space. The setting to generate background records can be provided through the `bg` argument to the `sdmData` function using a list. In addition to the method, `n` (the number of background records), and some optional arguments can be provided (e.g., `bias` which is a raster file can be used for a target-based background generation).

```
library(sdm)
```

```
## sdm 1.2-51 (2024-10-10)
```

```
## the first step in the sdm package is to create the data object:
# ---> 500 background records will be generated using "gRandom":
d <- sdmData(species~.,train=sp,predictors=bioc,bg=list(n=500,method='gRandom'))

# you may get a summary from the created object by executing the object:

d
```

```
## class                                       : sdmdata
## ============================================================
## number of species                           :  1
## species names                               :  species
## number of features                          :  9
## feature names                               :  bio2, bio3, bio8, ...
## type                                        :  Presence-Background
## has independent test data?                   :  FALSE
## number of records                           :  615
## has Coordinates?                            :  TRUE
```

Modelling and Evaluation: After creating the data object (`sdmdata`), the `sdm` function can be used to train and evaluate multiple models in parallel. 20+ modelling methods are available in the package. The models can be extended by users through using the `add` function. The list of existing methods can be checked using getmethodNames function:

```
getmethodNames(alt=F)
```

```
##  [1] "bioclim"       "bioclim.dismo" "brt"           "cart"
##  [5] "domain.dismo"  "fda"           "gam"           "glm"
##  [9] "glmnet"        "glmpoly"       "mahal.dismo"   "mars"
## [13] "maxent"        "maxlike"       "maxNet"        "mda"
## [17] "mlp"           "ranger"        "rbf"           "rf"
```

```
## [21] "rpart"          "svm"
```

Usage of sdm:

```
sdm(formula, data, methods,...)
```

In the sdm function, formula (to specify the structure of the model), data (the output of sdmData), and methods (to specify the names of modelling methods) are the minimum inputs to get the function works. Some additional inputs can be provided to specify settings for the replication methods which split data to training and test partitions for evaluating the models. The following sub-section provides some details about model evaluation:

Replication: To evaluate the models, if an independent dataset is available, thay can be used to test the performance of models. If that is the case, the independent test data can be provided to the sdmData function. However, such independent data are not available in most situations. Therefore, an alternative solution might be used which is to split the data into two partitions using a re-sampling method. Several re-sampling methods are available in the sdm package including "sub-sampling" (uses a random sampling without replacement to divide data given the percentage specified by a user; for example, if test.percent = 30 is specified by a user, 30% of records are randomly selected to be used as test data and the rest for training the models), "bootstrapping" (uses a random sampling with replacement; for this method, a random sample with replacement and with the same size as the original data is drawn from the records which are used for training the models, and the records that are not selected in the sample are used as the test data), and "cross-validation" (divide data into n folds/partitions [cv.fold specified by a user] at the beginning of the procedure and the models are trained n times and every time one of the folds is selected as the test data). A user can specify one or more of these methods through the replication argument, and the procedure can be repeated multiple times if a user add the n argument with the number of replications.

Here is an example of using the sdm function where we chose 5 modelling method, also used sub-sampling with 30% of data for test dataset, and repeated the procedure 3 times:

```
## second step: to train and evaluate the models:

#----> 5 modelling methods are used:
# =======>> glmp: polynomial generalised linear model;
# =======>> brt: Boosted Regression Trees;
# =======>> rf: Random Foress;
# =======>> maxent: Maximum Entropy;
# =======>> svm: Support Vector Machine;

#----> Replication method: sub-sampling (with 30 % of test data; repeats 3 times)

m <- sdm(species~., d, methods = c('glm','brt','rf','maxent','svm'),
         replication='sub', test.p=30, n = 3)


# to see a summary of the modelling outputs, you may check the object:

m
```

```
## class                                  : sdmModels
## ========================================================
## number of species                      :  1
## number of modelling methods            :  5
## names of modelling methods             :  glm, brt, rf, maxent, svm
## replicate.methods (data partitioning)  :  subsampling
```

```
## number of replicates (each method)    :  3
## toral number of replicates per model  :  3 (per species)
## test percentage (in subsampling)      :  30
## ----------------------------------------
## model run success percentage (per species)  :
## ----------------------------------------
## method          species
## ---------------------
## glm        :        100   %
## brt        :        100   %
## rf         :        100   %
## maxent     :        100   %
## svm        :        100   %
##
## ######################################################################
## model Mean performance (per species), using test dataset (generated using partitioning):
## ------------------------------------------------------------------------------
-----
##
##  ## species   :  species
## ========================
##
## methods    :     AUC     |     COR     |     TSS     |     Deviance
## ------------------------------------------------------------------------------
## glm        :     0.86    |     0.59    |     0.62    |     0.85
## brt        :     0.94    |     0.73    |     0.84    |     0.58
## rf         :     0.98    |     0.86    |     0.89    |     0.29
## maxent     :     0.96    |     0.79    |     0.86    |     0.41
## svm        :     0.95    |     0.75    |     0.82    |     0.47
```

