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 (depending on the number of training data and the size of the study area)!

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.

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

rmse <- function(pred,obs){sqrt( mean((pred - obs)^2, na.rm = TRUE) )}
```

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!

Getting started

Note: To reproduce results used in the paper use the following settings:

- Default Case study with random data: npoints=50,design="random",meansPCA= c(3, -1), sdPCA=c(2, 2),simulateResponse=c("bio2","bio5","bio10", "bio13","bio14","bio19"),studyarea=c(-15, 65, 30, 75), seed=10

- Case Study with clustered data: npoints=500,nclusters=50,design="clustered", maxdist <- 0.8, meansPCA= c(3, -1), sdPCA=c(2, 2), simulateResponse=c("bio2","bio5","bio10", "bio13","bio14", "bio19"), studyarea=c(-15, 65, 30, 75), seed=10

Further interesting settings:

- Case Study biased with outlier: npoints=50,design="biasedWithOutlier",countries = c("Germany","Ireland","France", "Sweden"), countriesOutlier= "Turkmenistan", meansPCA= c(3, -1), sdPCA=c(2, 2), simulateResponse=c("bio2","bio5","bio10", "bio13","bio14", "bio19"), studyarea=c(-15, 65, 30, 75), seed=10
- Case Study biased: same as biased with outlier but with design= "outlier"

```
npoints <- 50 # number of training samples

design <- "random"  # either clustered,biased,random, biasedWithOutlier

countries <- c("Germany","Ireland","France", "Sweden") #if design==biased
countriesOutlier <- "Turkmenistan" #if design==biasedWithOutlier A single point is set here
nclusters <- 50 #number of clusters if design==clustered
maxdist <- 0.8 #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 response
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 <- raster::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_vs <- generateSpFromPCA(predictors[[simulateResponse]],
                                 means = meansPCA,sds = sdPCA, plot=F)

response <- response_vs$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).

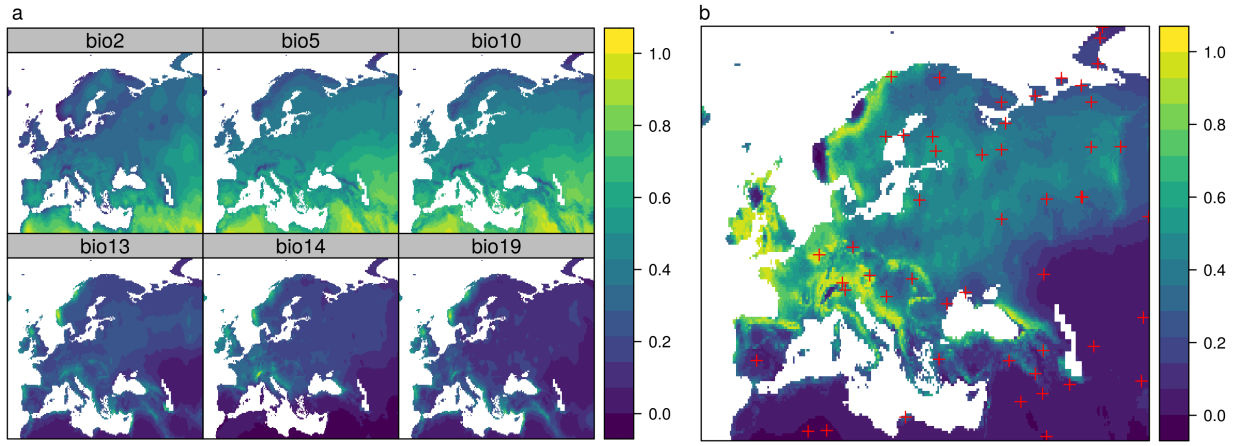

Figure 1: Subset of the predictors as well as the response variable and the selected training data points

Model training and prediction

Preparation

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

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

Visualize Response

```
#pca_pred <- predict(response_vs$details$pca,trainDat[,response_vs$details$variables])

#axes <- response_vs$details$axes
#axes.to.plot <- axes[1:2]
# xmin <- min(response_vs$details$pca$li[, axes.to.plot[1]]) - 0.3 * diff(range(response_vs$details$pca
# xmax <- max(response_vs$details$pca$li[, axes.to.plot[1]])
#
# pdf("../figures/plotresponse.pdf",width=6,height=6)
# plotResponse(response_vs,no.plot.reset=T)
# points(((pca_pred[,1]+10)/21.5),
#        pca_pred[,2],pch=16)
# dev.off()
```

