Machine Learning - TP

Nathalie Villa-Vialaneix - nathalie.villa@univ-paris1.fr
http://www.nathalievilla.org

IUT STID (Carcassonne) & SAMM (Université Paris 1)

Formation INRA, Niveau 3

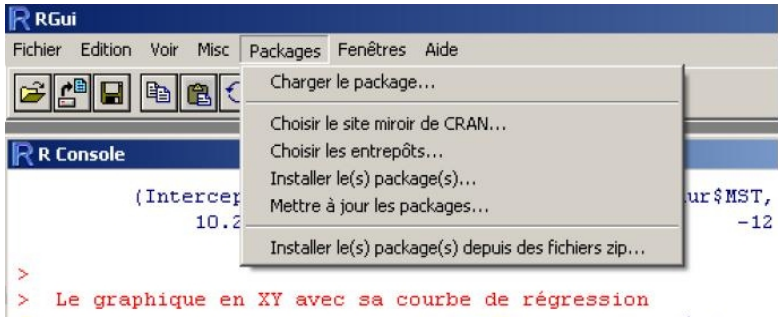
Packages in R

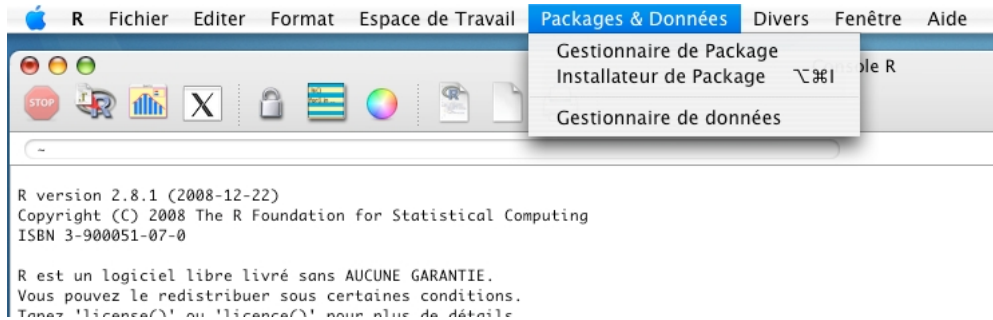
is provided with basic functions but more than 3,000 packages are available on the CRAN (Comprehensive R Archive Network) for additionnal functions (see also the project Bioconductor).

- **Installing new packages** (has to be done only once)
  - with the command line:

```
install.packages(c("nnet","e1071","rpart","car",
                   "randomForest"))
```

  - with the menu (Windows or Mac OS X)





- **Loading a package** (has to be done each time **R** is re-started)

```
library(nnet)
```

Working directory

All files in proper directories/subdirectories can be downloaded at: http://www.nathalievilla.org/docs/formation_inra, either as individual files or as a full zip file `inra-package.zip`.

The companion script `ML-scriptR.R` is made to be run from the subdirectory `ML/TP` of the provided material. For the computers in the classroom, this can be set by the following command line:

```
setwd("/home/fp/Bureau/inra-package/ML/TP")
```

If you are using Windows or Mac OS X, you can also choose to do it from the menu (in "Fichier" / "Définir le répertoire de travail").
In any case, **you must adapt the working directory** if you want to run the script on another computer!

Outline

Outline

Use case description

Data kindly provided by Laurence Liaubet described in [Liaubet et al., 2011]:

- microarray data: expression of 272 selected genes over 57 individuals (pigs);
- a phenotype of interest (muscle pH) measured over the 57 invididuals (numerical variable).

file 1: genes expressions
file 2: muscle pH

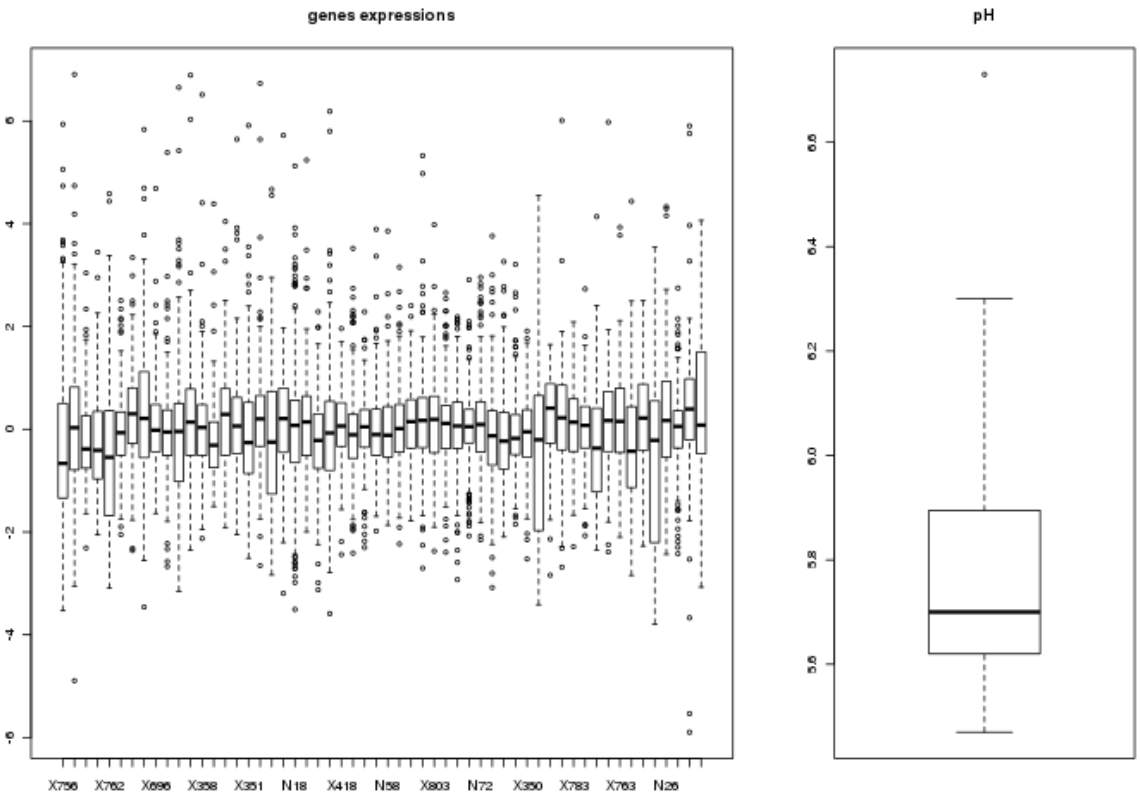
Load data

```
# Loading genes expressions
d <- read.table("../../Data/sel_data.csv",sep=";",
        header=T,row.names=1,dec=",")
dim(d)
names(d)
summary(d)
```

```
# Loading pH
pH <- scan("../../Data/sel_ph.csv",dec=",")
length(pH)
summary(pH)
```