As you can see above, in the summary information provided from the `sdmModels` object, 3 models were trained for each modelling method (15 models in total), and they were evaluated using multiple accuracy metrics (e.g., AUC, TSS). Mean values of these metrics can provide a quick way to compare the performance of different methods.

There is a user-friendly and interactive way to explore part of the results available within the sdmModels object through using the `gui` function. This function opens a graphical user interface with different parts (windows/tabs) which allows a user to explore the results (evaluation, variable importance, etc.). You may simply use the following line of code to use the function:

```
gui(m)
```

% Define a custom color for the header % Define a custom color for the row names

BOX 2: Model evaluations for SDMs

Evaluating the performance of models is a crucial step in species distribution modeling (SDM) to ensure the accuracy and reliability of predictions. Several evaluation metrics are commonly used to assess how well a model predicts the occurrence of species based on known data. Here are some of the most widely used methods:

Area Under the Receiver Operating Characteristic Curve (AUC):

Confusion Matrix and Threshold-Based Metrics

The confusion matrix is a tool used to evaluate the performance of a classification model by comparing the observed values with the predicted values. The matrix is composed of four key components:

	Predicted Presence	Predicted Absence
Observed Presence	True Positives (TP)	False Negatives (FN)
Observed Absence	False Positives (FP)	True Negatives (TN)

Description of the Metrics:

1. **True Positives (TP)**: The number of observations correctly predicted as presence by the model (i.e., the model correctly identifies the presence of the species).

2. **False Negatives (FN)**: The number of observations where the model incorrectly predicts absence, while the species is actually present (i.e., missed presences).

3. **False Positives (FP)**: The number of observations where the model incorrectly predicts presence, while the species is actually absent (i.e., false alarms).

4. **True Negatives (TN)**: The number of observations correctly predicted as absence by the model (i.e., the model correctly identifies where the species is absent).

Key Metrics Derived from the Confusion Matrix:

- **Accuracy**:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy measures the overall correctness of the model by calculating the proportion of correctly classified instances (both presences and absences) out of all predictions.

- **Sensitivity (True Positive Rate / Recall)**:

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

Sensitivity evaluates how well the model identifies actual presences (true positives). A higher sensitivity indicates fewer missed presences.

- **Specificity (True Negative Rate)**:

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Specificity assesses how well the model identifies actual absences (true negatives). A higher specificity means fewer false positives (wrongly predicted presences).

- **Precision (Positive Predictive Value)**:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Precision measures the proportion of predicted presences that are true presences. A higher precision means that when the model predicts presence, it is more likely to be correct.

- **F1 Score**:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1 score is the harmonic mean of precision and recall, giving a balanced measure of the model's accuracy when there is a trade-off between false positives and false negatives.

