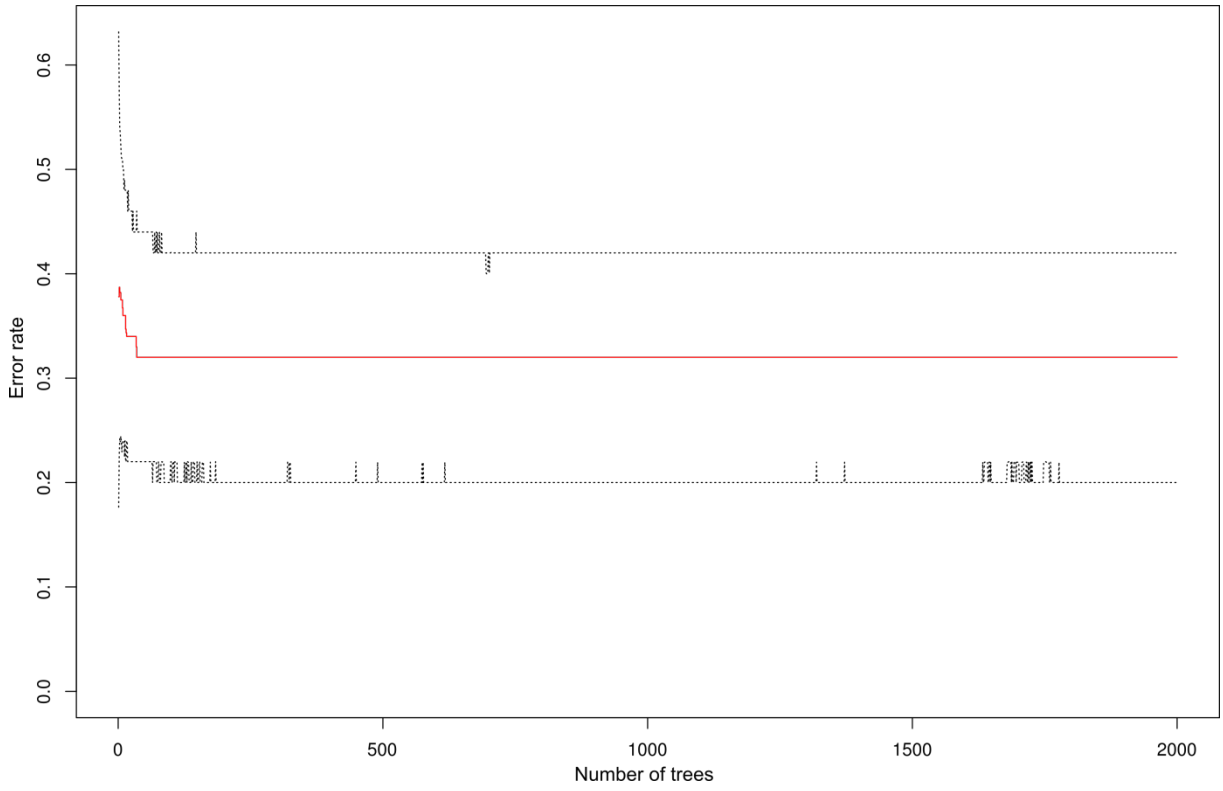
Supplementary Figure 1: Error rate of the random forest classification of randomly created test data sets (median, red, and the 2.5th and 97.5th percentiles obtained during 1,000 runs with randomly resampled data. After approximately 100 runs, the error rate stabilizes; more runs do not improve the classification performance but also do not worsen it. The figure has been created using the R software package (version 3.4.1 for Linux; http://CRAN.R-project.org/ (R Development Core Team, 2008)).



