Tutorial for SDMPlay: 4/ Spatial cross-validation

2021-02-26

Species distribution modelling (SDM) has been developed for several years to address conservation issues, to assess the direct impact of human activities on ecosystems and to predict the potential distribution shifts of invasive species (see Elith et al. 2006, Pearson 2007, Elith and Leathwick 2009). SDM relates species occurrences with environmental information and can predict species distribution on their entire occupied space. **This approach has been increasingly applied to Southern Ocean case studies, but requires corrections in such a context, due to the broad scale area, the limited number of presence records available and the spatial and temporal aggregations of these datasets.**

SDMPlay is a pedagogic package that will allow you to compute SDMs, to understand the overall method, and to produce model outputs. The package, along with its associated vignettes, highlights the different steps of model calibration and describes how to choose the best method to generate accurate and relevant outputs. SDMPlay proposes codes to apply a popular machine learning approach, BRT (Boosted Regression Trees) and introduces MaxEnt (Maximum Entropy). It contains occurrences of marine species and environmental descriptor datasets as examples associated with several vignette tutorials.

Objectives of tutorial #4/ Spatial cross-validation

Cross-validation is a method to evaluate your model. It consists in splitting your initial occurrence dataset into a subset to train the model, and another independent subset to test it. Generally, the random cross-validation procedure randomly partitions the data (e.g. 70% to train, 30% to test). However, when working with presence data spatially aggregated, you may violate the "independency" assumption you made with your test data, as they are spatially very close to the data you have used for training. The evaluation of your model will be consequently biased. In such cases, the spatial cross-validation procedure should be used: the splitting into training and test is not random, but spatially separated. This tutorial provides some elements to apply this method (refering to Guillaumot et al. 2019).

See also...

- **Tutorial #1/ Compute Species Distribution Models**
  Focusses on data structure, data preparation and general model computing.
- **Tutorial #2/ SDM outputs**
  Presents the main outputs you can generate with your SDM.
- **Tutorial #3/ Importance of model calibration**
  Highlights the procedure to accurately calibrate your model and proposes some methods to limit the influence of several biases.
- **Tutorial #5/ Spatial extrapolation**
  Models can extrapolate when projected on broad scale areas. This tutorial provides codes to calculate extrapolation scores and generate extrapolation maps that could be associated to SDM maps (refering to Guillaumot et al. 2020).

Let's begin !