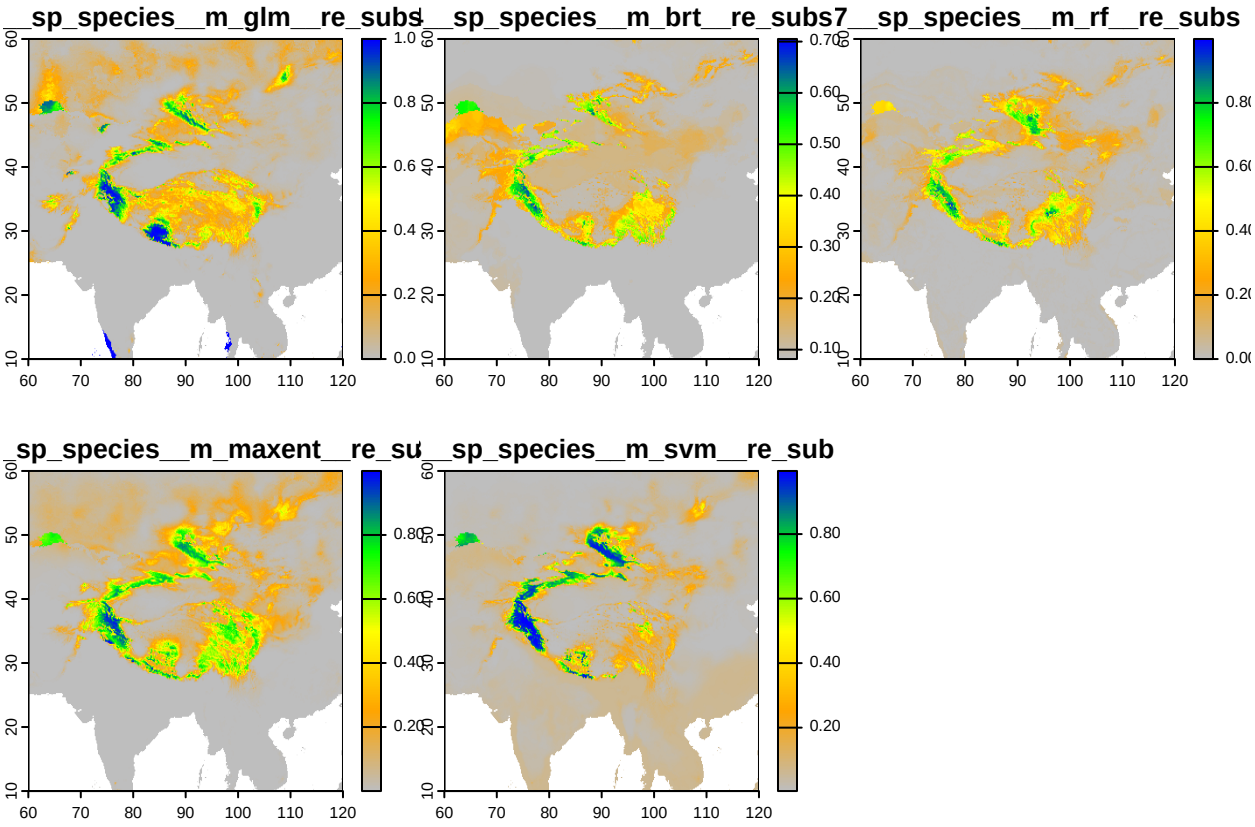
(iv) Predict/Project the map of habitat suitability in current and future times

After the models were trained and evaluated successfully, we can now use them to predict the probability of occurrence (it may be interpreted as the degree of habitat suitability) for each pixel across the study area. Here, the aim is to assess the geographic distribution of the species globally, therefore, we use the non-cropped predictor variables (`bio`) to predict the values in the current time:

```
## Predict habitat suitability of Snow leopard in the current time:

p <- predict(m, bioc)

# since we had 15 models in the `m` object, `p` is a SpatRaster with 15 layers
# corresponding to the 15 models:

p
```

```
## class       : SpatRaster
## dimensions  : 300, 360, 15  (nrow, ncol, nlyr)
## resolution  : 0.1666667, 0.1666667  (x, y)
## extent      : 60, 120, 10, 60  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## source(s)   : memory
## names     : id_1_~_subs, id_2_~_subs, id_3_~_subs, id_4_~_subs, id_5_~_subs, id_6_~_sub
## min values  : 2.220446e-16, 2.220446e-16, 2.220446e-16,  0.08173233,  0.08082456,  0.08162
## max values  : 1.000000e+00, 1.000000e+00, 9.999857e-01,  0.70507237,  0.70709992,  0.70496
```

```
# We have 5 modelling methods, and 3 models for each method (15 in total),
# the first 3 layers are for the first method ("glmp"), the secord 3 (4, 5, and 6)
# are related to the second method ("brt"), and so on.

# let's visualise one of the outputs for each method:

# appropriate color schemes:
cl <- colorRampPalette(c('gray','orange','yellow','green','blue'))

plot(p[[c(1,4,7,10,13)]],col=cl(200))
```