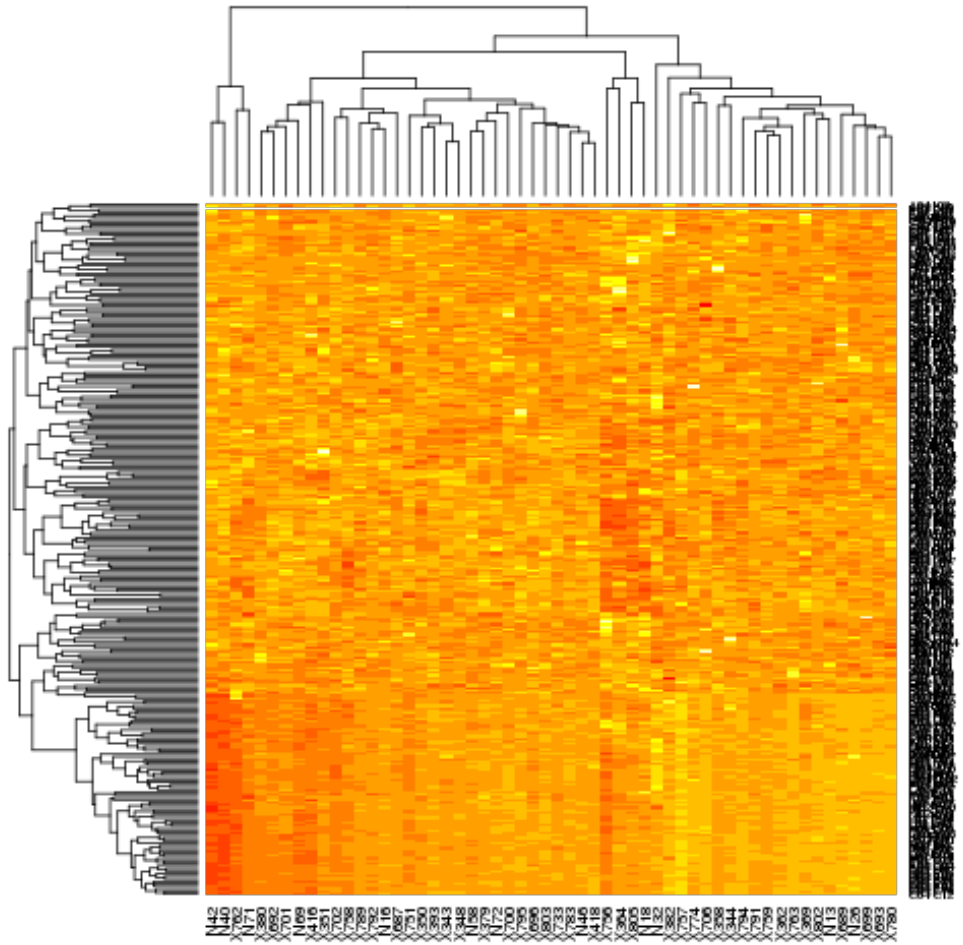
Basic analysis

```
# Data distribution
layout(matrix(c(1,1,2),ncol=3))
boxplot(d,main="genes expressions")
boxplot(pH,main="pH")
```



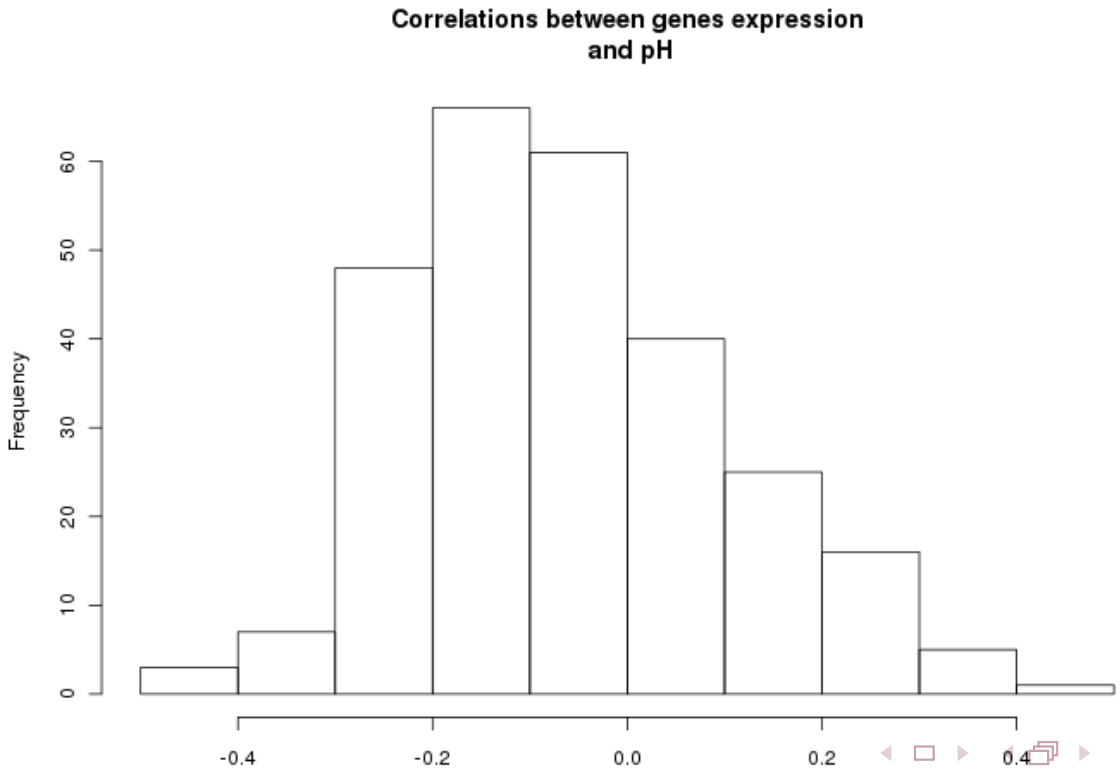
Correlation analysis

```
# Correlation analysis
heatmap(as.matrix(d))
```



Correlation with pH

```
# Correlation with pH
pH.cor <- function(x) {cor(x,pH)}
totalcor <- apply(d,1,pH.cor)
hist(totalcor,main="Correlations between genes
        expression\n and pH",xlab="Correlations")
```



