Supplementary Material S2: Case study for "A theoretical framework for upscaling species distribution models"

Christine N. Meynard, Cyril Piou, David M. Kaplan

Contents

Note that this pdf was generated using an R Markdown script that can be found in the github repository (see main text). Here you will find a simple summary of the aggregation and modelling stages for the desert locust case study, and some additional figures in the end, but in order to see the code you need to recover the rmarkdown file from github. Also note that if you run this on your personal computer, it will generate around 13GB of data.

S2.1 Load base data

Load the different source datasets (PC1, PC2, and occurrence grid at finest resolution).

S2.1.1 Upscaling (spatial aggregation)

Here we will aggregate spatial data by grid cells, from the native 1 x 1 grid cell (no aggregation) to the largest resolution of 19 x 19 grid cells, by intervals of 2 grid cells: 1 x 1, 3 x 3, 5 x 5, etc. For occurrences, the output in the aggregate is the maximum value, so that for any number of presences, the larger aggregate receives a presence. For the environmental predictors (PC1 and PC2), the output at the larger aggregate is the average of the values of the individual grid cells.

S2.1.2 Create data frames

This will just create the global dataframes that are required for modelling at each scale.

S2.1.3 Pre-calculate splits of the data

This will take the previously created global dataframes and generate the specified number of splits for each scale, so that a portion can be used for calibration purposes (80%), and the rest can be used for model validation (i.e. calculating classification rate success).

S2.1.4 Save base data

This saves the objects externally so that the process can be restarted later from here if needed.

S2.2 Create list of models to run

This serves to create a list of output files. Also, if the files already exist (because the script was run before), it will allow saving some time by skipping later the generation of these files.

S2.3 Run models

S2.3.1 Helper functions from Valavi paper

Here we adapted some of the functions presented in the supplements of Valavi et al (2022) Ecological Monographs 92 (1): e01486. Most notably, here we used the library stars to replace the raster and other dependencies that will go unsupported soon. The functions that we recovered from Valavi involve optimizing model parameters for MAXENT, and transforming predictors into quadratic for the glmnet implementation of lasso. The other prediction functions (see next section) work quite differently in stars, so we did not use Valavi's version, though we did use the same model parameter options in general.

S2.3.2 Functions to fit models

This section contains code to run the different models with the options we want them all to be run. As stated before, these follow the implementation in Valavi: use of weights to balance presences and absences (notice here we have real absences though, whereas Valavi was using background data); optimizing maxent parameters; using quadratic terms for lasso; using the downsampled version of random forests.

S2.3.3 Functions to predict models

These are just adapting the different predictors in the projection area so that we can predict with the models that were calibrated in the previous section.

S2.3.4 Execute models

Here we execute the previous models. The script is set to run in a cluster, but will work in any computer with several cores, as there is some parallelization of procedures.