We will work with the *Odontaster validus* case study. Load the occurrence records and environmental layers.

```r
library(SDMPlay)
data("Odontaster.validus") # Species distributed around the entire Southern Ocean, table
                           # with longitude and latitude only
#head(Odontaster.validus)
```

```r
library(SDMPlay)
library(raster)
data("depth_SO")
data("ice_cover_mean_SO")
data("seafloor_temp_2005_2012_mean_SO")

predictors_stack_SO <- raster::stack(depth_SO,ice_cover_mean_SO,
                                     seafloor_temp_2005_2012_mean_SO)
names(predictors_stack_SO)<-c("depth","ice_cover_mean","seafloor_temp_mean")
predictors_stack_SO
```

```
## class      : RasterStack
## dimensions : 350, 3600, 1260000, 3  (nrow, ncol, ncell, nlayers)
## resolution : 0.1, 0.1  (x, y)
## extent     : -180, 180, -80, -45  (xmin, xmax, ymin, ymax)
## crs        : NA
## names      : depth, ice_cover_mean, seafloor_temp_mean
```

```r
 library(RColorBrewer)
 data("worldmap")
 my.palette.oranges <- brewer.pal(n = 9, name = "Oranges")
 my.palette.blue <- rev(brewer.pal(n = 9, name = "Blues"))

#You can load the SOmap package to generate nice figures
#remotes::install_github("AustralianAntarcticDivision/SOmap")
library(SOmap)
```

```r
# create your input matrix
SDMtable <- SDMPlay:::SDMtab(xydata=Odontaster.validus,
                             predictors=predictors_stack_SO,
                             unique.data=FALSE,
                             same=TRUE)
head(SDMtable)
```

```
##   id longitude latitude     depth ice_cover_mean seafloor_temp_mean
## 1  1    166.65   -77.85 -112.5210       0.386371                NA
## 2  1    166.65   -77.85 -112.5210       0.386371                NA
## 3  1    166.65   -77.85 -112.5210       0.386371                NA
## 4  1    166.65   -77.85 -112.5210       0.386371                NA
## 5  1    166.65   -77.85 -112.5210       0.386371                NA
## 6  1    166.45   -77.45 -384.7778       0.384223                NA
```

```r
# sort presence and background data
presences <- subset(SDMtable,SDMtable$id==1)
head(presences)
```

```
##   id longitude latitude     depth ice_cover_mean seafloor_temp_mean
## 1  1    166.65   -77.85 -112.5210       0.386371                NA
## 2  1    166.65   -77.85 -112.5210       0.386371                NA
## 3  1    166.65   -77.85 -112.5210       0.386371                NA
## 4  1    166.65   -77.85 -112.5210       0.386371                NA
## 5  1    166.65   -77.85 -112.5210       0.386371                NA
## 6  1    166.45   -77.45 -384.7778       0.384223                NA
```

```
background <- subset(SDMtable,SDMtable$id==0)
head(background)
```

```
##     id longitude latitude      depth ice_cover_mean seafloor_temp_mean
## 327  0   -106.55   -72.85  -594.4722     0.74236727                NA
## 328  0     41.95   -60.55 -5343.5557     0.19950417         0.7883283
## 329  0    -51.65   -74.65  -459.8889     0.93878442                NA
## 330  0   -139.85   -46.05 -5115.8613     0.00000000         2.9465933
## 331  0   -144.05   -58.45 -3422.0833     0.00271775         0.8161358
## 332  0    161.65   -49.75 -3771.8333     0.00000000         2.8946247
```

Generate a model with a standard random cross-validation

Randomly select 70% of your presence and background data to train your model and the 30% remaining is kept for testing

```
random_choice <- sample (seq(1:nrow(presences)), size=round((70/100)*nrow(presences)))
remaining <- seq(1:nrow(presences))[-random_choice]
presence_training <- presences[random_choice,]
presence_test <- presences[remaining,]

random_choice2 <- sample (seq(1:nrow(background)), size=round((70/100)*nrow(background)))
remaining_back <- seq(1:nrow(background))[-random_choice2]
background_training <- background[random_choice2,]
background_test <- background[remaining_back,]

SDMtable_training <- rbind(presence_training,background_training)
head(SDMtable_training)
```

