TESTING

This is simply for using the self-created functions in the 'Functions.R'script.

```
source("Functions.R")
#Mainly used for loading all models into memory
restore.session("TESTING.Rda")
```

```
## Loading all data...
## Loading packages...
```

```
## Registered S3 method overwritten by 'cli':
##   method     from
##   print.tree tree
```

```
## rstan (Version 2.19.2, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
## For improved execution time, we recommend calling
## Sys.setenv(LOCAL_CPPFLAGS = '-march=native')
## although this causes Stan to throw an error on a few processors.
```

```
##
## Attaching package: 'rstan'
```

```
## The following object is masked from 'package:raster':
##
##     extract
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday,
##     week, yday, year
```

```
## The following object is masked from 'package:raster':
##
##     shift
```

```
## Restoring search path...
```

```
## Error in attach(`NA`): object 'NA' not found
```

1 Testing methods for generating Psuedo-Abscene points

```
#Load relevant models. Random forest model using 100 trees and 5 k-fold validation

#P = only use prescene and letting model generate PA using its recommended
# stragety, BG = our background data assumption , PA = psuedo points that are
# very close to observation points using function in spatialEco (https://rdrr.io/cran/spatialEco/man/ps
```

```r
#Will be loaded into memory directly by restoring the session
# A.RF_PA = readRDS("Agile_RF_PA")
# A.RF_P = readRDS("Agile_RF_P")
# A.RF_BG = readRDS("Agile_RF_BG")


slot(A.RF_PA, "evaluation")
```

```
##   threshold       AUC omission.rate sensitivity specificity prop.correct
## 1      0.66 0.9077236     0.0923018   0.9076982   0.9077491    0.9077228
##       Kappa
## 1 0.8152711
```

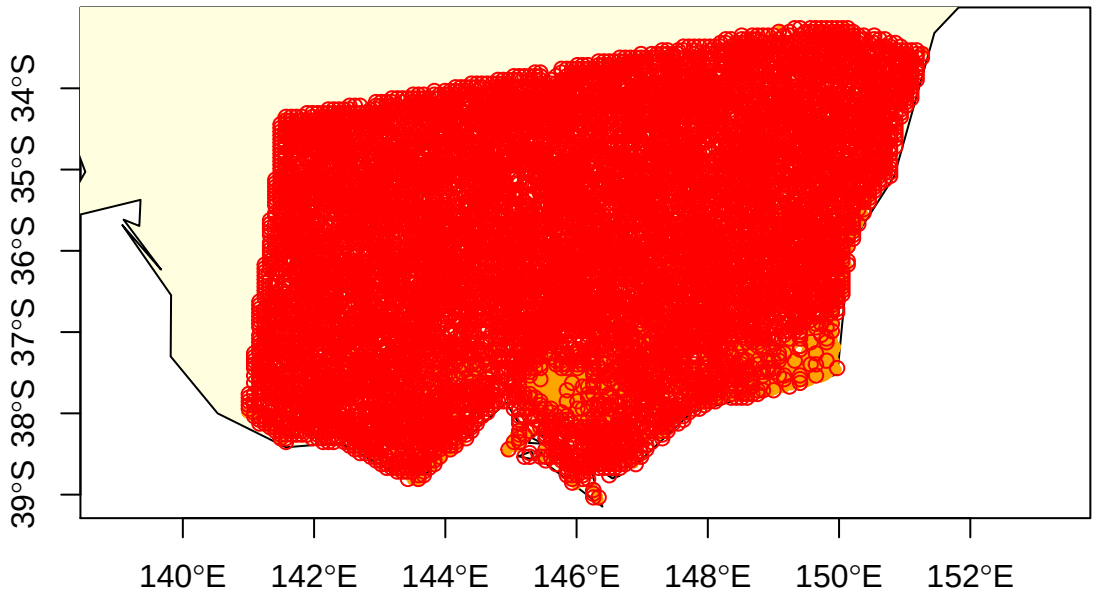
```r
slot(A.RF_P, "evaluation")
```

```
##   threshold       AUC omission.rate sensitivity specificity prop.correct
## 1    0.8235 0.9827652    0.01723478   0.9827652   0.9827652    0.9827652
##       Kappa
## 1 0.9655304
```