Ensemble forecasting: Employing multiple modelling methods for species distribution modelling which is usually applied to several replications of data (e.g., in the example above, we produced 15 models), often results in varying predictions. While these models may all perform well according to evaluation metrics (e.g., AUC), their predictions can still differ significantly. These variations in outcomes of models are referred to model-based uncertainty, which is one of the key sources of uncertainty affecting SDMs.

To address this uncertainty, an effective approach is ensemble forecasting, where the predictions from multiple models are combined into a single, consensus prediction (Araujo and New 2007). Ensemble forecasting helps to smooth out the variability in individual models, providing a more robust and reliable prediction of species distribution. According to Araújo and New (2007), ensemble methods can improve the accuracy and reliability of SDMs by leveraging the strengths of different models while minimizing their individual weaknesses.

Ensemble forecasting methods: In the sdm package, the `ensemble` function can be used for generating the consensus prediction that supports multiple methods for combining the predictions. Several approaches are available to combine predictions from multiple models in the `ensemble` function (the method is specified within a list passed through the `setting` argument:

- **Mean** (simple averaging): This is the simplest method, where the predictions from all models are averaged to produce the final consensus prediction. Each model contributes equally to the final result. To use this method, `method = 'mean'` or `method = 'unweighted'` can be used in the `setting` argument of the `ensemble` function.

- **Weighted Mean**: In this method, models with better performance (as determined by evaluation metrics such as AUC or TSS) are given more weight in the final prediction. For example, models that perform better based on AUC would contribute more significantly to the final ensemble prediction than models with lower AUC scores. Examples of settings in the `ensemble` function include: `setting = list(method = 'weighted', stat = 'auc')` which uses AUC values as weight; OR `setting`

= list(method = 'weighted', stat = 'tss', opt=2) which uses TSS values as weight. Since TSS is a threshold-based metric, the opt argument is also added which specifies which threshold optimisation method should be used to select the best threshold for calculating TSS. 15 optimisation methods are available in the sdm package (check the help page of the evaluates function), and opt = 2 refers to the second method that selects threshold under which [sensitivity + specificity] is maximized (i.e., max[se+sp]).

- **Median**: This method takes the median of predictions from all models. To use this method (method = 'median').

- **Thresholding and stacking**: This method involves thresholding predictions to determine areas of presence or absence given the best threshold for each model specified using a threshold optimisation method (the opt argument). The predicted presence-absence values from multiple models are stacked and averaged (method = 'pa'). Example: setting = list(method = 'pa', opt=2)

- **Two-step averaging**: When multiple predictions are generated for each modelling method (from multiple data replications), the averaging of predictions can be done in two steps, across replications for each modelling method and then across modelling methods. The available two-step methods include mean-weighted, mean-unweighted, median-weighted and median-unweighted. For instance, using the method = 'mean-weighted' in the setting argument first takes the mean of different replications of each modelling methods, then they are combined through using a weighted-mean function.