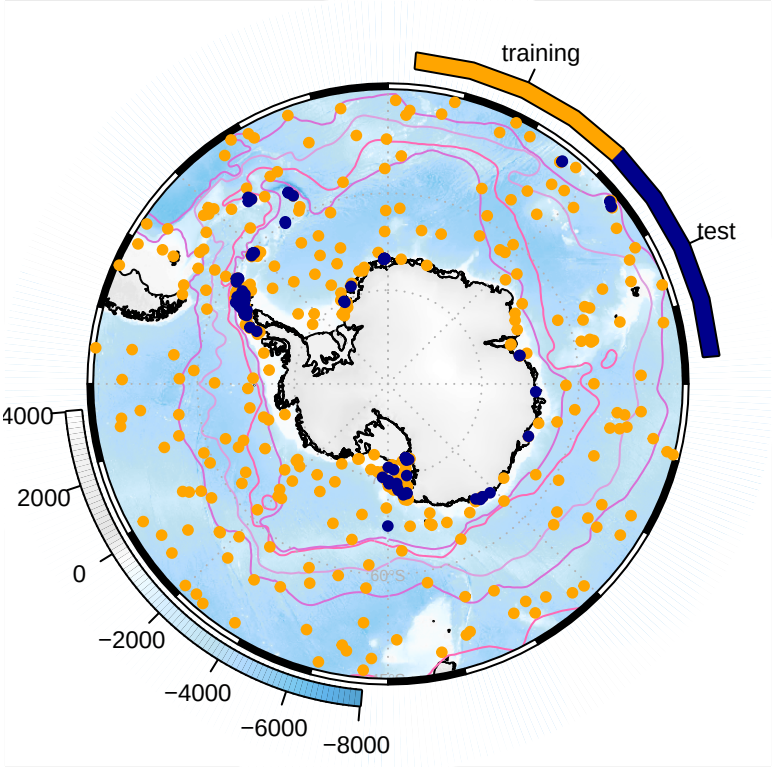
```
##     id longitude latitude      depth ice_cover_mean seafloor_temp_mean
## 221  1    -62.55   -64.25  -57.57156      0.1387166         -0.5271336
## 227  1    -64.15   -64.75 -143.76871      0.2105842          0.4973567
## 2    1    166.65   -77.85 -112.52102      0.3863710                 NA
## 119  1    -58.25   -62.95 -390.63889      0.1800755         -1.3562200
## 103  1    -56.15   -62.65 -375.50000      0.2268452         -1.5231029
## 206  1    -67.35   -65.55 -168.61111      0.2568383          0.8559521
```

```
location_presence_test <- presence_test[,c(2,3)]
head(location_presence_test)
```

```
##    longitude latitude
## 1     166.65   -77.85
## 6     166.45   -77.45
```

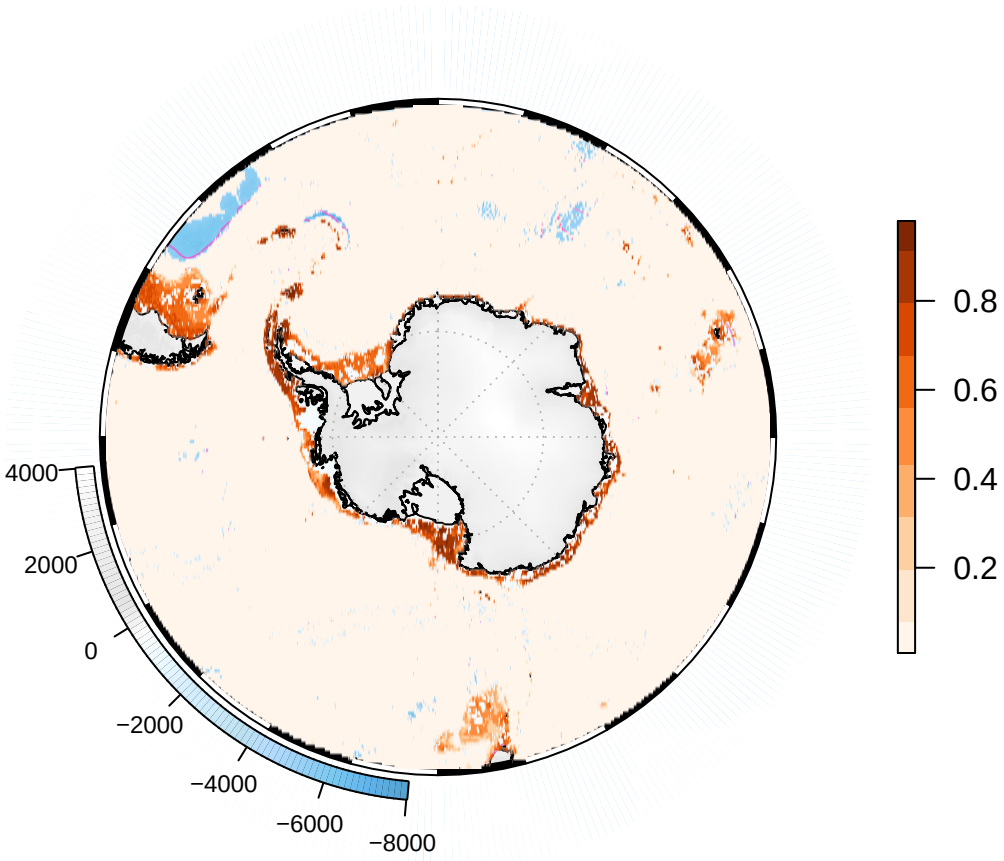
```
## 10    -36.45   -54.25
## 11    -37.45   -54.25
## 13    -45.55   -60.55
## 17    -57.95   -63.35
```

```
# Plot training and test data for the random cross-validation example
basemap <- SOmap(bathy_legend= T, graticules= T, fronts= T, border_width= 0.8)
plot(basemap)
SOplot(SDMtable_training[,2],SDMtable_training[,3], col="orange", pch=20)
SOplot(location_presence_test[,1],location_presence_test[,2], col="darkblue", pch=20)
SOleg(col=c("orange","darkblue"),position = "topright",
      tlabs = c("training","test"), type = "discrete")
```




```
#Run your model
Model_output <- SDMPlay:::compute.brt(x=SDMtable_training,
                                      proj.predictors=predictors_stack_SO,
                                      tc = 2, lr = 0.001, bf = 0.75, n.trees = 500)
```

```
# Plot the result
Model_output_map <- Model_output$raster.prediction
crs(Model_output_map)<-"+proj=longlat + ellps=WGS84"

yy <- SOproj(Model_output_map)
plot(basemap)
plot(yy, col=my.palette.oranges, add=T)
```