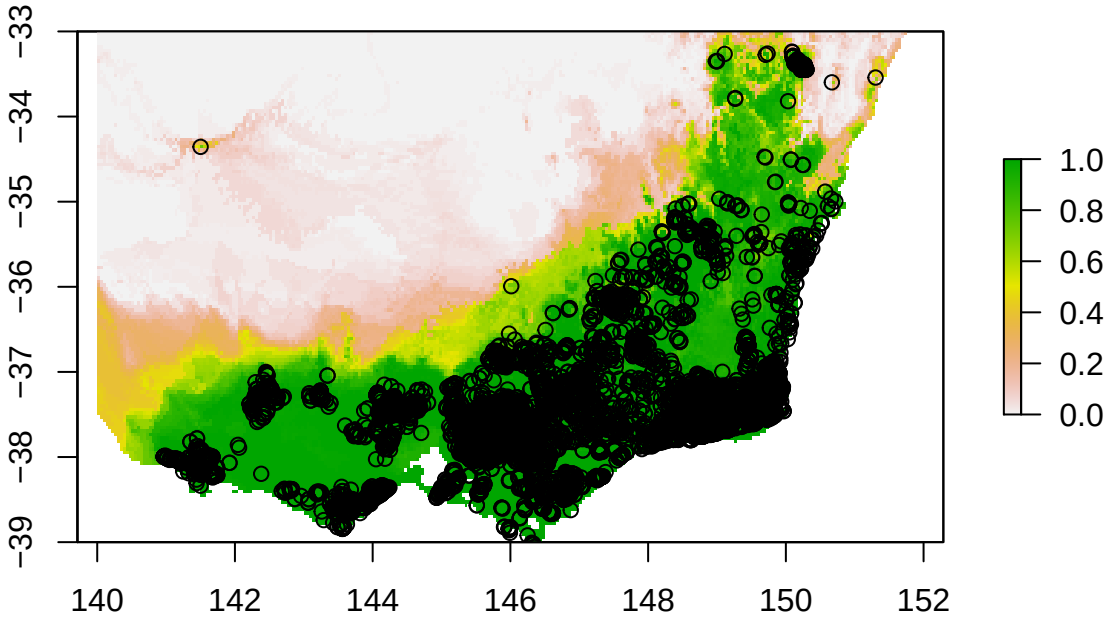
```r
slot(A.RF_BG, "evaluation")
```

```
##   threshold       AUC omission.rate sensitivity specificity prop.correct
## 1    0.5095 0.9992221    0.00076599    0.999234   0.9992101    0.9992222
##       Kappa
## 1 0.9984441
```

```r
#visualize the points. Red = psuedo abscene, orange = prescene
data("wrld_simpl")
plot_(slot(A.RF_PA,"data")[slot(A.RF_PA,"data")$Presence == 1,], slot(A.RF_PA,"data")[slot(A.RF_PA,"data
```




```
plot(slot(A.RF_BG, "projection"),  xlim = c(140,152), ylim = c(-39,-33))
points(slot(A.RF_P,"data")[slot(A.RF_P,"data")$Presence == 1,])
```