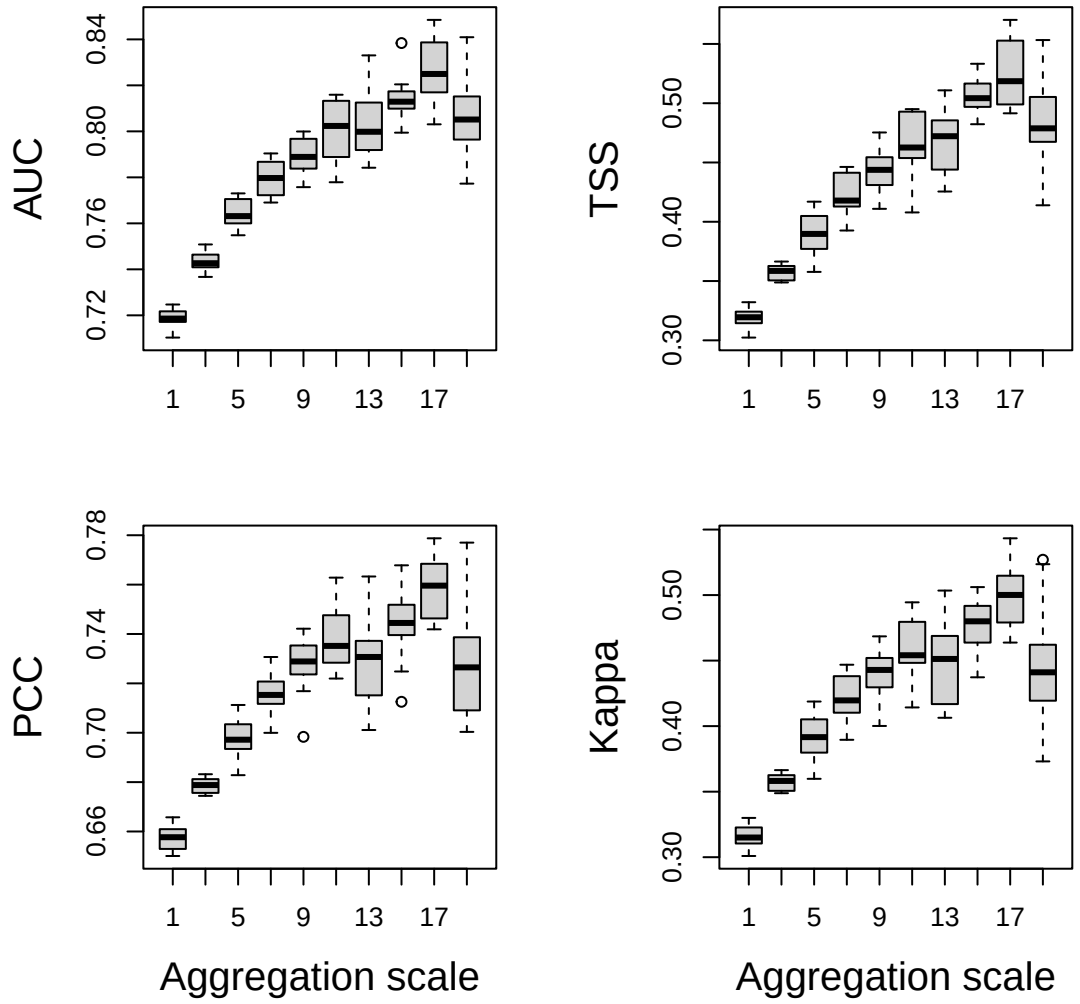
S2.4 Ensemble mean and performance statistics

Here we calculate different performance statistics, especially around presence-absence classification rates. We also generate the necessary predictions to draw the functional responses for each model and for the ensemble (i.e. need the predictions for PC1 while PC2 remains constant, and vice versa).

```
## Le chargement a nécessité le package : rJava
```