Finally, you can evaluate your predictions with your test data. Binarize your predictions with the maxSSS threshold (see Tutorial #2/ Model outputs) and evaluate if your test data correctly fall into suitable areas.

```
maxSSS <- Model_output$eval.stats$maxSSS

# extract predictions at test data location
extracted_values <- raster::extract(Model_output_map,location_presence_test)
head(extracted_values)
```

```
## [1]        NA        NA 0.9197329 0.9239488        NA        NA
```

```
# compare values with the maxSSS value and evaluate the percentage of
# correctly classified presence test data
100* length (which(na.omit(extracted_values) >= maxSSS)) / length(na.omit(extracted_values))
```

```
## [1] 100
```

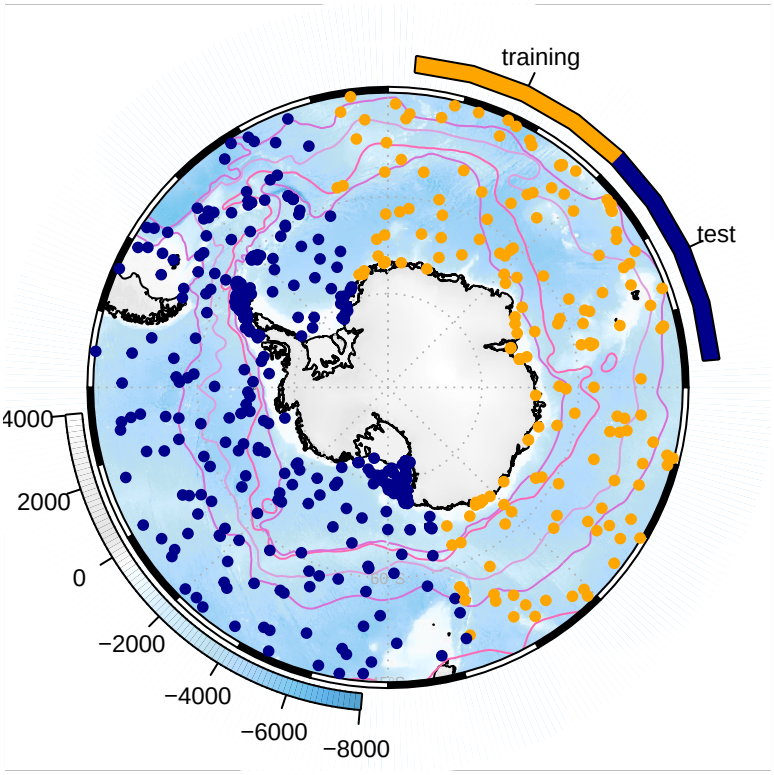
Generate a model with a spatial cross-validation

This time, you will make the training/test partition with a spatial condition. You can split your environment into 2, 3, 4, 5... areas that will contain either training or test data. See Guillaumot et al. (2019) for further details.