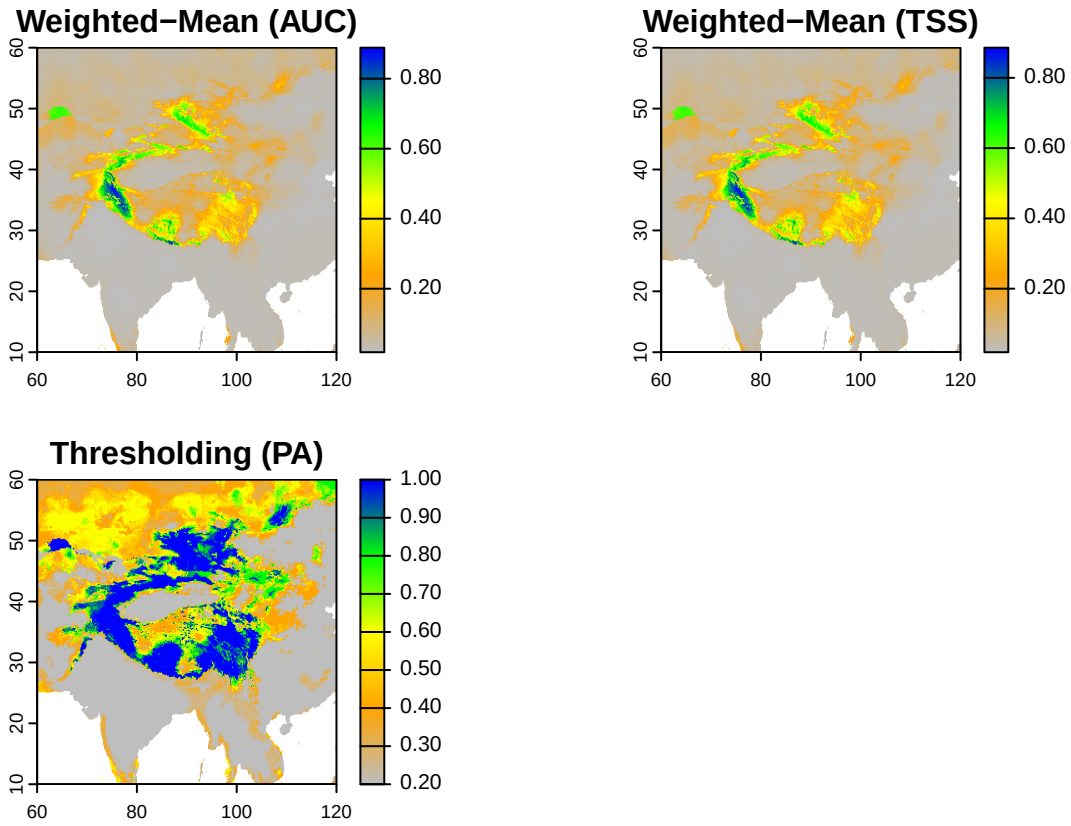
- **Assess variability in predictions**: In addition to generating a consensus prediction, several methods are avaialble to explore the variability of predictions for each location. These methods include coefficient of variation (method = 'cv'), standard deviation (method = 'stdev'), confidence interval (method = 'ci'; this generates the range of confidence interval which is "upper range - lower range"), and uncertainty (method = 'uncertainty'). The uncertainty method can characterise model-based uncertainty or inconsistency among predictions of different models. An entropy metric is used to characterise uncertainty, ranging between 0 and 1. The value of 1 at a certain location refers to maximum inconsistency which is the case when half of the models predict 'presence' for the location and the other half predict 'absence'.

The above methods mentioned methods can be employed together at the same time.