```r
#Unsupervised learning
#Schwarm
AktuelleDaten <-  as.matrix(HPTCapsaicinNGS_Shann_ML_ABC[grep("X",names
(HPTCapsaicinNGS_Shann_ML_ABC))])
#AktuelleDaten[AktuelleDaten > 0] <- 1
mode(AktuelleDaten) <-'numeric'
Distance=DistanceMatrix(AktuelleDaten, method="manhattan")
set.seed(42)
Projection <- Pswarm(Distance)
visualization=GeneratePswarmVisualization(AktuelleDaten,Projection
$ProjectedPoints,Projection$LC,FALSE)
plotTopographicMap(visualization$Umatrix,visualization$Bestmatches)
#snapshot3d("Swarm3d.png","png")
par(mfrow=c(2,2))
SchwarmCls=DBSclustering(k = 2, Data = AktuelleDaten, BestMatches =
                           Projection$ProjectedPoints, LC = Projection$LC, PlotIt
= T)
mosaicplot(table(HPTCapsaicinNGS_Shann_ML_ABC$CapsEff_Groups,SchwarmCls), col = c
("chartreuse3", "dodgerblue"),
           main = NULL, ylab="Genotype clusters [1 = Few variant alleles, \n2 =
many variant alleles]",
           xlab = "Phenotype clusters [1 = low capsaicin response, \n2 = high
capsaicin response]", cex = 1.2)
par(mfrow=c(1,1))

#Supervised learning
#Random Forests, AdaBoost, kNN, Regression, Naive Bayes, SVM
require(caTools)
require(sampling)
library("randomForest")
library(caret)
library(ada)
library(kernlab)
library(e1071)
library(pROC)
library(matrixStats)

PerformanceRF <- matrix(, nrow = 11, ncol = 0)
RFVarImp <- vector()
PerformanceAda <- matrix(, nrow = 11, ncol = 0)
AdaVarImp <- vector()
PerformanceKNNe <- matrix(, nrow = 11, ncol = 0)
PerformanceRegress <- matrix(, nrow = 11, ncol = 0)
PerformanceBayes <- matrix(, nrow = 11, ncol = 0)
PerformanceSVM <- matrix(, nrow = 11, ncol = 0)
AUROC_RF <- vector()
AUROC_Ada <- vector()
AUROC_KNN <- vector()
AUROC_Regress <- vector()
AUROC_Bayes <- vector()
AUROC_SVM <- vector()
RFconfmatrix <- matrix(c(0,0,0,0),ncol = 2,nrow = 2)

for (i in 1:1000)
{
  set.seed(42+i)
  sample <- sample.split(HPTCapsaicinNGS_Shann_ML_ABC$CapsEff_Groups, SplitRatio
= .67)
  TrainData <- subset(HPTCapsaicinNGS_Shann_ML_ABC, sample == TRUE)
  TestData <- subset(HPTCapsaicinNGS_Shann_ML_ABC, sample == FALSE)

  RandomForest_Classifier <- randomForest(as.factor(CapsEff_Groups) ~ .,
TrainData, ntree = 1000,
                                          mtry = 1*sqrt(ncol(TrainData)-1),
                                          na.action = na.roughfix,
                                          importance=TRUE, proximity=TRUE)

  PredRF <- predict(RandomForest_Classifier,TestData)
  PerformanceRF <- cbind(PerformanceRF,confusionMatrix(PredRF,TestData
```

```
$CapsEff_Groups,positive = "2")$byClass)
  cMatrix <- confusionMatrix(PredRF,TestData$CapsEff_Groups,positive = "2")
  RFconfmatrix <- RFconfmatrix+cMatrix$table
  RFVarImp <- append(RFVarImp,RandomForest_Classifier$importance[,3])
  AUROC_RF <- append(AUROC_RF,roc(TestData$CapsEff_Groups,as.numeric(PredRF))$auc)

  CartBoost_Classifier <- ada(CapsEff_Groups ~ ., data = TrainData,
                              control = rpart::rpart.control(maxdepth = 30,
                                                             cp = 0.001,
                                                             split="information",
                                                             minsplit = 20, xval =
10),
                              iter = 1000)

  PredAda <- predict(CartBoost_Classifier,TestData)
  PerformanceAda <- cbind(PerformanceAda,confusionMatrix(PredAda,TestData
$CapsEff_Groups,positive = "2")$byClass)
  AdaVarImp <- append(AdaVarImp,varplot(CartBoost_Classifier,type = "scores",
plot.it = F))
  AUROC_Ada <- append(AUROC_Ada,roc(TestData$CapsEff_Groups,as.numeric(PredAda))
$auc)

  kNNe_Classifier <- KNNclassifier(3, TrainData = TrainData[grep("X",names
(TrainData))],
                                   TestData = TestData[grep("X",names(TestData))],