2 Evaluating the peformance various modelling techniques

```
#Not necessary anymore, session, including all models, is saved and can be
#loaded into memory

# A.RF = readRDS("Agile_RF_P")
# A.MAXENT = readRDS("Agile_MAXENT_P")
# A.GBM = readRDS("Agile_GBM_P")
# A.GLM = readRDS("Agile_GLM_P")
# A.GAM = readRDS("Agile_GAM_P")
# A.ANN = readRDS("Agile_ANN_P")

A.ALL = ensemble(A.RF,A.MAXENT,A.GBM,A.GLM,A.GAM,A.ANN)
```

```
## Creation of one ensemble niche model by algorithm...   done.
## Projection, evaluation and variable importance computing...   done
## Model evaluation...   done
## Axes evaluation...   done
## Algorithms correlation...   done
## uncertainty mapping...   done
## Algorithms evaluation...   done
```

```
A.ALL2 = ensemble(A.RF,A.MAXENT,A.GBM,A.GAM,A.ANN)
```

```
## Creation of one ensemble niche model by algorithm...   done.
## Projection, evaluation and variable importance computing...   done
## Model evaluation...   done
## Axes evaluation...   done
```

```
## Algorithms correlation...   done
## uncertainty mapping...   done
## Algorithms evaluation...   done
```

```
A.ALL3 = ensemble(A.RF,A.MAXENT,A.GAM,A.ANN)
```

```
## Creation of one ensemble niche model by algorithm...   done.
## Projection, evaluation and variable importance computing...   done
## Model evaluation...   done
## Axes evaluation...   done
## Algorithms correlation...   done
## uncertainty mapping...   done
## Algorithms evaluation...   done
```

```
A.ALL4 = ensemble(A.RF,A.GAM,A.ANN)
```

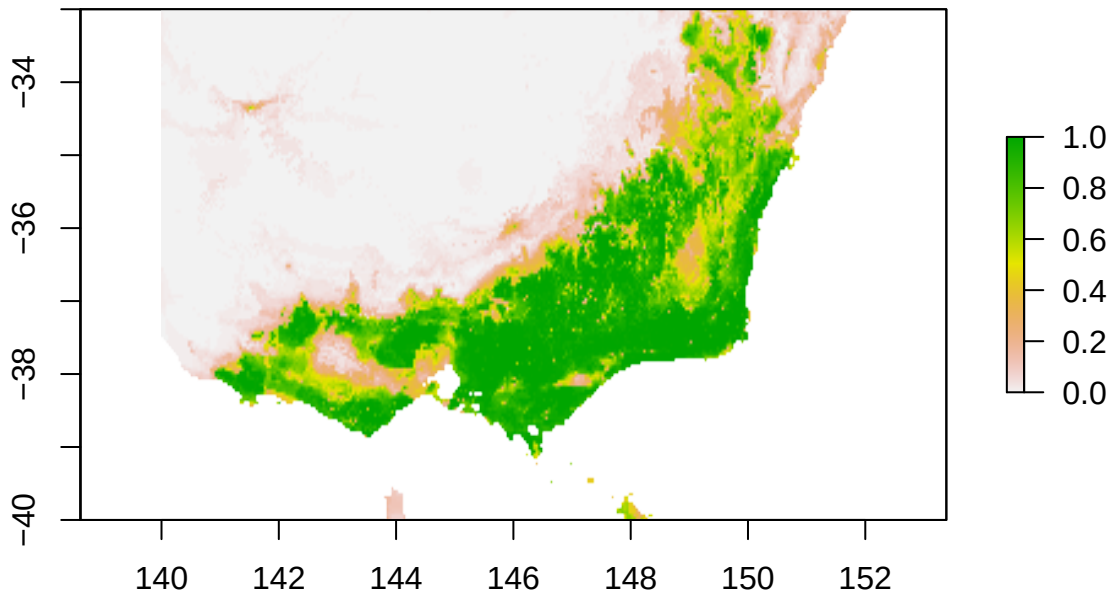
```
## Creation of one ensemble niche model by algorithm...   done.
## Projection, evaluation and variable importance computing...   done
## Model evaluation...   done
## Axes evaluation...   done
## Algorithms correlation...   done
## uncertainty mapping...   done
## Algorithms evaluation...   done
```

```
A.ALL5 = ensemble(A.RF,A.ANN)
```

