Mapping the area of applicability - Case Study

Supplementary to the paper ‘Predicting into unknown space? Estimating the area of applicability of spatial prediction models’

Hanna Meyer

1/4/2020

Introduction

This script contains the complete code for the case study within the paper and creates the figures presented there. It can further be used to run experiments under different settings. Note that running the code takes a few minutes!

Getting started

Major functionality needed is the ‘aoa’ function from the ‘CAST’ package that is doing the distance based estimation of the area of applicability. ‘Caret’ is needed for model training. The case study uses a simulated prediction task based on the ‘virtualspecies’ package.

```r
rm(list=ls())
#install_github("HannaMeyer/CAST")
library(virtualspecies)
library(caret)
library(CAST)
library(viridis)
library(gridExtra)
library(parallel)
library(knitr)
library(grid)
library(latticeExtra)
library(hydroGOF)
```

Settings for generating the predictors and response need to be defined, as well as the number of training data points and the seed used for all functions that involve randomness. The settings specified here are used for the case study published in the paper. Feel free to change them to see how things work under different scenarios!

```r
npoints <- 50 # number of training samples
clustered <- FALSE #TRUE clustered design
nlusters <- 25 #number of clusters if design=clustered
maxdist <- 0.7 #maxdist for clustered samples if design=clustered
meansPCA <- c(3, -1) # means of the gaussian response functions to the 2 axes
sdPCA <- c(2, 2) # sd's of the gaussian response functions to the 2 axes
simulateResponse <- c("bio2","bio5","bio10", "bio13","bio14","bio19") # variables used to simulate the
studyarea <- c(-15, 65, 30, 75) # extent of study area. Default: Europe
seed <- 10
```

Get data

Bioclim data are downloaded and cropped to the study area.

```
predictors_global <- getData('worldclim', var='bio', res=10, path='../data/')
wp <- extent(studyarea)
predictors <- crop(predictors_global,wp)

#create a mask for land area:
mask <- predictors[[1]]
values(mask)[!is.na(values(mask))] <- 1
```

Generate Predictors and Response

The virtual response variable is created based on the PCA of a subset of bioclim predictors. See the virtualspecies package for further information.

```
response <- generateSpFromPCA(predictors[[simulateResponse]],
means = meansPCA,sds = sdPCA, plot=F)$suitab.raster
```

Simulate training points

To simulate field locations that are typically used as training data, "npoints" locations are randomly selected. If a clustered design is used, the "npoints" are distributed over "nclusters" with a maximum distance between each point of a cluster (maxdist, in degrees).

```
mask <- rasterToPolygons(mask,dissolve=TRUE)
set.seed(seed)
if (clustered){
  samplepoints <- csample(mask,npoints,nlusters,maxdist=maxdist,seed=seed)


}else{
  samplepoints <- spsample(mask,npoints,"random")
}
```

Model training and prediction

To prepare model training, predictor variables are extracted for the location of the selected sample data locations.

```
trainDat <- extract(predictors,samplepoints,df=TRUE)
trainDat$response <- extract (response,samplepoints)

if (clustered){
  trainDat <- merge(trainDat,samplepoints,by.x="ID",by.y="ID")
}

trainDat <- trainDat[complete.cases(trainDat),]
```