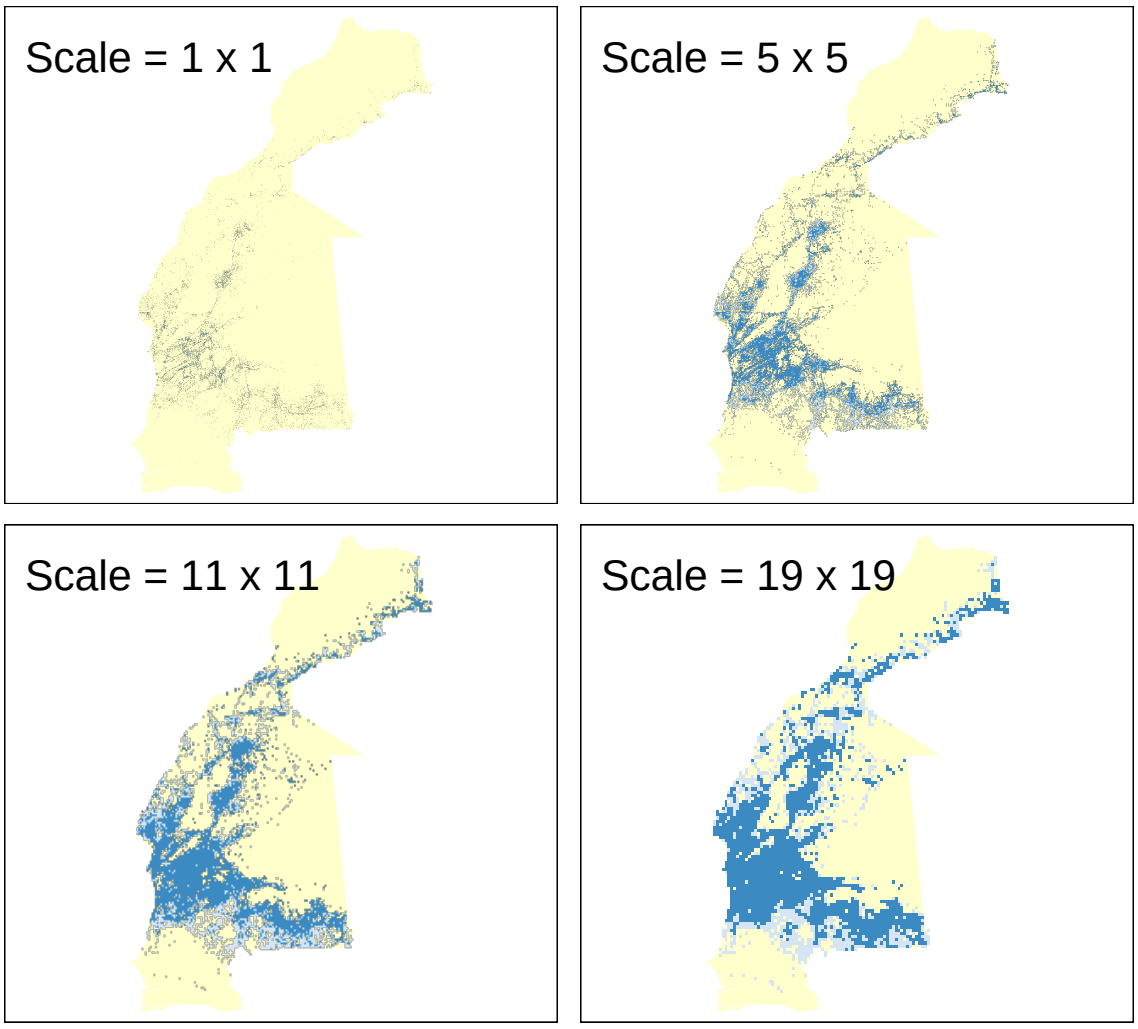
After all this, we just need some plots to summarize the results. First recover all the performance indices of interest per scale in a more workable format.

Then do the plot of different presence-absence performance indices per aggregation scale.