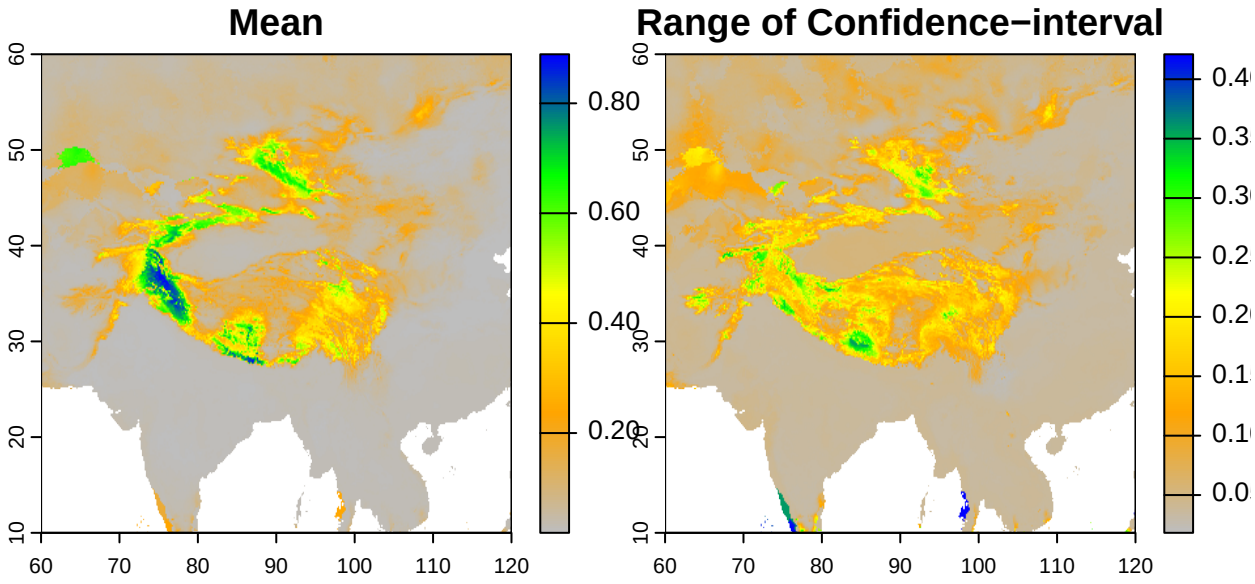
Following is examples of how to generate an ensemble prediction using the ensemble() function in the sdm package. Please **note** that the ensemble function first calls the 'predict' function to generate predictions, and then use an ensemble method to combine them into a consensus map. However, if the predictions are generated beforehand, the output of the predict function can be introduced as the second argument to the ensemble function (in our case, p), otherwise, the predictors object (in our case, bioc) should be used as the second argument:

```
## Ensemble:
#----> first argument: the model (output of the sdm function)
#---->  second argument: either the output of the predict function (if available)
#---------------------- OR the predictors object (bioc; )
#----> third argument (setting): a list with settings for ensemble procedure

### We fitst try 3 methods and compare them in a plot:

## (i) Weighted-mean using values of AUC as weights
en1 <- ensemble(m, p, setting = list(method='weighted', stat= 'auc'))

#----
## (ii) Weighted-mean using values of TSS as weights
```

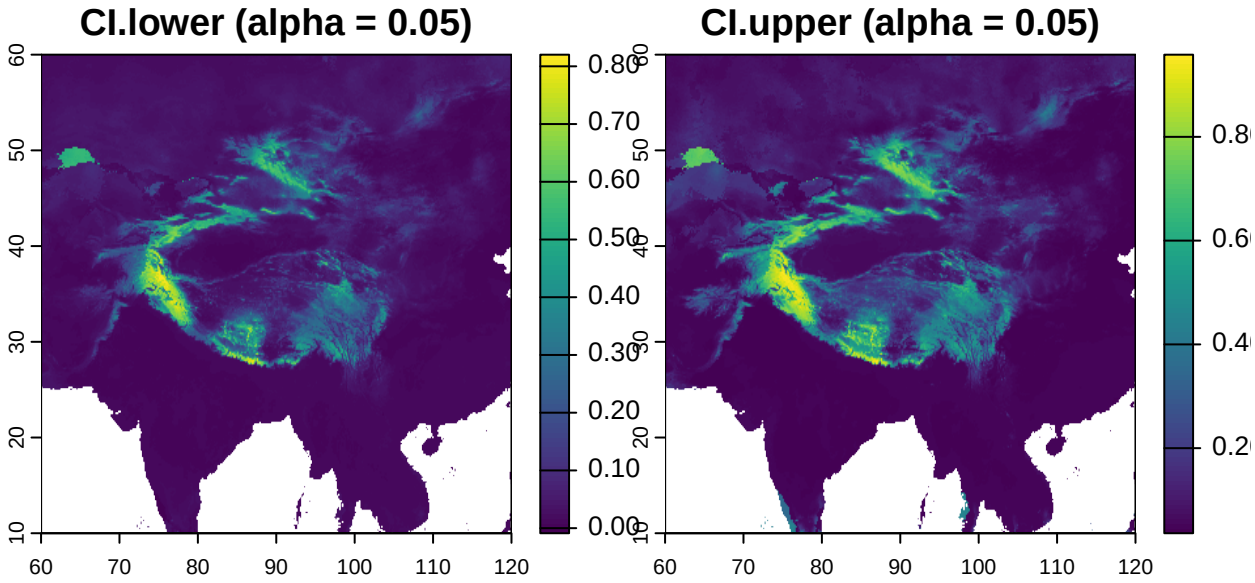
```
# opt=2 refers to the threshold obtained using max[Sensitivity + Specificity]

en2 <- ensemble(m, p, setting = list(method='weighted', stat= 'tss', opt=2))

## (iii) PA: thresholding and stacking:
en3 <- ensemble(m, p, setting = list(method='pa', opt=2))

#------
## Visualising the outputs of 3 different ensembles:


en <- c(en1,en2,en3) # 3 objects stacked in a single object

plot(en, col=cl(200),
     main=c('Weighted-Mean (AUC)','Weighted-Mean (TSS)','Thresholding (PA)'))
```