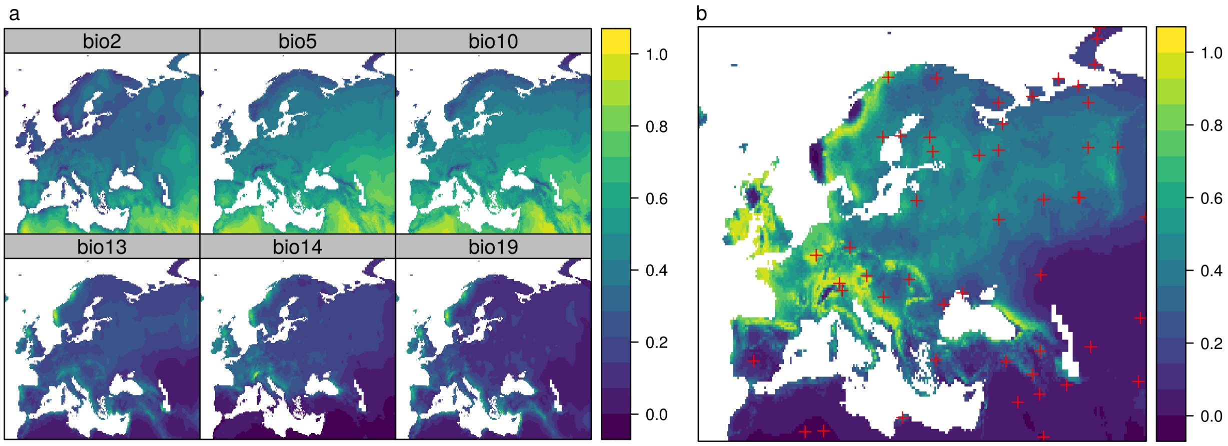

Figure 1: Subset of the predictors as well as the response variable and the selected training data points

Model training is then done using the caret package. Note that other packages work as well as long as variable importance can be derived. The model output gives information on the general estimated model performance based on random cross validation.

```
set.seed(seed)
model <- train(trainDat[,names(predictors)],trainDat$response,
               method="rf",importance=TRUE,tuneGrid = expand.grid(mtry = c(2:length(names(predictors)))
               trControl = trainControl(method="cv"))
print(model)
```

```
## Random Forest
##
## 50 samples
## 19 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 46, 45, 45, 45, 46, 46, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE        Rsquared   MAE
##    2    0.08312237  0.9449585  0.06038504
##    3    0.08134313  0.9491834  0.05831224
##    4    0.08270354  0.9500045  0.05912983
##    5    0.08255321  0.9448124  0.05969416
##    6    0.08354799  0.9466461  0.06076259
##    7    0.08258087  0.9483848  0.05983775
##    8    0.08358856  0.9414773  0.06077974
##    9    0.08285660  0.9471475  0.06024775
##   10    0.08549180  0.9417758  0.06160476
##   11    0.08714468  0.9394383  0.06346486
##   12    0.08701203  0.9336757  0.06361495
##   13    0.08738857  0.9354565  0.06383382
##   14    0.08733814  0.9365332  0.06300664
##   15    0.08859858  0.9324766  0.06452674
##   16    0.08794901  0.9354402  0.06460736
```

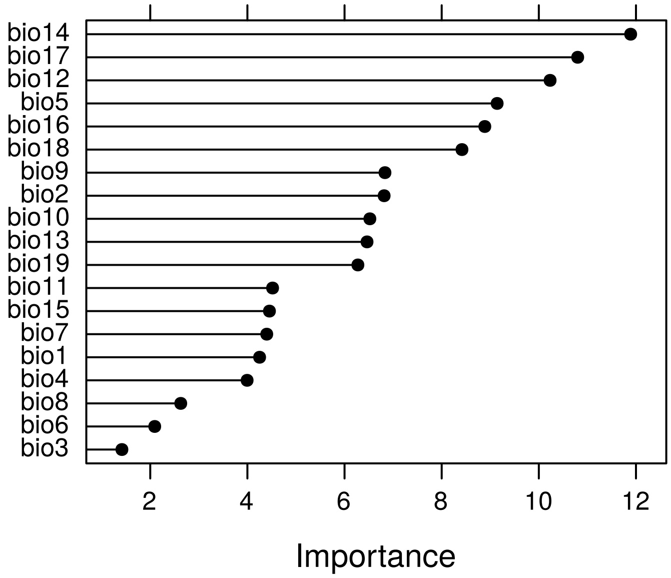

Figure 2: Variable importance

```
##   17    0.08750610  0.9333760  0.06407203
##   18    0.08907661  0.9320242  0.06453999
##   19    0.08606493  0.9362805  0.06282772
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 3.
```