Training and cross-validation

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)
if(design!="clustered"){
  model <- train(trainDat[,names(predictors)],
                 trainDat$response,
                 method="rf",
                 importance=TRUE,
                 tuneGrid = expand.grid(mtry = c(2:length(names(predictors)))),
                 trControl = trainControl(method="cv",savePredictions = TRUE))


  print("random cross-validation performance:")
  print(model)
}
```

```
## [1] "random cross-validation performance:"
## 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
##   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.
```

```
#if data are clustered, clustered CV is used:
if(design=="clustered"){
  folds <- CreateSpacetimeFolds(trainDat, spacevar="clstrID",k=nclusters)
  model <- train(trainDat[,names(predictors)],
                 trainDat$response,
                 method="rf",
                 importance=TRUE,
                 tuneGrid = expand.grid(mtry = c(2:length(names(predictors)))),
                 trControl = trainControl(method="cv",index=folds$index,savePredictions = TRUE))
  print("leave-cluster-out cross-validation performance:")
  print(model)

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

  print("random cross-validation performance:")
  print(model_random)
}
```

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 dissimilarity index.

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

Estimating the area of applicability

Variable importance to get weights

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

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

```
#with variable weighting:
AOA <- aoa(predictors,model=model,returnTrainDI = TRUE)
#without weighting:
AOA_noWeights <- aoa(predictors,train=trainDat,variables = names(predictors))

if (design=="clustered"){
  AOA_random<- aoa(predictors, model_random)
}

preds <- model$pred[model$pred$mtry==model$bestTune$mtry,]
absError <- abs(preds$pred[order(preds$rowIndex)]-preds$obs[order(preds$rowIndex)])
```

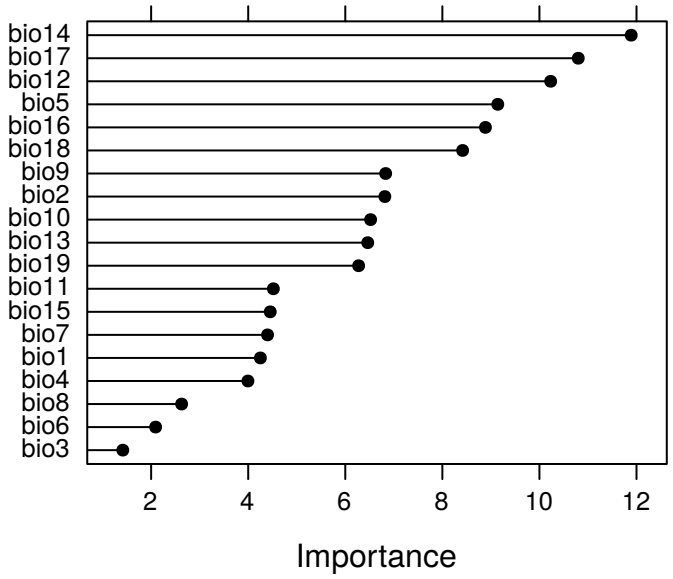

Figure 2: Variable importance

```
predtrain <- predict(model$finalModel,trainDat,predict.all=TRUE)
sdpred <-  apply(predtrain$individual,1,sd)
```

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 between prediction, error, DI and cross validation

```
## [1] "correlation coefficient (true error~DI)= 0.714964231889227"

## [1] "correlation coefficient (true error~DI estimated without weights)= 0.618232851966639"

## [1] "R² (prediction~reference)= 0.856409149140994"

## [1] "r (prediction~reference)= 0.925423767330942"

## [1] "RMSE (prediction,reference)= 0.0953649379038976"

## $Mean_train
## [1] 37.62449
##
## $threshold_stats
##       25%       50%       75%       90%       95%       99%      100%
```

```
## 0.1096012 0.1904797 0.3577250 0.5047949 0.5878770 0.6974194 0.7554121
##
## $threshold
## [1] 0.6370596
##
## $lower_threshold
## [1] 0.01530886