Classification and regression tasks

1. **Regression**: predict pH (numerical variable) from genes expressions.
   Useful to: help verify the strength of the relation between genes expressions and pH; help understand the nature of the relation.
2. **Classification**: predict whether the pH is smaller or greater than 5.7 from genes expressions.
   (toy example)

```
# Classes definition
pH.classes <- rep(0,length(pH))
pH.classes[pH>5.7] <- 1
table(pH.classes)
```

Train/Test split: regression framework

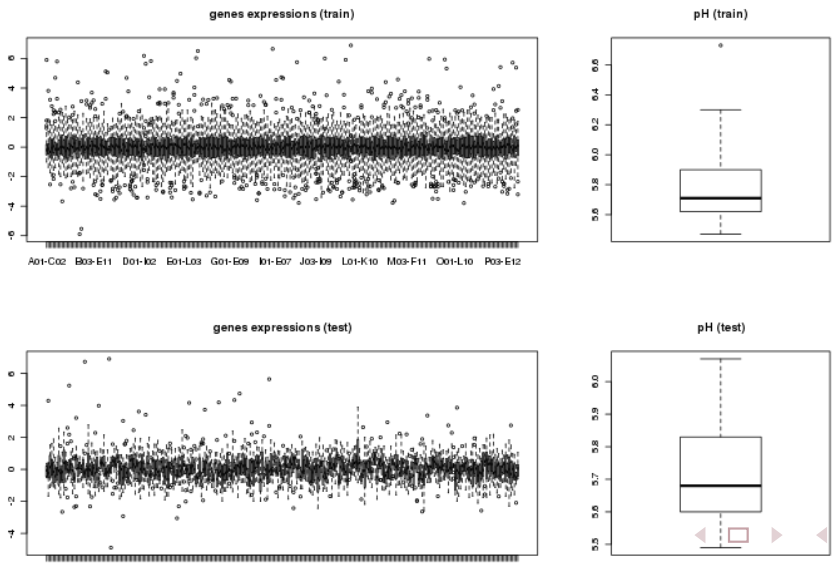
```
# Initialization and training sample selection
set.seed(16011357)
training <- sample(1:ncol(d),round(0.8*ncol(d)),
         replace=F)
# Matrices definition
d.train <- t(d[,training])
pH.train <- pH[training]
d.test <- t(d[,-training])
pH.test <- pH[-training]
# Data frames definition
r.train <- cbind(d.train,pH.train)
r.test <- cbind(d.test,pH.test)
colnames(r.train) <- c(colnames(d.train),"pH")
colnames(r.test) <- c(colnames(d.train),"pH")
r.train <- data.frame(r.train)
r.test <- data.frame(r.test)
```

Train/Test split: classification framework

```
# Vectors definition
pHc.train <- pH.classes[training]
pHc.test <- pH.classes[-training]
# Data frames definition
c.train <- cbind(d.train,pHc.train)
c.test <- cbind(d.test,pHc.test)
colnames(c.train) <- c(colnames(d.train),"pHc")
colnames(c.test) <- c(colnames(d.train),"pHc")
c.train <- data.frame(c.train)
c.test <- data.frame(c.test)
# Transforming pHc into a factor
c.train$pHc <- factor(c.train$pHc)
c.test$pHc <- factor(c.test$pHc)
```

Training / Test sets (short) analysis

```
layout(matrix(c(1,1,2,3,3,4),ncol=3,nrow=2,
       byrow=T))
boxplot(d.train,main="genes expressions (train)")
boxplot(pH.train,main="pH (train)")
boxplot(d.test,main="genes expressions (test)")
boxplot(pH.test,main="pH (test)")
table(pHc.train)
table(pHc.test)
```



Outline

Loading the data (to be consistent) and nnet library

```
# Loading the data
load("../../Data/train-test.Rdata")
```

MLP are **unusable with a large number of predictors** (here: 272 predictors for 45 observations only in the training set) $\Rightarrow$ selection of a relevant subset of variables (with LASSO):

```
# Loading the subset of predictors
load("../../Data/selected.Rdata")
```

MLPs are provided in the **nnet** package:

```
# Loading nnet library
library(nnet)
```

Simple use: MLP in the regression framework

Training

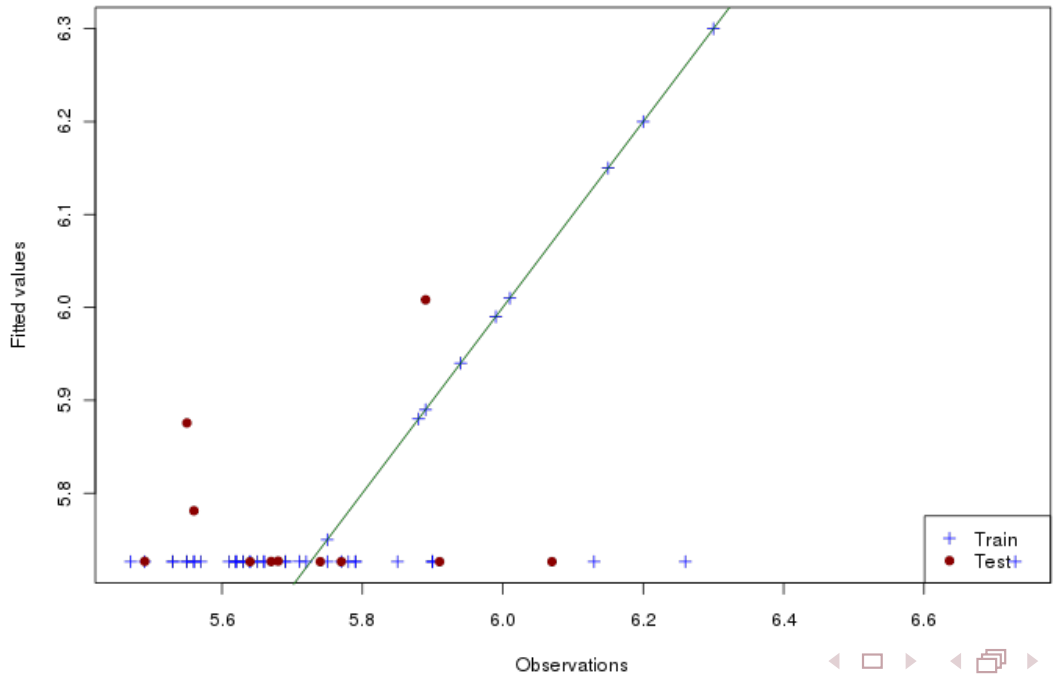
```
# Simple use: MLP with p=3 and no decay
set.seed(17011644)
nn1 <- nnet(d.train[,selected],pH.train,size=3,
        decay=0,maxit=500,linout=T)
```