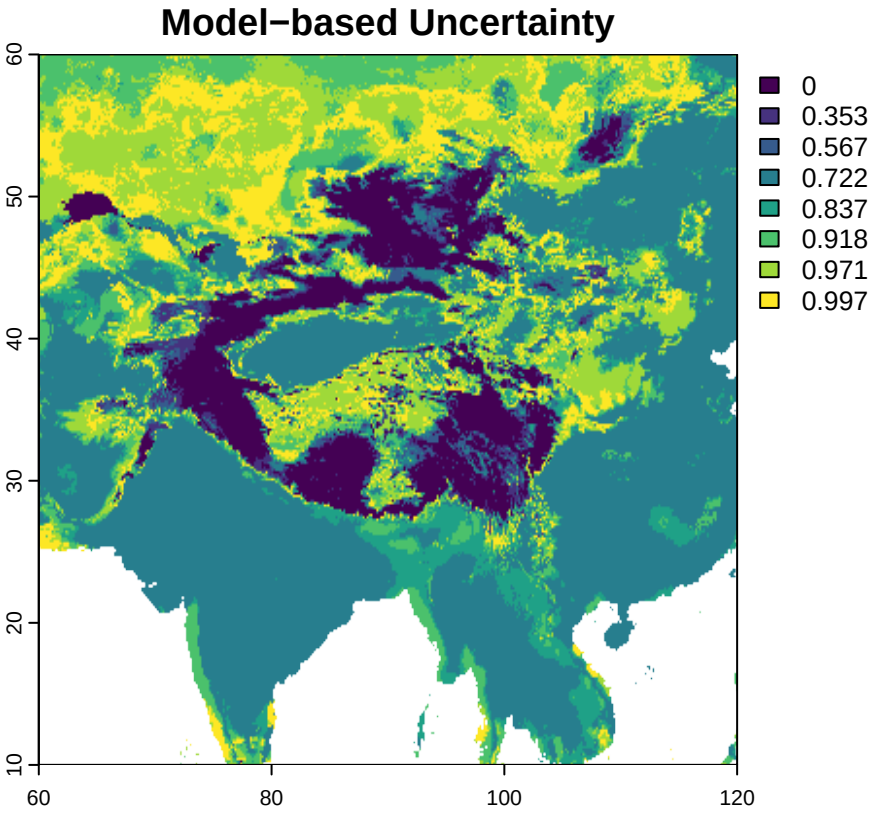

```
#------------------------------
## Assessing variability among predictions:

en4 <- ensemble(m, p, setting = list(method=c('mean','ci'), opt=2))

plot(en4, col=cl(200),main=c('Mean','Range of Confidence-interval'))
```




```
## to assess the upper and lower limits of confidence intervals:
ci.lower <- en4[[1]] - (en4[[2]] / 2)
ci.upper <- en4[[1]] + (en4[[2]] / 2)

plot(c(ci.lower,ci.upper),main=c('CI.lower (alpha = 0.05)','CI.upper (alpha = 0.05)'))
```




```
#----------
## Uncertainty
en5 <- ensemble(m, p, setting = list(method='uncertainty', opt=2))

plot(en5, main='Model-based Uncertainty')
```