                                   TrainCls = TrainData$CapsEff_Groups)
  PerformanceKNNe <- cbind(PerformanceKNNe,confusionMatrix(kNNe_Classifier
$KNNTestCls,TestData$CapsEff_Groups,positive = "2")$byClass)
  AUROC_KNN <- append(AUROC_KNN,roc(TestData$CapsEff_Groups,kNNe_Classifier
$KNNTestCls)$auc)

  Regress <- glm(as.factor(CapsEff_Groups-1) ~.,family=binomial,data=TrainData)
  PredRegress <- predict(Regress,TestData)
  PredRegress <- ifelse(PredRegress > 0.5,2,1)
  PerformanceRegress <- cbind(PerformanceRegress,confusionMatrix
(PredRegress,TestData$CapsEff_Groups,positive = "2")$byClass)
  AUROC_Regress <- append(AUROC_Regress,roc(TestData$CapsEff_Groups,PredRegress)
$auc)

  TrainData1 <- cbind(TrainData$CapsEff_Groups,apply(TrainData[2:ncol
(TrainData)],2,jitter))
  colnames(TrainData1)[1]  <- "CapsEff_Groups"
  Bayes_Classifier <- naiveBayes(as.factor(CapsEff_Groups) ~ ., data=as.data.frame
(TrainData), usekernel = F)
  predBayes <- (predict(Bayes_Classifier,TestData))
  PerformanceBayes <- cbind(PerformanceBayes,(caret::confusionMatrix(as.numeric
(predBayes),as.numeric(TestData$CapsEff_Groups))$byClass))
  AUROC_Bayes <- append(AUROC_Bayes,roc(TestData$CapsEff_Groups,as.numeric
(predBayes))$auc)

  SVM_Classifier <- ksvm(as.factor(CapsEff_Groups) ~ .,
                         data=TrainData,
                         kernel="rbfdot",
                         prob.model=TRUE,
                         type = "nu-svc")
  PredSVM <- predict(SVM_Classifier,TestData)
  PerformanceSVM <- cbind(PerformanceSVM,(caret::confusionMatrix(as.numeric
(PredSVM),as.numeric(TestData$CapsEff_Groups))$byClass))
  AUROC_SVM <- append(AUROC_SVM,roc(TestData$CapsEff_Groups,as.numeric(PredSVM))
$auc)
}

library(psych)
library(matrixStats)
print(describe(t(PerformanceRF), na.rm = T))
print(rowQuantiles(PerformanceRF, probs=c(0.025,0.5,0.975), na.rm = T)*100)
TestPerformanceRF <- rowQuantiles(PerformanceRF, probs=c(0.025,0.5,0.975), na.rm =
T)*100
rownames(TestPerformanceRF) <- rownames(PerformanceRF)
```

```r
quantile(AUROC_RF, probs=c(0.025,0.5,0.975))*100

print(describe(t(PerformanceAda), na.rm = T))
print(rowQuantiles(PerformanceAda, probs=c(0.025,0.5,0.975), na.rm = T)*100)
TestPerformanceAda <- rowQuantiles(PerformanceAda, probs=c(0.025,0.5,0.975), na.rm
= T)*100
rownames(TestPerformanceAda) <- rownames(PerformanceAda)
quantile(AUROC_Ada, probs=c(0.025,0.5,0.975))*100

print(describe(t(PerformanceKNNe),na.rm = T))
print(rowQuantiles(PerformanceKNNe, probs=c(0.025,0.5,0.975), na.rm = T)*100)
TestPerformanceKNNe <- rowQuantiles(PerformanceKNNe, probs=c(0.025,0.5,0.975),
na.rm = T)*100
rownames(TestPerformanceKNNe) <- rownames(PerformanceKNNe)
quantile(AUROC_KNN, probs=c(0.025,0.5,0.975))*100

print(describe(t(PerformanceRegress),na.rm = T))
print(rowQuantiles(PerformanceRegress, probs=c(0.025,0.5,0.975), na.rm = T)*100)
TestPerformanceRegress <- rowQuantiles(PerformanceRegress, probs=c
(0.025,0.5,0.975), na.rm = T)*100
rownames(TestPerformanceRegress) <- rownames(PerformanceRegress)
quantile(AUROC_Regress, probs=c(0.025,0.5,0.975))*100

print(describe(t(PerformanceBayes), na.rm = T))
print(rowQuantiles(PerformanceBayes, probs=c(0.025,0.5,0.975), na.rm = T)*100)
TestPerformanceBayes <- rowQuantiles(PerformanceBayes, probs=c(0.025,0.5,0.975),
na.rm = T)*100
rownames(TestPerformanceBayes) <- rownames(PerformanceBayes)
quantile(AUROC_Bayes, probs=c(0.025,0.5,0.975),na.rm = T)*100

print(describe(t(PerformanceSVM), na.rm = T))
print(rowQuantiles(PerformanceSVM, probs=c(0.025,0.5,0.975), na.rm = T)*100)
TestPerformanceSVM <- rowQuantiles(PerformanceSVM, probs=c(0.025,0.5,0.975), na.rm
= T)*100
rownames(TestPerformanceSVM) <- rownames(PerformanceSVM)
quantile(AUROC_SVM, probs=c(0.025,0.5,0.975))*100
```