Analysis

```
print(nn1)
summary(nn1)
# Training error and pseudo-R2
mean((pH.train-nn1$fitted)^2)
1-mean((pH.train-nn1$fitted)^2)/var(pH.train)
# Predictions (test set)
pred.test <- predict(nn1,d.test[,selected])
# Test error and pseudo-R2
mean((pH.test-pred.test)^2)
1-mean((pH.test-pred.test)^2)/var(pH.train)
```

Predictions vs observations

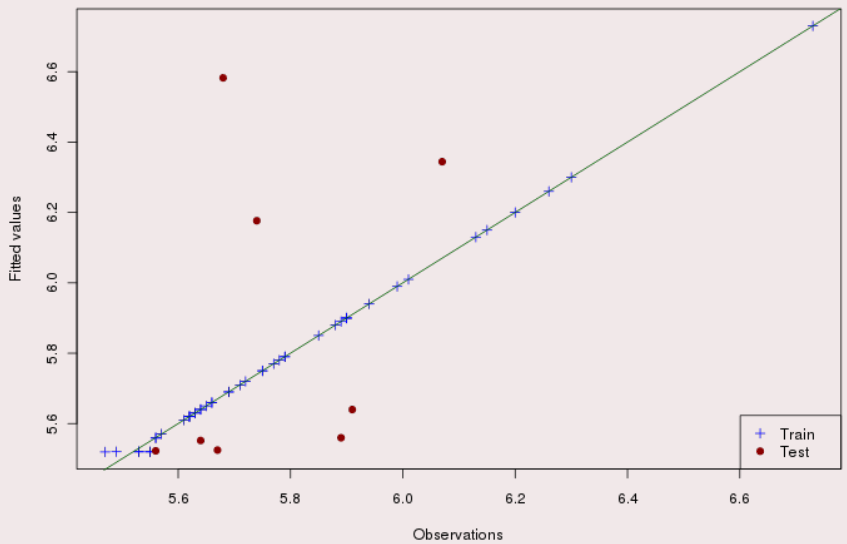
```
plot(pH.train,nn1$fitted,xlab="Observations",
     ylab="Fitted values",main="",pch=3,
     col="blue")
points(pH.test,pred.test,pch=19,col="darkred")
legend("bottomright",pch=c(3,19),col=c("blue",
     "darkred"),legend=c("Train","Test"))
abline(0,1,col="darkgreen")
```



Check results variability

```
set.seed(17012120)
nn2 <- nnet(d.train[,selected],pH.train,size=3,
            decay=0,maxit=500,linout=T)
...
```

Summary



	Train	Test
nn1	30.8%	44.9%
nn2	99.8%	-34!!

Simple use: MLP in the classification framework

Training

```
# Simple use: MLP with p=3 and no decay
set.seed(17011716)
nnc1 <- nnet(pHc~.,data=c.train,size=3,decay=0,
             maxit=500,linout=F)
```

Analysis

```
print(nnc1)
summary(nnc1)
# Predictions
nnc1$fitted
# Recoding
pred.train <- rep(0,length(nnc1$fitted))
pred.train[nnc1$fitted>0.5] <- 1
# Training error
table(pred.train,c.train$pHc)
sum(pred.train!=c.train$pHc)/length(pred.train)
```

Test error

```
# Predictions and recoding
raw.pred.test <- predict(nnc1,c.test)
pred.test <- rep(0,length(raw.pred.test))
pred.test[raw.pred.test>0.5] <- 1
# Test error
table(pred.test,c.test$pHc)
sum(pred.test!=c.test$pHc)/length(pred.test)
```

Summary

Train

Observations / Predictions	0	1
0	22	1
1	0	22

Test

Observations / Predictions	0	1
0	4	3
1	2	2

Overall misclassification rate:

2.22% 45.45%

Tuning MLP with the **e1071** package

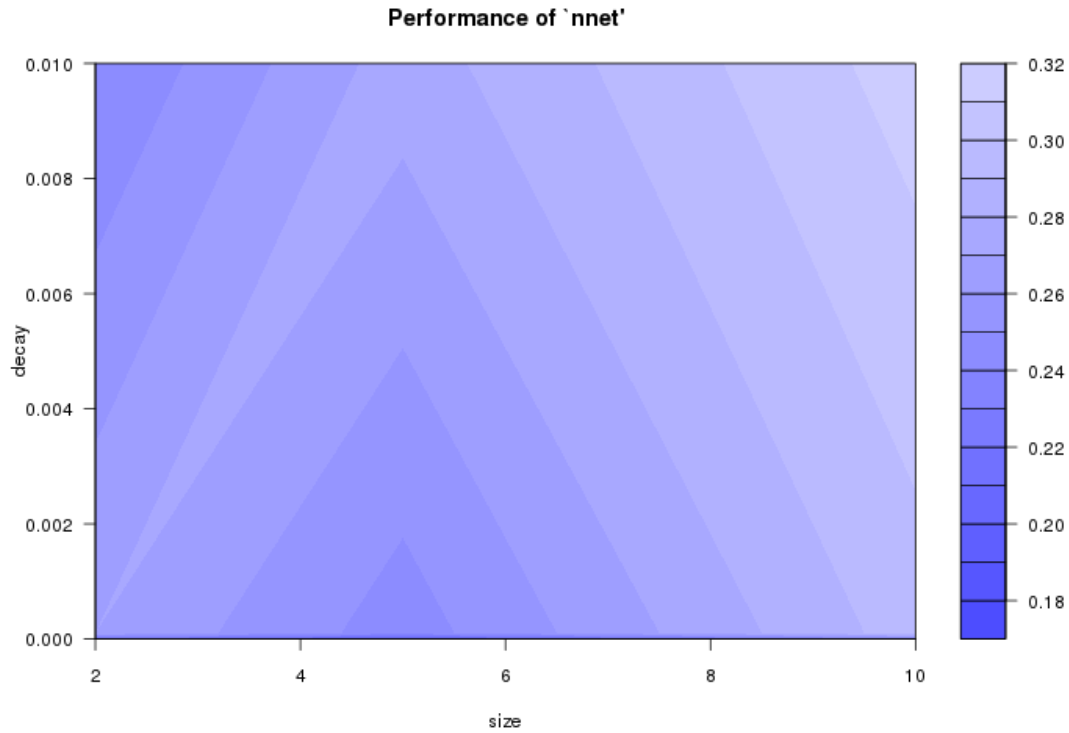
Search for the best parameters with 10-fold CV

```
library(e1071)
set.seed(1643)
t.nnc2 <- tune.nnet(pHc~.,data=c.train[,
        c(selected,ncol(c.train))],size=c(2,5,10),
        decay=10^(-c(10,8,6,4,2)),maxit=500,
        linout=F,tunecontrol=
        tune.control(nrepeat=5,sampling="cross",
        cross=10))
```