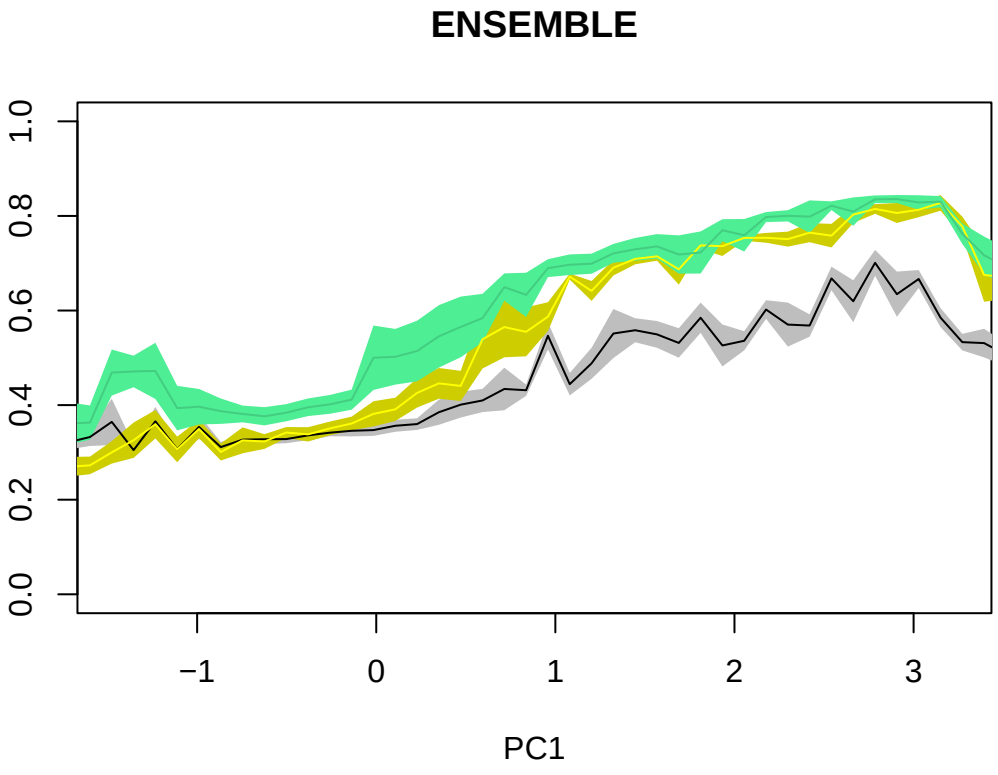

```
## Reading layer 'StArea' from data source 'D:\Meynard_et_al_ScalingSDMs_V1\rawData' using driver 'ESRI
## Simple feature collection with 5 features and 4 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -17.53278 ymin: 12.30175 xmax: -1.011806 ymax: 35.91917
## CRS:           NA
```

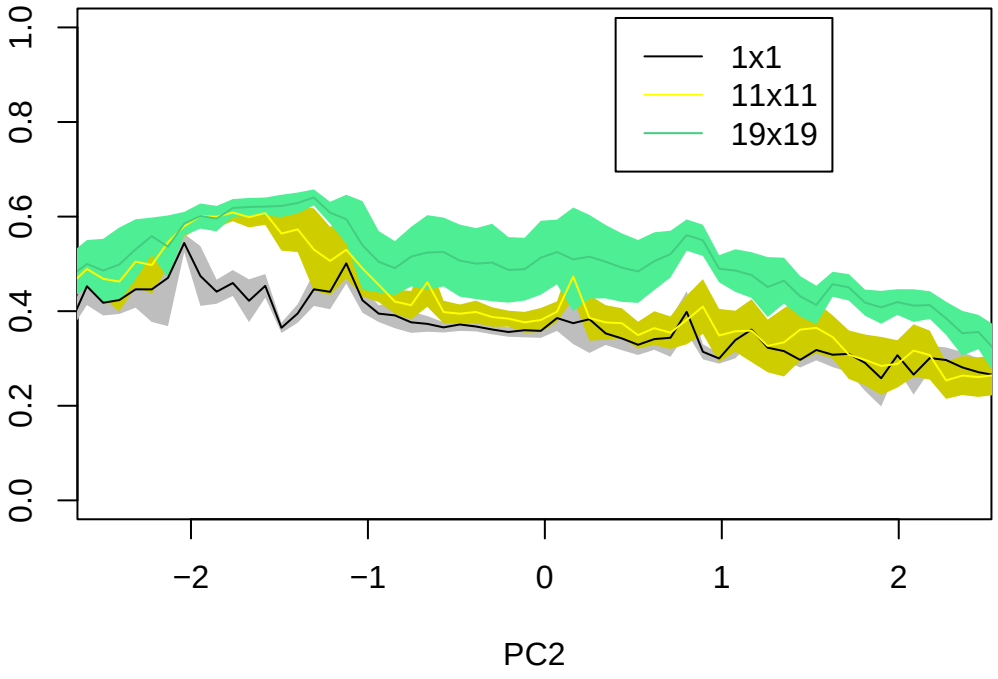
```
## stars object downsampled to 837 by 1195 cells. See tm_shape manual (argument raster.downsample)
```



Now in order to make the plots for response curves, we need to recover the predictions of the ensemble and of the different models when PC1 varies and PC2 remain constant, and vice versa. Also, because when we previously extracted the range of PC1 and PC2 values we did so from all of the study area, now we need to restrict it to the range that was actually surveyed. We will actually limit it to the 95% central values to avoid extreme values.