Example of partition into 2 areas

```
# Tag presence and background datasets with a new variable
idP <- which(SDMtable$id == 1) #  id of presence data

partition_function <- SDMPlay:::clock2(SDMtable[idP, c("longitude", "latitude")],
                                       SDMtable [-idP, c("longitude", "latitude")])

# Generate a variable that will contain the spatial splitting information (factor format)
MyFold <- rep(NA, nrow(SDMtable)) # training/test data will be labelled 1 or 2

MyFold[idP] <- partition_function$occ.grp # splitting within the presence data
MyFold[-idP] <- partition_function$bg.coords.grp # splitting within the background data
```

```
# Plot training and test data
basemap <- SOmap(bathy_legend= T, graticules= T, fronts= T, border_width= 0.8)
plot(basemap)
SOplot(SDMtable[,c(2,3)], col=c("orange","darkblue")[as.factor(MyFold)], pch=20)
SOleg(col=c("orange","darkblue"), position = "topright",
      tlabs = c("training","test"), type = "discrete")
```

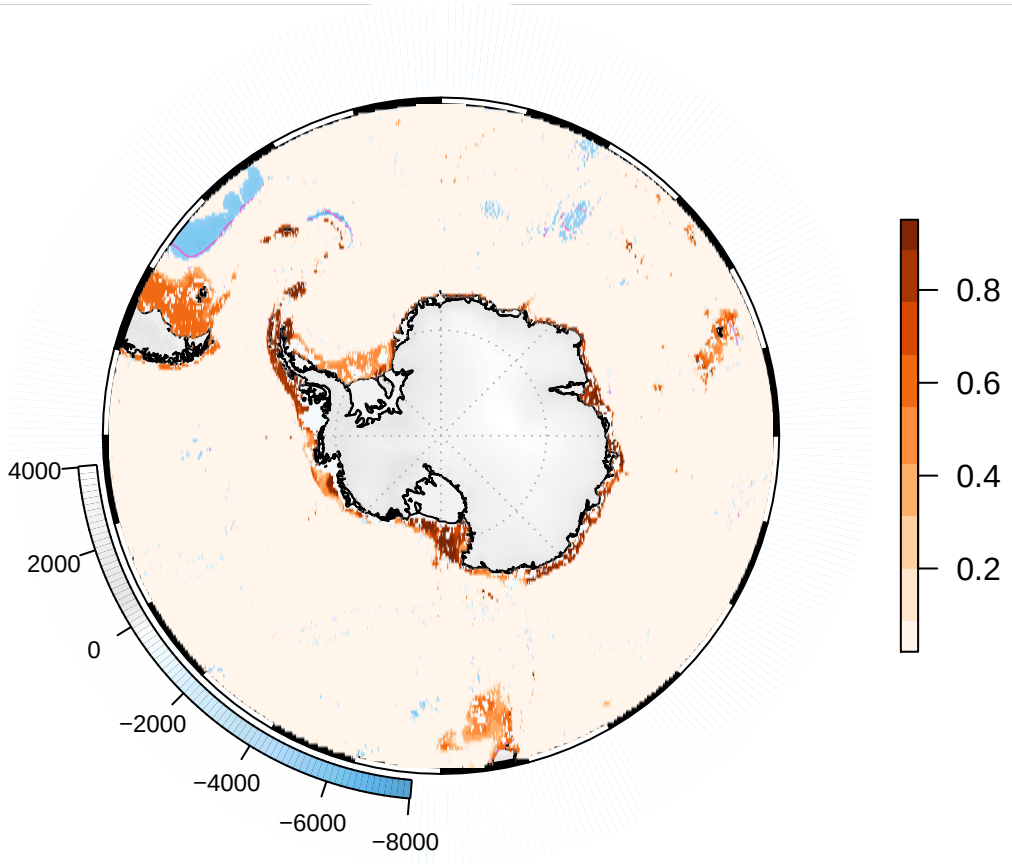


Notice that if you run the code several time, the partition changes: it depends on random selections. You can therefore run the code several time in a loop to make several splitting replicates (see one example below).

```