Prediction and error calculation

The trained model is used to make predictions for the entire study area. The absolute error between prediction and reference is calculated for later comparison with the area of applicability index.

```
prediction <- predict(predictors,model)
truediff <- abs(prediction-response)
```

Estimating the area of applicability

The variable importance from model training can be visualized to get information on how variable weighting will be done during to estimation of the area of applicability index

The area of applicability and the applicability index are then calculated. First using weighted variables, and second (for comparison) without weighting. Everything is run in parallel to speed things up.

```
if(clustered){
  clstr <- trainDat$clstrID
}else{
  clstr <- NULL
  }
```

```
cl <- makeCluster(detectCores()-1)
#with variable weighting:
AOA <- aoa(trainDat,predictors, variables = names(predictors),model=model,cl=cl,clstr=clstr)
#without weighting:
AOA_noWeights <- aoa(trainDat,predictors, variables = names(predictors),cl=cl,clstr=clstr)

stopCluster(cl)
```

Standard deviation from individual trees for comparison

For camparison to what is often used as uncertainty information, the standard deviations of the individual predictions from the 500 developed trees within the Random Forest model are calculated.

```
predsd <- RFsd(predictors,model)
```

Comparison

The Area of Applicability Index, as well as the standard deviations can then be compared to the true error.

```
compare <- stack(response,prediction,
              predsd,truediff,
                 AOA$AOAI)
names(compare) <- c("response","prediction", "sd","true_diff","AOAI")
summary(values(compare))
```

```
##     response       prediction           sd          true_diff
##  Min.   :0.00    Min.   :0.00    Min.   :0.01    Min.   :0.00
##  1st Qu.:0.07    1st Qu.:0.14    1st Qu.:0.06    1st Qu.:0.01
##  Median :0.32    Median :0.34    Median :0.08    Median :0.04
##  Mean   :0.31    Mean   :0.32    Mean   :0.10    Mean   :0.06
##  3rd Qu.:0.45    3rd Qu.:0.44    3rd Qu.:0.15    3rd Qu.:0.07
##  Max.   :1.00    Max.   :0.81    Max.   :0.31    Max.   :0.76
##  NA's   :47804   NA's   :47804   NA's   :47804   NA's   :47804
##       AOAI
##  Min.   :-2.89
##  1st Qu.:-0.34
##  Median :-0.20
##  Mean   :-0.25
##  3rd Qu.:-0.12
##  Max.   : 0.00
##  NA's   :47804
```

Relationship with the true error

The general relationship is then visualized via scatterplots, linear models between the true error and the AOAI and RMSE calculation

```
#AOAI with weights:
summary(lm(values(truediff)~values(AOA$AOAI)))$r.squared
```

```
## [1] 0.5111739
```