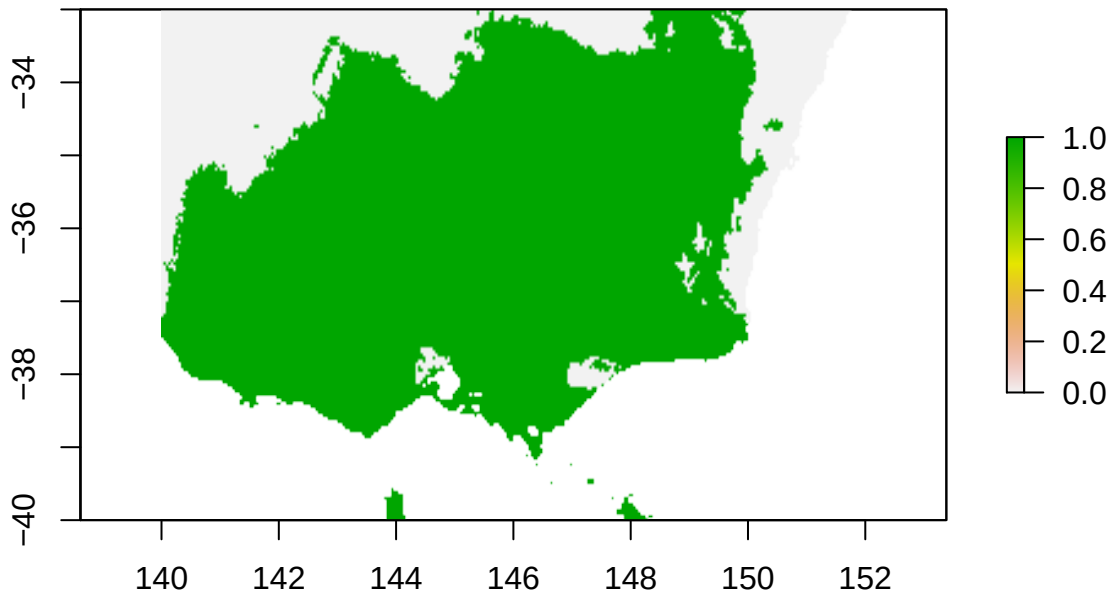
```
## Creation of one ensemble niche model by algorithm...   done.
## Projection, evaluation and variable importance computing...   done
## Model evaluation...   done
## Axes evaluation...   done
## Algorithms correlation...   done
## uncertainty mapping...   done
## Algorithms evaluation...   done
```

```
model_info = c("Random forest (RF)",
               "Maximum entropy (MAXENT)",
               "Generalized boosted regressions model (GBM)",
               "Generalized linear model (GLM)",
               "Generalized additive mode (GAM)",
               "Artificial neural network (ANN)",
               "ALL",
               "RF + MAXENT + GBM + GAM + ANN",
               "RF + MAXENT + GAM + ANN",
               "RF + GAM + ANN",
               "RF + ANN")


RF = slot(A.RF, "evaluation")[,c(2,4,5)]
MAXENT = slot(A.MAXENT, "evaluation")[,c(2,4,5)]
GBM = slot(A.GBM, "evaluation")[,c(2,4,5)]
GLM = slot(A.GLM, "evaluation")[,c(2,4,5)]
GAM = slot(A.GAM, "evaluation")[,c(2,4,5)]
ANN = slot(A.ANN, "evaluation")[,c(2,4,5)]
ALL = slot(A.ALL, "evaluation")[,c(2,4,5)]
ALL2 = slot(A.ALL2, "evaluation")[,c(2,4,5)]
ALL3 = slot(A.ALL3, "evaluation")[,c(2,4,5)]
ALL4 = slot(A.ALL4, "evaluation")[,c(2,4,5)]
```