Basic analysis of the output

```
# Looking for the best parameters
plot(t.nnc2)
```

Best parameters?




```
summary(t.nnc2)
t.nnc2$best.parameters
# Selecting the best MLP
nnc2 <- t.nnc2$best.model
```

Results analysis

```
# Training error
pred.train <- rep(0,length(nnc2$fitted))
pred.train[nnc2$fitted>0.5] <- 1
table(pred.train,c.train$pHc)
sum(pred.train!=c.train$pHc)/length(pred.train)
# Predictions and test error
raw.pred.test <- predict(nnc2,c.test)
pred.test <- rep(0,length(raw.pred.test))
pred.test[raw.pred.test>0.5] <- 1
table(pred.test,c.test$pHc)
sum(pred.test!=c.test$pHc)/length(pred.test)
```

Summary

	Train	Test
nnc1	2.22%	45.45%
nnc2	0%	22.27%

Outline

Regression tree training

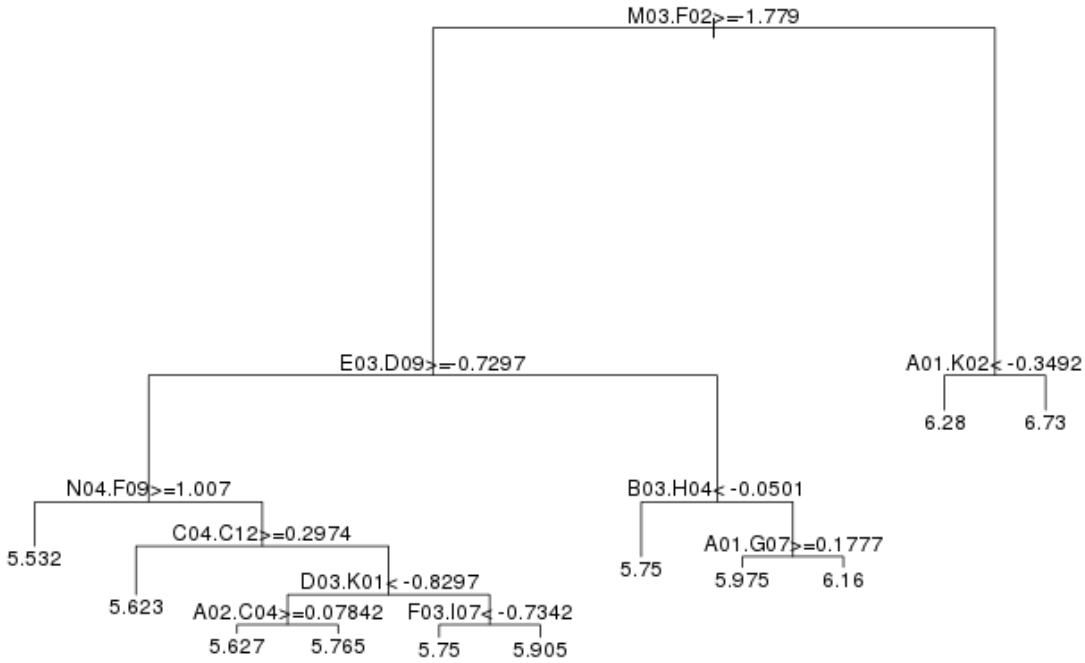
Training with the rpart package

```
library(rpart)
# Regression tree
tree1 <- rpart(pH~.,data=r.train,control=
          rpart.control(minsplit=2))
```

Basic analysis of the output

```
print(tree1)
summary(tree1)
plot(tree1)
text(tree1)
```

Resulting tree

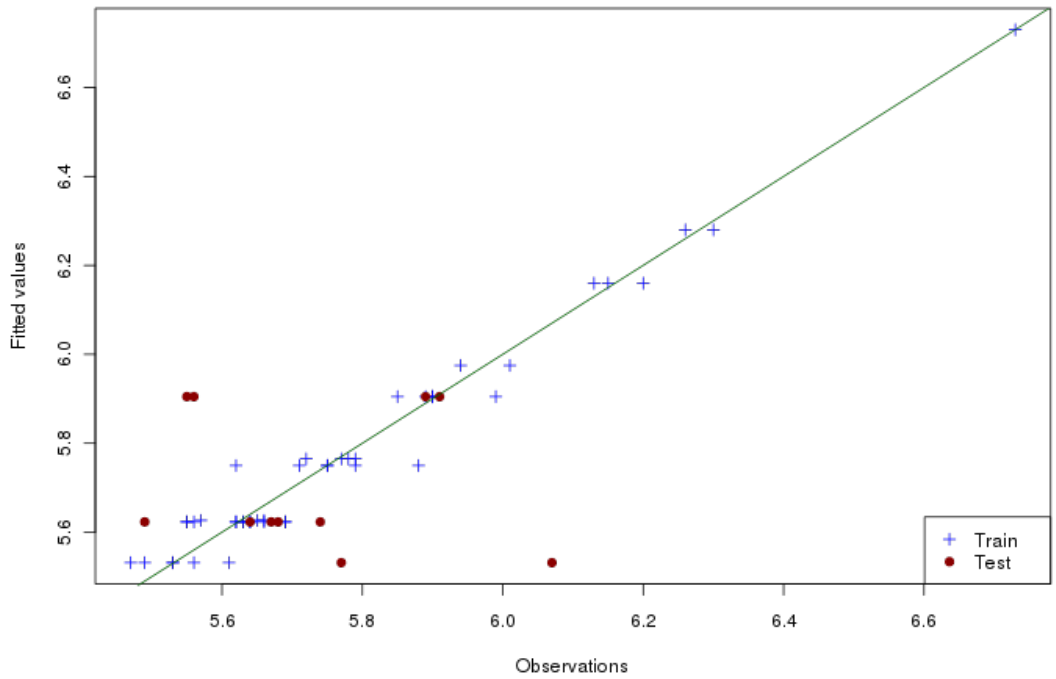



```
tree1$where # leaf number
tree1$frame # nodes features
```

Performance analysis

```
# Training predictions and error
pred.train <- tree1$frame$yval[tree1$where]
mean((pred.train-r.train$pH)^2)
1-mean((pred.train-r.train$pH)^2)/var(pH.train)
# Test predictions and error
pred.test <- predict(tree1,r.test)
mean((pred.test-r.test$pH)^2)
1-mean((pred.test-r.test$pH)^2)/var(pH.train)
# Fitted values vs True values
plot(pH.train,pred.train,xlab="Observations",
        ylab="Fitted values",main="",pch=3,
        col="blue")
points(pH.test,pred.test,pch=19,col="darkred")
legend("bottomright",pch=c(3,19),col=c("blue",
        "darkred"),legend=c("Train","Test"))
abline(0,1,col="darkgreen")
```

Summary



Numerical performance

	Train	Test
tree1	96.6%	11.6%
nn2	99.8%	-34!!