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

As you can see above, a single map is generated by combining the 15 predicted rasters (output of the predict function) using the ensemble method specified in the setting. There are other optional arguments avaialable to control the ensemble procedure including:

- `id`: By default, predictions from all models are contributing to generate the consensus map unless a user specify the `id` of certain predictions (based on their `modelIDs`). For example, adding `id = 1:10` to the `setting` list, only uses the first 10 predictions in the above example.
- `expr`: An expression can be specified by a user to control or limit which predictions should contribute into generating the consensus map. For example, using `expr = auc > 0.7` uses the specified condition to only include well-performing models (those with AUC > 0.7) contributing to the ensemble procedure.
- `power`: This argument is used to change weights by calculating `weights = weights ^ power`. A value of greater than 1, gives more emphasise on better performing models.
- Check the help page of the `ensemble` function in the `sdm` package for the other arguments and more details about the function.

(v) Assessing range shifts of the species in response to climate change:

Araujo, M, and M New. 2007. "Ensemble Forecasting of Species Distributions." Trends in Ecology & Evolution 22 (1): 42–47. https://doi.org/10.1016/j.tree.2006.09.010.

Araújo, Miguel B., Robert P. Anderson, A. Márcia Barbosa, Colin M. Beale, Carsten F. Dormann, Regan Early, Raquel A. Garcia, et al. 2019. "Standards for Distribution Models in Biodiversity Assessments." Science Advances 5 (1): eaat4858. https://doi.org/10.1126/sciadv.aat4858.

Ceballos, Gerardo, Paul R. Ehrlich, and Rodolfo Dirzo. 2017. "Biological Annihilation via the Ongoing Sixth

Mass Extinction Signaled by Vertebrate Population Losses and Declines." Proceedings of the National Academy of Sciences 114 (30). https://doi.org/10.1073/pnas.1704949114.

Dirzo, Rodolfo, Hillary S. Young, Mauro Galetti, Gerardo Ceballos, Nick J. B. Isaac, and Ben Collen. 2014. "Defaunation in the Anthropocene." Science 345 (6195): 401–6. https://doi.org/10.1126/science.1251817.

Dormann, Carsten F, Jane Elith, Sven Bacher, Carsten Buchmann, Gudrun Carl, Gabriel Carré, Jaime R García Marquéz, et al. 2013. "Collinearity: A Review of Methods to Deal with It and a Simulation Study Evaluating Their Performance." Ecography 36 (1): 27–46.

Dornelas, M., J. M. Chase, N. J. Gotelli, A. E. Magurran, B. J. McGill, L. H. Antão, S. A. Blowes, G. N. Daskalova, B. Leung, et al. 2023. "Looking Back on Biodiversity Change: Lessons for the Road Ahead." Philosophical Transactions of the Royal Society B: Biological Sciences. https://doi.org/10.1098/rstb.2022.0199.

Guisan, Antoine, and Wilfried Thuiller. 2005. "Predicting Species Distribution: Offering More Than Simple Habitat Models." Ecology Letters 8 (9): 993–1009. https://doi.org/10.1111/j.1461-0248.2005.00792.x.

Guisan, Antoine, and Niklaus E. Zimmermann. 2000. "Predictive Habitat Distribution Models in Ecology." Ecological Modelling 135 (2-3): 147–86. https://doi.org/10.1016/S0304-3800(00)00354-9.

Hughes, A. C. 2023. "The Post-2020 Global Biodiversity Framework: How Did We Get Here, and Where Do We Go Next?" Integrative Conservation 2. https://doi.org/10.1002/inc3.16.

Kolbert, E. 2014. The Sixth Extinction: An Unnatural History. https://doi.org/10.1038/ngeo1895.

Naimi, Babak, and Miguel B. Araújo. 2016. "Sdm: A Reproducible and Extensible R Platform for Species Distribution Modelling." Ecography 39 (4): 368–75. https://doi.org/10.1111/ecog.01881.

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 (2): 191–203. https://doi.org/10.1111/j.1600-0587.2013.00205.x.