#Run your model with the splitting, fill the 2 new arguments (n.folds and fold.vector)
# that indicate to the function that you are using a spatial cross-validation procedure
```

```
Model_output <- SDMPlay:::compute.brt(x=SDMtable,
                                      proj.predictors=predictors_stack_SO,
                                      tc = 2, lr = 0.001, bf = 0.75, n.trees = 500,
                                      n.folds = 2,
                                      fold.vector = MyFold)
```

```
# Plot the result
Model_output_map <- Model_output$raster.prediction
crs(Model_output_map)<-"+proj=longlat + ellps=WGS84"

yy <- SOproj(Model_output_map)
plot(basemap)
plot(yy, col=my.palette.oranges, add=T)
```



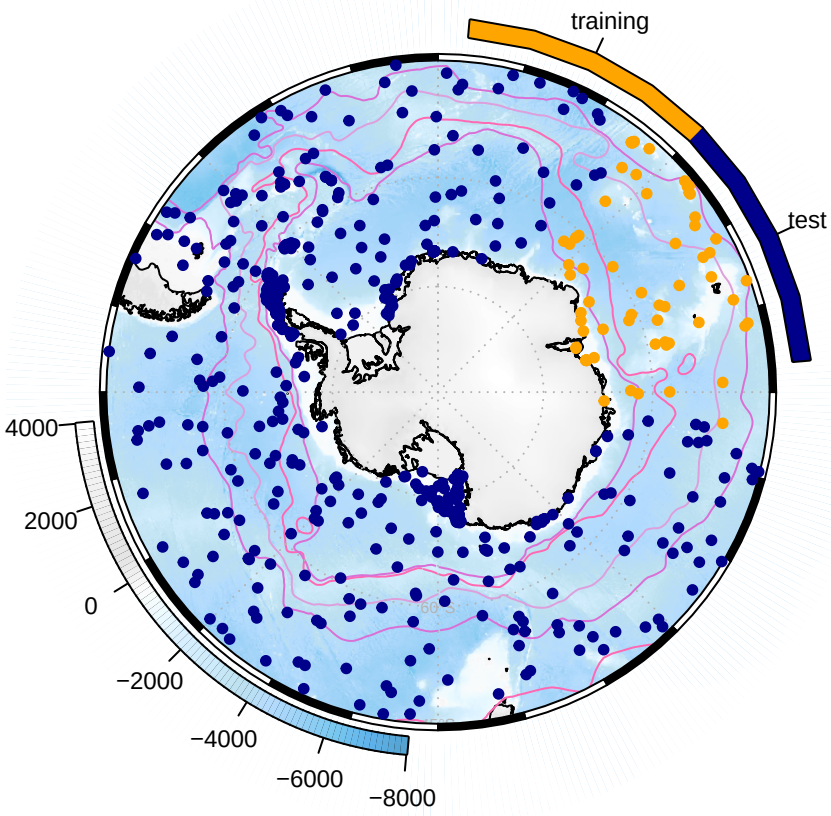

```
# Calculate evaluation scores
maxSSS <- Model_output$eval.stats$maxSSS

# extract predictions at test data location
location_presence_test <- SDMtable[as.factor(MyFold)==1,c(2,3)]
extracted_values <- raster::extract(Model_output_map,location_presence_test)
#extracted_values

# compare the values with the maxSSS value and evaluate the percentage
# of correctly classified presence test data
100* length (which(na.omit(extracted_values) >= maxSSS)) / length(na.omit(extracted_values))
```

```
## [1] 14
```

You can choose and play with 3, 4 or 6 areas !