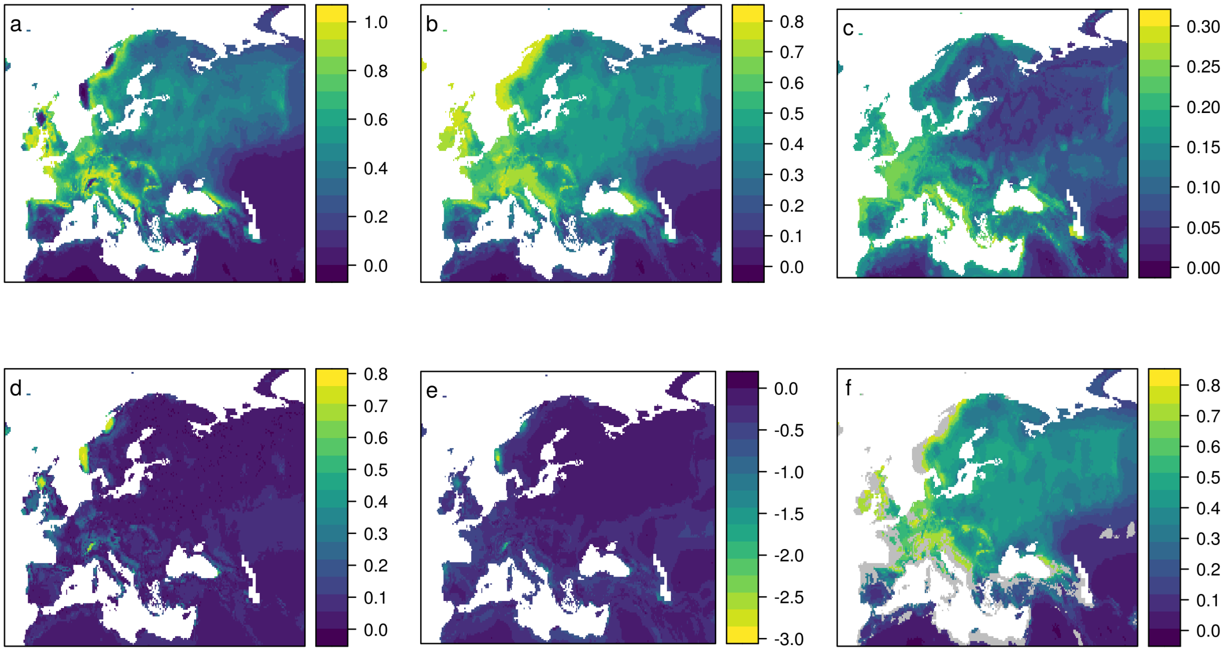

Figure 3: Comparison between reference (a), prediction (b), standard deviation of predictions (c), the true error (d), AOAI based on weighted variables (e), masked predictions (f)

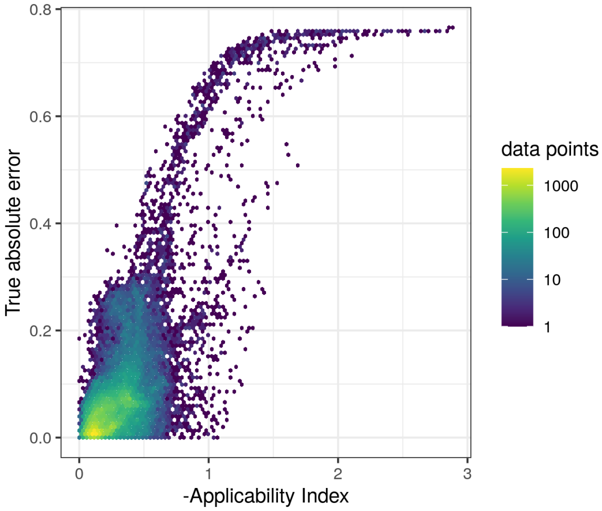

Figure 4: Relationship between the AOAI and the true error

```r
#AOAI no weights:
summary(lm(values(truediff)~values(AOA_noWeights$AOAI)))$r.squared
```

```
## [1] 0.3822119
```

```r
#comparison prediction~ref
summary(lm(values(response)~values(prediction)))$r.squared
```

```
## [1] 0.8564091
```

```r
rmse(values(response),values(prediction))
```

```
## [1] 0.09536494
```

```r
#comparison prediction for the AOA~ref
print(attributes(AOA)$aoa_stats)
```

```
## $AvrgMean_train
## [1] 37.62449
##
## $AvrgMin_train
## [1] 9.370358
##
## $SdMin_train
## [1] 6.707771
##
## $threshold
## [1] -0.4273315
```

```r
predictionAOI <- prediction
values(predictionAOI)[values(AOA$AOA)==0] <- NA
summary(lm(values(response)~values(predictionAOI)))$r.squared
```

```
## [1] 0.9545497
```

```r
rmse(values(response),values(predictionAOI))
```

```
## [1] 0.06047159
```

```r
# ...and for outside the AOA
predictionNOTAOI <- prediction
values(predictionNOTAOI)[values(AOA$AOA)==1] <- NA
summary(lm(values(response)~values(predictionNOTAOI)))$r.squared
```

```
## [1] 0.503903
```

```r
rmse(values(response),values(predictionNOTAOI))
```

```
## [1] 0.2214552
```