## [1] "R^2 AOA= 0.938743450608877"

## [1] "r AOA= 0.968887738909352"

## [1] "RMSE AOA= 0.0711663273926504"

## [1] "R^2 outside AOA= 1.7919287941121e-06"

## [1] "r outside AOA= -0.00133862944615461"

## [1] "RMSE outside AOA= 0.436650330543256"
```

Standard deviations of predictions within the AOA

The standard deviations of an ensemble are not helpful outside the AOA, however, inside the AOA they can provide helpful indicators of uncertainty, e.g. due to low data point densities.

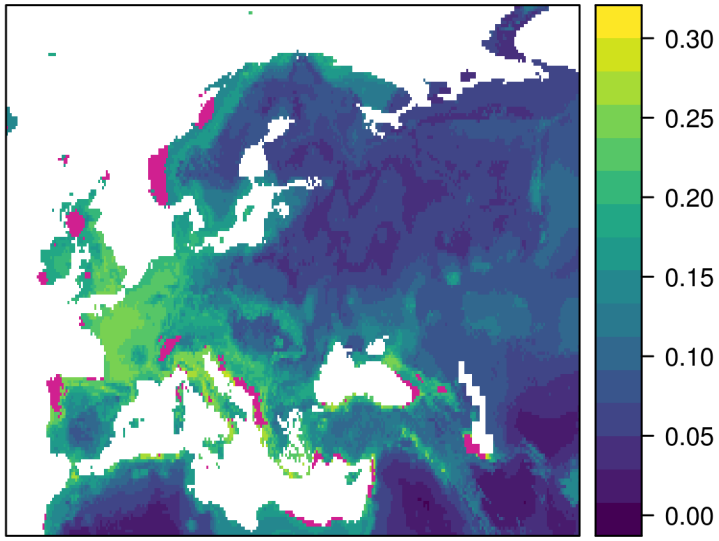

Figure 3: Standard deviations of the random forest ensemble within the AOA

Comparison

The dissimilarity index, as well as the standard deviations can then be compared to the true error.

```
compare <- stack(response,prediction,
                 predsd,truediff,
                 AOA$DI)
names(compare) <- c("response","prediction", "sd","true_diff","DI")
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
##        DI
##  Min.   :0.00
##  1st Qu.:0.12
##  Median :0.20
##  Mean   :0.25
##  3rd Qu.:0.34
##  Max.   :2.89
##  NA's   :47804
```

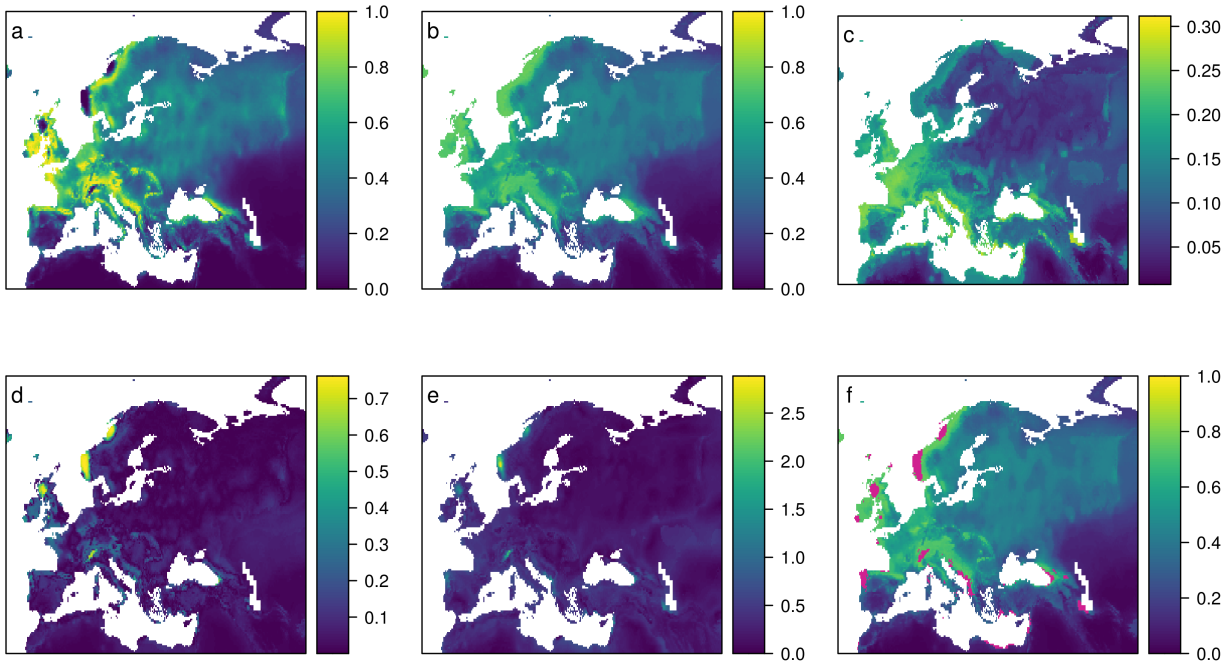

Figure 4: Comparison between reference (a), prediction (b), standard deviation of predictions (c), the true error (d), DI based on weighted variables (e), masked predictions (f)

If applicable: Comparison random vs spatial CV and AOA estimation