Classification tree training and tuning

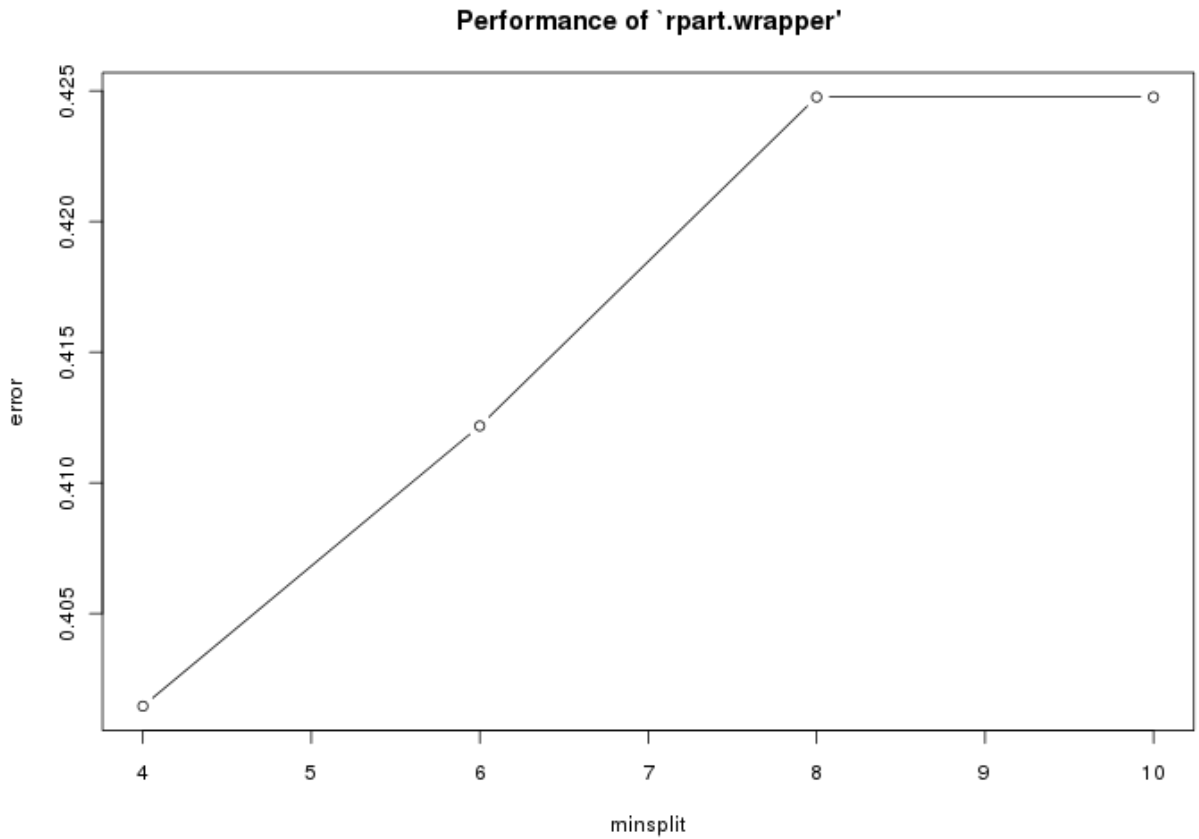
Training with the rpart package and tuning with the e1071 package

```
# Random seed initialization
set.seed(20011108)
# Tuning the minimal number of observations in a
# node (minsplit; default value is 20)
t.treec1 <- tune.rpart(pHc~.,data=c.train,
        minsplit=c(4,6,8,10),tunecontrol=
        tune.control(sampling="bootstrap",
        nboot=20))
```

Basic analysis of the output

```
plot(t.treec1)
```

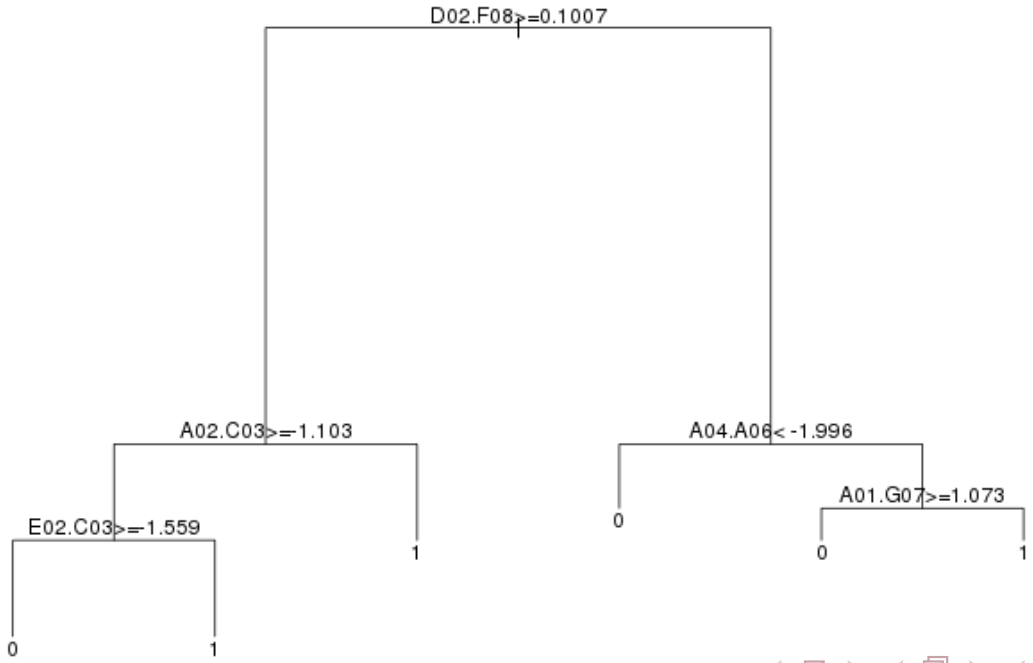
Tuning results




```
t.treec1$best.parameters
treec1 <- t.treec1$best.model
```

