Preprocessing data for the presence-only distribution model of pines (family Pinacea)

Juan M. Escamilla Mólgora,

October 25, 2021

This is a script for loading and preprocessing the data for example 1; inferring the distribution of pines.

1 Necessary libraries

- **CAR-1SDM** loads the necessary functions for the models.

```
setwd("R")
#library("CAR-1SDM")
```

- **CARBayes** is a CAR implementation developed by [1]. We can use this library with adequate data transformation to apply models I, II and III.

```
library(CARBayes)
```

- Some libraries for data handling and loading numpy arrays.

```
library(dplyr)
library(purrr)
library(reticulate)
```

1.1 Loading the *choosing principle* functions

These are functions that define the presence, relative absences and missing observations.

```
source("ChoosingPrinciples.R")
```

1.2 Read the adjancency matrix of the spatial lattice

The spatial lattice is represented as an adjacency matrix obtained by another sofware. However you could generate this matrix of any given spatial vector format (e.g. ESRI Shapefile) using the function 'combine.data.shapefile' provided by **CARBayes**. Refer to the CARBayes documentation.

```
#file = '../data/presence_only_models/predictors/dataset100x100-puebla-p9/0-pred.csv'
#PDF = read.csv(file)
## REad adjancency matrix
# Import adjancency matrix generated from region
mat_filename = "../data/training_data_sample_puebla_p9_abies_pinophyta_adjmat.npy"
# Use numpy functions
np <- import("numpy")
M <- np$load(mat_filename)
```

1.3 Read *training* dataframe

We will first load the dataframe as 'TDF' and then sort the rows according to their 'cell_{ids}' to match that of the adjancency matrix M.

```
TDF = read.csv("../data/training_data_sample_puebla_p9_abies_pinophyta.csv")
#TDF = read.csv("../data/training_data_sample_puebla_p9_tyrannidae_birds.csv")
## Order it according to the id of the cell
TDF = TDF[order(TDF$cell_ids),]
# Convert the columns to numeric

## Beware older implementations of as.numeric function turns strings to (apparently) ra
TDF = mutate_at(TDF,vars(Dist.to.road_m,Elevation_m,
                         MaxTemp_m,MeanTemp_m,
                         MinTemp_m,Population_m,
                         Precipitation_m,
                         SolarRadiation_m,
                         VaporPres_m,
                         WindSp_m),na_if,"N.A.")



TDF = mutate_at(TDF,vars(Dist.to.road_m,Elevation_m,
                         MaxTemp_m,MeanTemp_m,
                         MinTemp_m,Population_m,
```

```
                        Precipitation_m,
                        SolarRadiation_m,
                        VaporPres_m,
                        WindSp_m),as.numeric)

# Remove unnecessary symbols in variable names
names(TDF) = lapply(names(TDF),function(x) gsub("_","",x))
names(TDF) = lapply(names(TDF),function(x) gsub("\\.","",x))
```

1.4 Generating relative absences with the *choosing principle*

There are two treatments for the assigning missing data, treatment I that assumes missing data in both processes and treatment II that assigns missing data only to the sampling effort. This later treatment has less uncertainty but the assumption of absences is strong. Uncomment the code to use the other treatment.

```
# Change the name of a column that for some reason is called the same
names(TDF)[23] <- 'code_id2'

## Treatment I, missing values in X and Y, comment this if using treatment II
DataFrame = TDF %>% rowwise() %>%
              mutate(sample=pseudo_absence_naive(Plantae,LUCA),
                     species=pseudo_absence_naive(Pinophyta,Plantae))

## Uncomment this for treatment II (i.e. missing values only in X)
#DataFrame = TDF %>% rowwise() %>%
#            mutate(sample=pseudo_absence_naive(Plantae,LUCA),
#                   species=pseudo_absence_trivial(Pinophyta,Plantae))

## Uncomment this if you want to assume that all missing data are absences
## i.e. remove NAs
#
## Remove entries in the adjacency matrix that correspond to missing data
#rr <- DataFrame %>%
#    filter(!is.na(species) & !is.na(sample))
#
#sam_idx_nan <- which(is.na(DataFrame$sample))
#
```

```
#M = M[-c(sam_idx_nan),-c(sam_idx_nan)]

## Remove missig data in DataFrame
#DataFrame = TDF %>% rowwise() %>%
#  mutate(sample=pseudo_absence_trivial(Plantae,LUCA),
#         species=pseudo_absence_trivial(Pinophyta,Plantae))
```

1.5 Preprocess adjancency matrix M

There is a caveat, though. The CAR model works for connected regions only. That is, when the adjacency matrix corresponds to a connected graph. Blocks of cells that are not connected to each other are called islands and are considered independent. That is, a new model should be run for each block of disconnected cells.

That is, we need to identify the blocks of isolated cells and remove them from the adjacency matrix. On order to comply with the dimension of the associate covariate (design) matrix we need to remove the associated rows (entries) corresponding to the isolated cells.

```
## Remove entries with zero neighbours (adjacency matrix)
### Calculates number of neighbours in D (sum)
D = apply(M,MARGIN = 1,sum)
### get index with 0 neighbours
idx = which(D == 0)

### select cells with no neighbours
cell_with_no_neighbour = TDF$cellids[idx]
## Remove island for TDF
TDF <- TDF[-c(idx),]
## Erase idx for M and for TDF
M_bis = M[-c(idx),-c(idx)]

## remove rows that have no neighbours (islands)
DataFrame <- DataFrame[-c(idx),]

n <- dim(M_bis)[1]
trials <- rep(1,n)
```

As the current implementation of CARBayes does not allow missing values in the covariates we will perform a naive imputation of missing values on

each column of interest (i.e covariates associated with the models' formulas).

```
## Replace missing values with mean, Of course we could do this using other more fancy
covs2work = c("Disttoroadm","Populationm","Elevationm","Precipitationm","MeanTempm")
for(i in covs2work){
    DataFrame[,i][is.na(DataFrame[,i])] <- mean(DataFrame[,i][!is.na(DataFrame[,i])],na
}
```

References

[1] Duncan Lee. CARBayes : An R Package for Bayesian Spatial Modeling with Conditional Autoregressive Priors. *Journal of Statistical Software*, 55(13):1–24, nov 2013.