Several functions to do so are provided in this package (clock3, clock4, clock6). Make some trials. Your results will be more robust with several replicates each time. Don't forget to also apply all the advices given in the previous tutorials (these were not applied here because of computering matters).

```
  idP <- which(SDMtable$id == 1)
  partition_function <- SDMPlay:::clock6(SDMtable[idP, c("longitude", "latitude")],
                                         SDMtable [-idP, c("longitude", "latitude")])

  MyFold <- rep(NA, nrow(SDMtable))
  MyFold[idP] <- partition_function$occ.grp
  MyFold[-idP] <- partition_function$bg.coords.grp


  basemap <- SOmap( bathy_legend = T, graticules = T, fronts = T, border_width = 0.8)
plot(basemap)
SOplot(SDMtable[,c(2,3)],
       col=c("orange","darkblue","darkblue","darkblue","darkblue","darkblue")[as.factor(MyFold)],
       pch=20)
SOleg(col=c("orange","darkblue"), position = "topright",
      tlabs = c("training","test"), type = "discrete")
```



The model will consider at each time that among the 6 areas, one will be used for training and the other

ones for testing. You need to manually transform the labelling of the MyFold if you want 2 or 3 out of the 6 areas to be used for training.

Example of a loop

```
# create an empty stack that you will fill with produced maps during the loop
stack.pred <- subset(predictors_stack_SO, 1);values(stack.pred) <- NA

nb_replicates <- 3
for (i in nb_replicates){
  idP <- which(SDMtable$id == 1)
  partition_function <- SDMPlay:::clock6(SDMtable[idP, c("longitude", "latitude")],
                                         SDMtable [-idP, c("longitude", "latitude")])

  MyFold <- rep(NA, nrow(SDMtable))
  MyFold[idP] <- partition_function$occ.grp
  MyFold[-idP] <- partition_function$bg.coords.grp

  Model_output <- SDMPlay:::compute.brt(x=SDMtable,
                                  proj.predictors=predictors_stack_SO,
                                  tc = 2, lr = 0.001, bf = 0.75, n.trees = 500,
                                  n.folds = 6,
                                  fold.vector = MyFold)

  stack.pred <- stack(stack.pred, Model_output$raster.prediction)
}
```

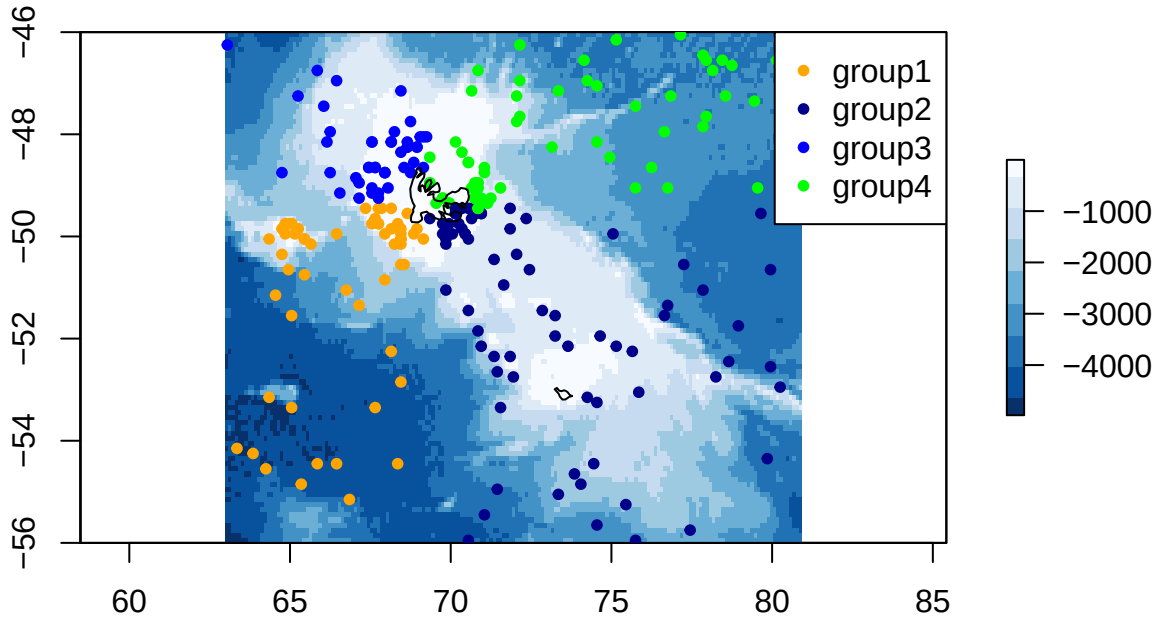
```
# at the end of the loop, calculate the average predictions of your 3 replicates

average_pred <- calc(stack.pred, mean)

# This is a raster layer, you can then plot it as previously !
```