Basic analysis of the best tree

```
summary(treec1)
plot(treec1)
text(treec1)
treec1$where # leaf number
treec1$frame # nodes features
```



Predictions and errors

Training set

```
# Make the prediction
pred.train <- predict(treec1,c.train)
# Find out which class is predicted
pred.train <- apply(pred.train,1,which.max)
library(car) # to have the "recode" function
pred.train <- recode(pred.train,"2=1;1=0")
# Calculate misclassification error
table(pred.train,pHc.train)
sum(pred.train!=pHc.train)/length(pred.train)
```

Test set

```
pred.test <- predict(treec1,c.test)
pred.test <- apply(pred.test,1,which.max)
pred.test <- recode(pred.test,"2=1;1=0")
table(pred.test,pHc.test)
sum(pred.test!=pHc.test)/length(pred.test)
```

Classification performance summary

Overall misclassification rates

	Train	Test
nnc1	2.22%	45.45%
nnc2	0	22.27%
treec1	0	45.45%

Outline

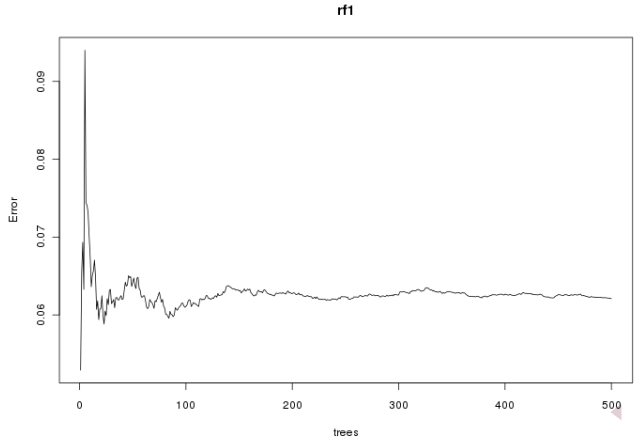
Random forest training (regression framework)

Training with the randomForest package

```
library(randomForest)
set.seed(21011144)
rf1 <- randomForest(d.train,pH.train,importance=T,
        keep.forest=T)
```

Basic analysis of the output

```
plot(rf1)
rf1$ntree
rf1$mtry
rf1$importance
```



Predictions and errors

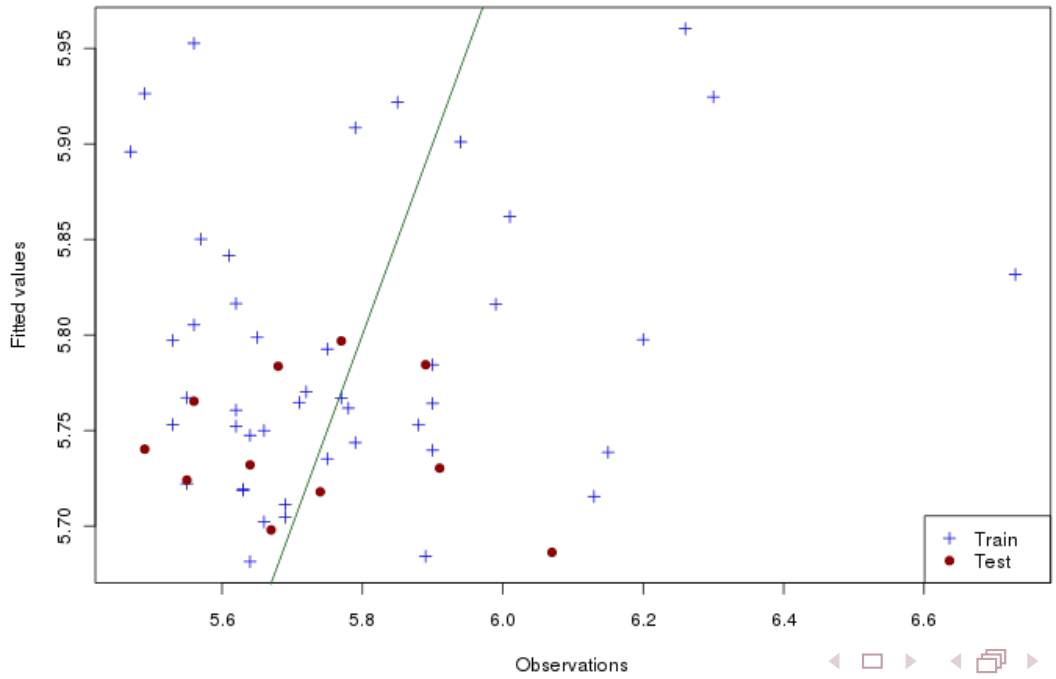
```
# Training set (oob error and training error)
mean((pH.train-rf1$predicted)^2)
rf1$mse
1-mean((pH.train-rf1$predicted)^2)/var(pH.train)
1-mean((pH.train-predict(rf1,d.train))^2)
        /var(pH.train)
# Test set
pred.test <- predict(rf1,d.test)
mean((pH.test-pred.test)^2)
1-mean((pH.test-pred.test)^2)/var(pH.train)
```