```
## [1] "not applicable here because no clustered design was chosen"
```

DI as a quantitative uncertainty measure?

Relationship between the DI and the absolute error on a data-point-level

Plotting the RMSE of cross-validated prodictions as well as the corresponding DI gives an impression about the error~DI relationship. This can also be visualized via hexbins for the true error~DI relationship.

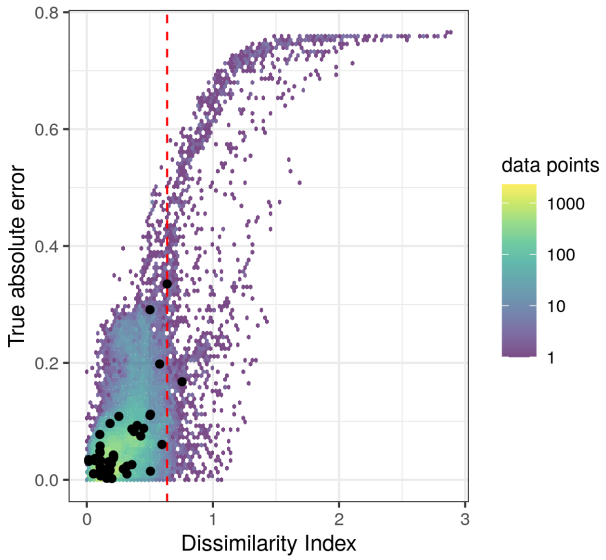

Figure 5: Relationship between the DI and the true error for the cross-validated training data (black points) as well as for the entire spatial reference dataset (hexbins)

Relationship between the DI and the requested performance measure (here RMSE)

The relationship between error and DI can be used to limit predictioons to an area (within the AOA) where a required performance (e.g. RMSE, R2, Kappa, Accuracy) applies. This can be done using the result of calibrate_aoa which used the relationship analyzed in a window of DI values. The corresponding model (here: shape constrained additive models which is the default: Monotone increasing P-splines with the dimension of the basis used to represent the smooth term is 6 and a 2nd order penalty.) can be used to estimate the performance on a pixel level, which then allows limiting predictions using a threshold.

Using muliple CV's

The calibration shown above uses the cross-validion folds used during model training only. However, to be able to test the ability of the model to do both, interpolation and extrapolation, we suggest to test multiple cross-validations. Here, we split the data into folds by clusterin in the predictor space. We use all cross-validated predictions and DI values to then estimate the error~Di relationship. Since cross-validation is done with respect to extrapolation as well, the estimated AOA is in most cases larger compared to the application of a CV which is not explicitely testing for extrapolation.

We compare the relationship estimated by cross-validation with the true relationship from the reference.

```
# model using single-CV
print(summary(attributes(AOA_new$AOA)$calib$model))
```

```
##
## Family: gaussian
```

```
## Link function: identity
##
## Formula:
## metric ~ s(DI, k = k, bs = bs, m = m)
## <environment: 0x563256af6408>
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.032998   0.004162   7.929 7.06e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##         edf Ref.df   F p-value
## s(DI) 2.023  2.045 133  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.8725   Deviance explained = 87.8%
## GCV score = 0.00028074  Scale est. = 0.00026188  n = 45
```