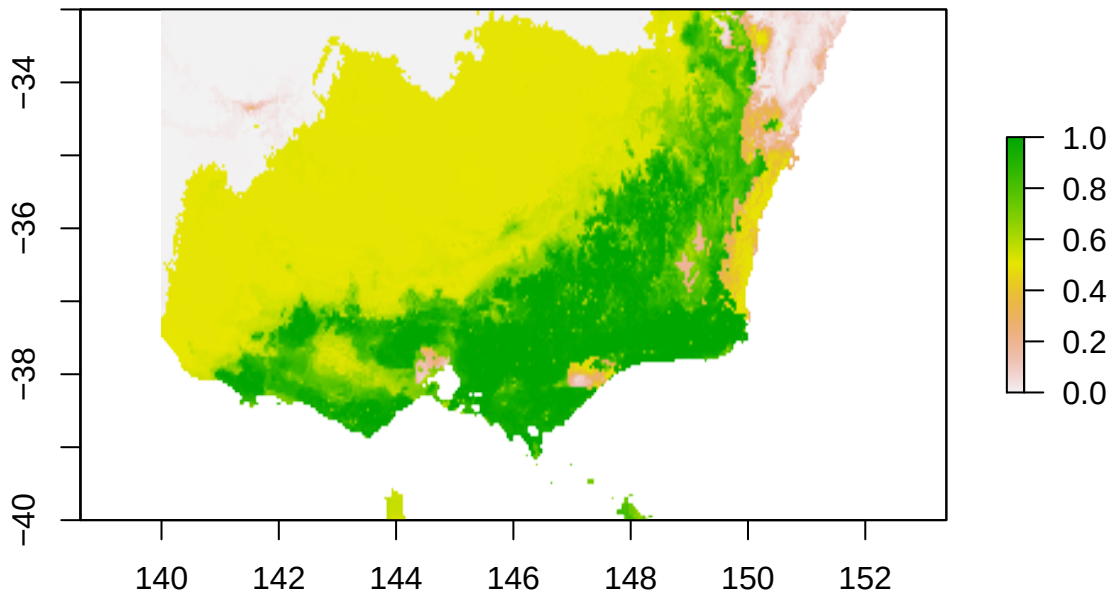
```
ALL5 = slot(A.ALL5, "evaluation")[,c(2,4,5)]

#Bind resultws into a dataframe and then add model_info column
View(data.frame(cbind( model_info,rbind(RF,MAXENT,GBM,GLM,GAM,ANN, ALL, ALL2,
                                        ALL3, ALL4, ALL5))))

plot(slot(A.RF, "projection"), xlim = c(140,152), ylim = c(-40,-33))
```




```
plot(slot(A.ANN, "projection"), xlim = c(140,152), ylim = c(-40,-33))
```




```
plot(slot(A.ALL5, "projection"), xlim = c(140,152), ylim = c(-40,-33))
```




```
#Save everything into memory, so models
```

3 Testing the final models

```
#This final model should be in memory from the restore.session at the start
#Dataframe containing species, and mode metrics such as AUC, sensitivity, and
#specificity
metrics = cbind(slot(slot(SSDM, "enms")[[1]], "name"),slot(slot(SSDM, "enms")
                                                           [[1]], "evaluation"))

colnames(metrics)[1]
```

```
## [1] "slot(slot(SSDM, \"enms\")[[1]], \"name\")"
```

```
for (i in 2:6) {

  metrics = bind_rows(metrics, cbind(slot(slot(SSDM, "enms")[[i]], "name"),
                                     slot(slot(SSDM, "enms")[[i]], "evaluation")))
}
```

```
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
```

```
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

```
metrics = metrics[, c(3, 5, 6,9)]
metrics = metrics[, c(4,1,2,3)]
metrics[1,1] = "White-browed Treecreeper"


colnames(metrics)[1] = "Species"

metrics
```

```
##                    Species       AUC sensitivity specificity
## 1 White-browed Treecreeper 0.8958866   0.8956274   0.8961458
## 2 Southern Brown Tree Frog 0.9768876   0.9768835   0.9768918
## 3       Small Triggerplant 0.9533278   0.9532713   0.9533843
## 4       Common Beard-heath 0.9794707   0.9794665   0.9794748
## 5         Agile antechinus 0.9880505   0.9879344   0.9881665
## 6        Brown Treecreeper 0.9756574   0.9756574   0.9756574
```