Spatial cross-validation without a Southern Ocean circle shape

If you are working in another area than the entire Southern Ocean and you want to split your occurrence records without this "clock" shape, your can have a look at the ENMeval R package (Muscarella et al. 2014) and their proposed "get.block" function

```
library(SDMPlay)
library(ENMeval)
library(dismo)
library(raster)

data("ctenocidaris.nutrix")
data(predictors2005_2012)
ctenocidaris.nutrix.occ <- ctenocidaris.nutrix[,c(7,8)]

SDMtable_ctenocidaris <- SDMPlay:::SDMtab(xydata=ctenocidaris.nutrix.occ,
                                          predictors=predictors2005_2012,
                                          unique.data=FALSE,
                                          same=TRUE)
```

```
idP <- which(SDMtable_ctenocidaris$id == 1)
partition_function <- ENMeval::get.block(SDMtable_ctenocidaris[idP, c("longitude", "latitude")],
                                         SDMtable_ctenocidaris [-idP, c("longitude", "latitude")])

MyFold <- rep(NA, nrow(SDMtable_ctenocidaris))
MyFold[idP] <- partition_function$occ.grp
MyFold[-idP] <- partition_function$bg.grp

plot(subset(predictors2005_2012,1), col=my.palette.blue)
points(SDMtable_ctenocidaris[,c("longitude", "latitude")],
     col=c("orange","darkblue","blue","green")[as.factor(MyFold)],
     pch=20)
points(worldmap, type="l")
legend("topright", legend=c("group1","group2","group3","group4"),
       col=c("orange","darkblue","blue","green"),
       pch=20, bg = "white")
```



REFERENCES

Elith, J., Anderson, R., Dudík, M., Ferrier, S., Guisan, A., J Hijmans, R., Huettmann, F., ... & A Loiselle, B. (2006). Novel methods improve prediction of species' distributions from occurrence data. *Ecography*, 29(2), 129-151.

Elith, J. & Leathwick, J. R. (2009). Species distribution models: ecological explanation and prediction across space and time. *Annual Review of Ecology, Evolution, and Systematics*, 40, 677-697.

Guillaumot, C., Artois, J., Saucède, T., Demoustier, L., Moreau, C., Eléaume, M., ... & Danis, B. (2019). Broad-scale species distribution models applied to data-poor areas. *Progress in Oceanography*, 175, 198-207.

Guillaumot, C., Moreau, C., Danis, B. & Saucède, T. (2020). Extrapolation in species distribution modelling. Application to Southern Ocean marine species. *Progress in Oceanography*, 188, 102438.

Muscarella, R., Galante, P.J., Soley-Guardia, M., Boria, R.A., Kass, J.M., Uriarte, M. & Anderson, R.P. (2014). ENMeval: An R package for conducting spatially independent evaluations and estimating optimal model complexity for Maxent ecological niche models. *Methods in Ecology and Evolution*, 5(11), 1198-1205.

Pearson, R.G. (2007). Species' distribution modeling for conservation educators and practitioners. *Synthesis. American Museum of Natural History*, 50.