And draw the response curves:


ENSEMBLE
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
2
3
PC1

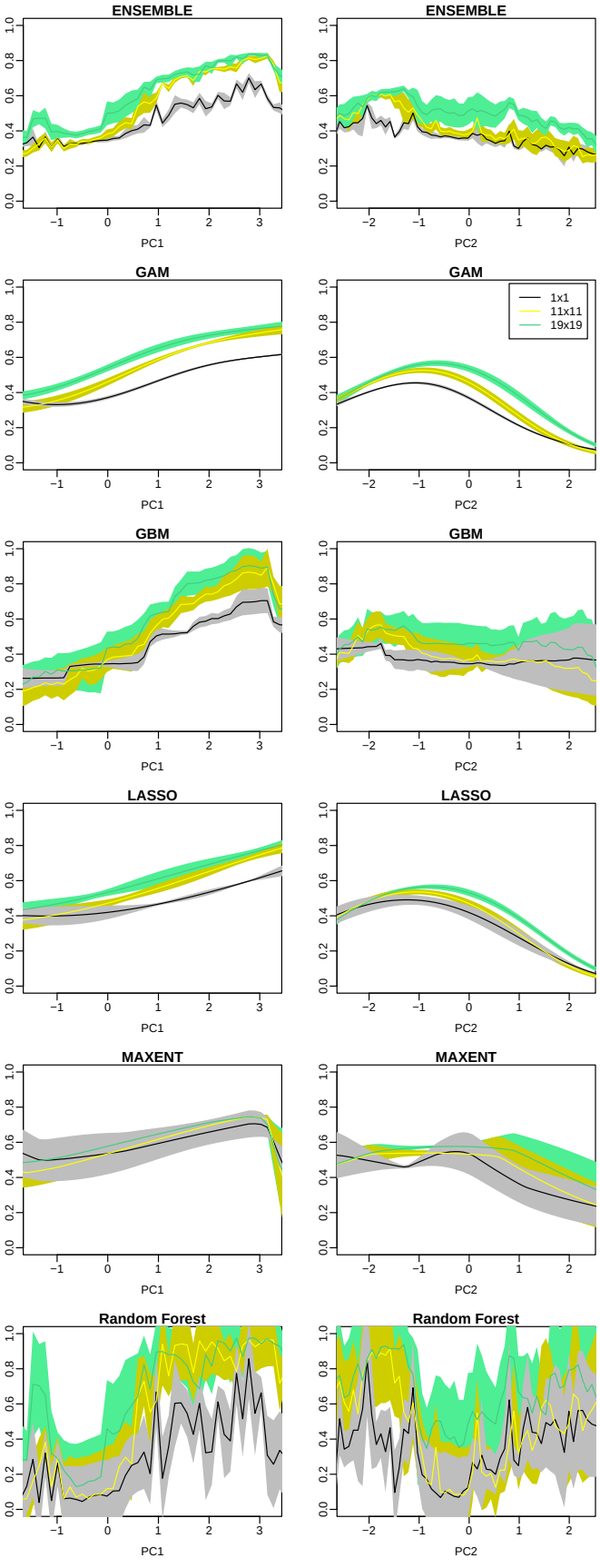

1x1
11x11
19x19
1.0
0.8
0.6
0.4
0.2
0.0
−2
−1
0
1
2
PC2

83

We

would like to see how the GAM response curves look like (probably smoother than the ensemble); and since we are at it, looking at individual response curves for each type of model would also be interesting, so we'll recover the data for each model below.

And draw a single figure with all response curves for each model type + the ensemble.Notice that Random Forests produce a lot of variability. This is what is driving the ensemble's ups and downs.


ENSEMBLE
ENSEMBLE
GAM
GAM
1x1
11x11
19x19
GBM
GBM
LASSO
LASSO
MAXENT
MAXENT
Random Forest
Random Forest
PC1
PC2

89