```
# model using multiple-CV
print(summary(attributes(AOA_new_MCV$AOA)$calib$model))
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## metric ~ s(DI, k = k, bs = bs, m = m)
## <environment: 0x5632381f2570>
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.017034   0.003632    4.69  3.8e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##         edf Ref.df    F p-value
## s(DI) 3.574  3.872 2674  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.9622   Deviance explained = 96.3%
## GCV score = 5.8216e-05  Scale est. = 5.7535e-05  n = 391
```

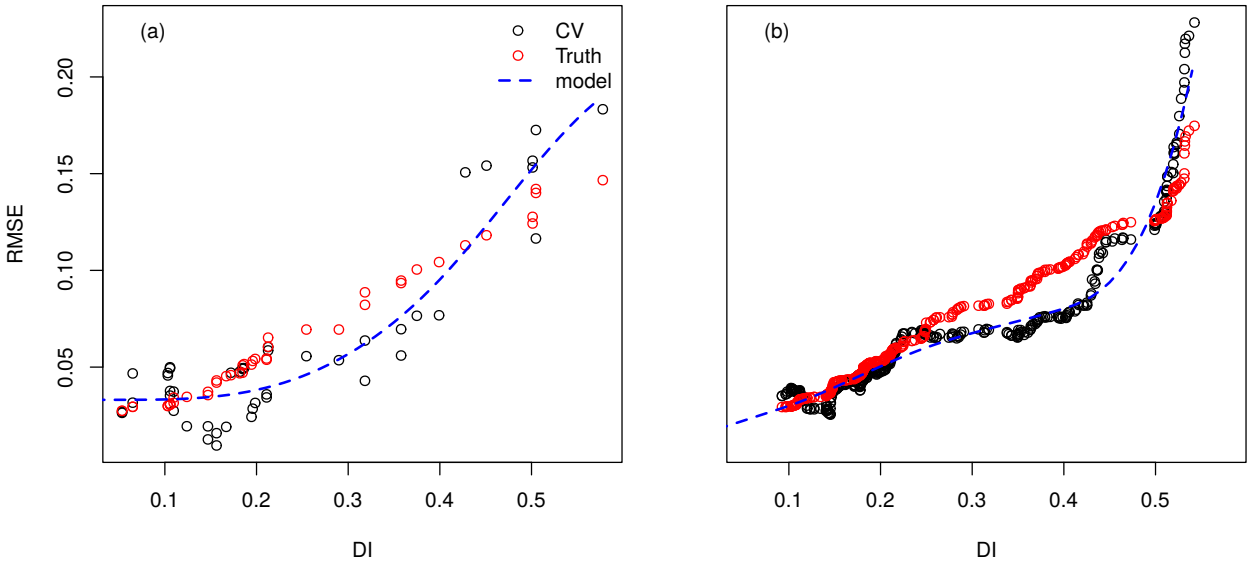

Figure 6: Relationship between the cross-validated DI and the RMSE using the CV strategy used during initial model training (a) as well as multiple cross-validation folds (clusters in predictor space from few to many clusters) (b). The RMSE was estimated using predictions and observed values within a window of DI values.

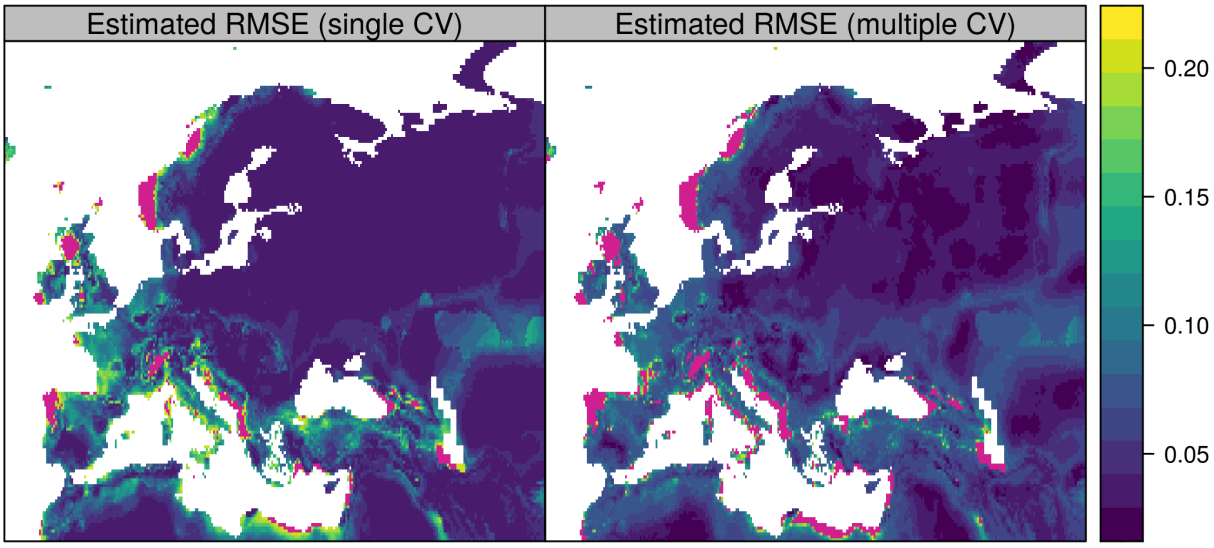

Figure 7: Expected performance using either single CV for calibration (b) or multiple CV (c) which can be used to limit predictions to areas where a required performance applies. For comparison, the true prediction error is shown (a).