MSE

	Train	Test
nn1	30.8%	44.9%
nn2	99.8%	-34
rf1	84.5%	51.2%%

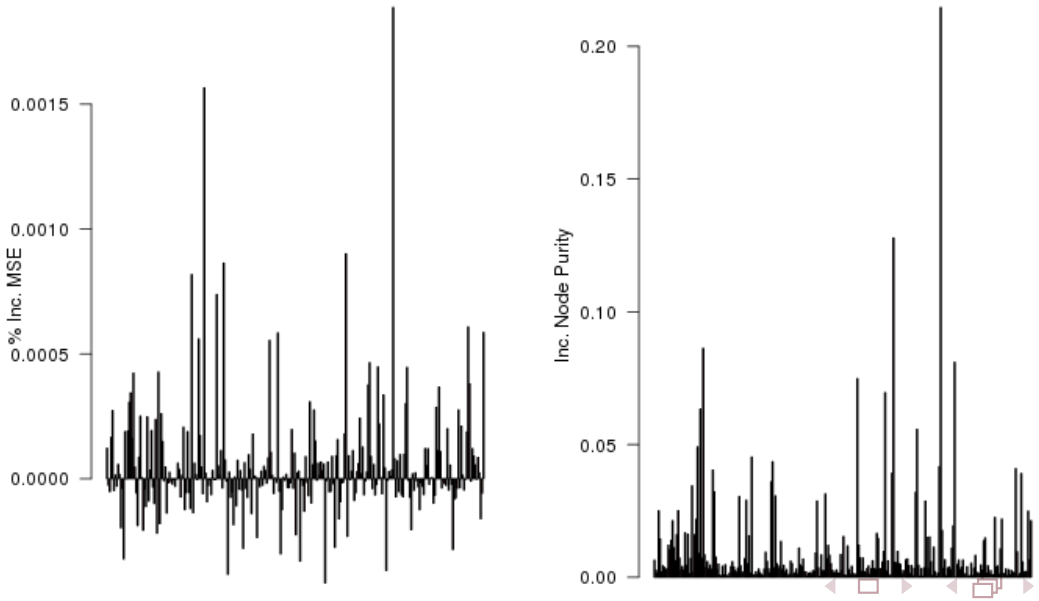
Predictions vs observations

```
plot(pH.train,rf1$predicted,xlab="Observations",
        ylab="Fitted values",main="",pch=3,
        col="blue")
points(pH.test,pred.test,pch=19,col="darkred")
legend("bottomright",pch=c(3,19),col=c("blue",
        "darkred"),legend=c("Train","Test"))
abline(0,1,col="darkgreen")
```



Importance analysis

```
layout(matrix(c(1,2),ncol=2))
barplot(t(rf1$importance[,1]),xlab="variables",
        ylab="% Inc. MSE",col="darkred",las=2,
        names=rep(NA,nrow(rf1$importance)))
barplot(t(rf1$importance[,2]),xlab="variables",
        ylab="Inc. Node Purity",col="darkred",
        las=2,names=rep(NA,nrow(rf1$importance)))
which(rf1$importance[,1]>0.0005)
```



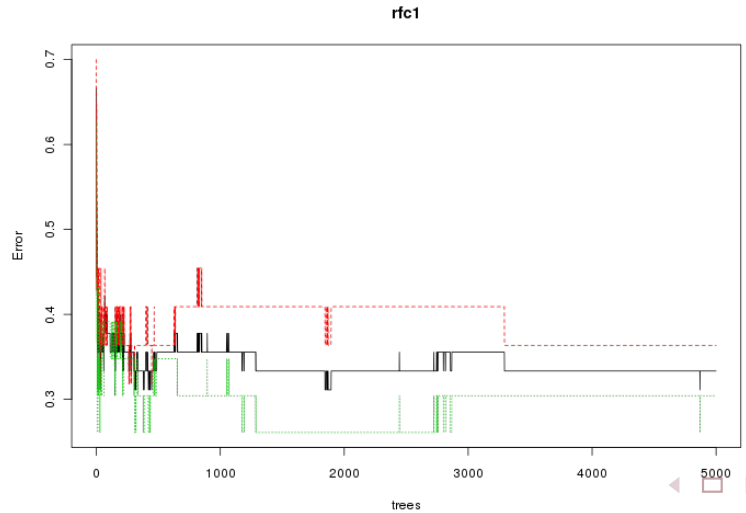
Random forest training (classification framework)

Training with the randomForest package (advanced features)

```
set.seed(20011203)
rfc1 <- randomForest(d.train,factor(pHc.train),
        ntree=5000,mtry=150,sampsize=40,nodesize=
        2,xtest=d.test,ytest=factor(pHc.test),
        importance=T,keep.forest=F)
```

Basic analysis of the output

```
plot(rfc1)
rfc1$importance
```



Predictions and errors

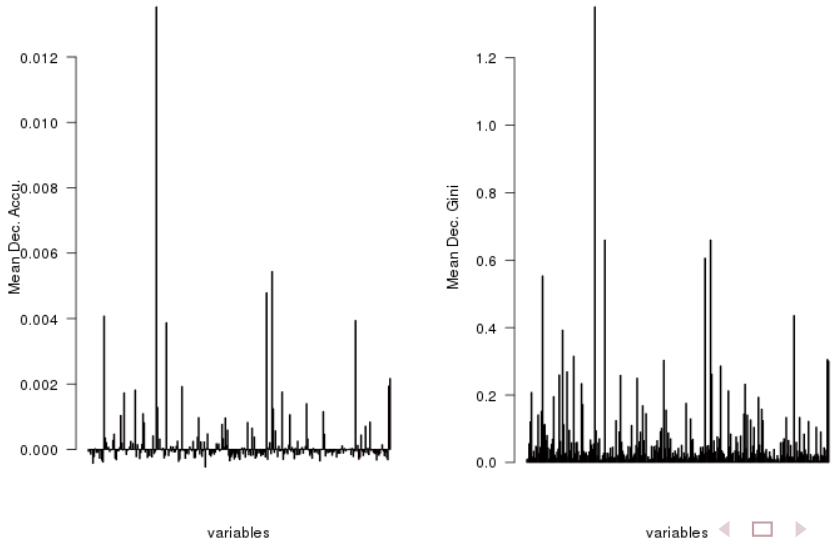
```
# Training set (oob error and training error)
table(rfc1$predicted,pHc.train)
sum(rfc1$predicted!=pHc.train)/length(pHc.train)
# Test set
table(rfc1$test$predicted,pHc.test)
sum(rfc1$test$predicted!=pHc.test)/length(pHc.test)
```

Overall misclassification rates

	Train	Test
nnc1	2.22%	45.45%
nnc2	0	22.27%
treec1	0	45.45%
rfc1	*33.33%*	36.36%

Importance analysis

```
# Importance analysis
layout(matrix(c(1,2),ncol=2))
barplot(t(rfc1$importance[,3]),xlab="variables",
        ylab="Mean Dec. Accu.",col="darkred",
        las=2,names=rep(NA,nrow(rfc1$importance)))
barplot(t(rfc1$importance[,4]),xlab="variables",
        ylab="Mean Dec. Gini",col="darkred",
        las=2,names=rep(NA,nrow(rfc1$importance)))
which(rfc1$importance[,3]>0.002)
```



Liaubet, L., Lobjois, V., Faraut, T., Tircazes, A., Benne, F., Iannuccelli, N., Pires, J., Glénisson, J., Robic, A., Le Roy, P., SanCristobal, M., and Cherel, P. (2011).

Genetic variability or transcript abundance in pig peri-mortem skeletal muscle: eQTL localized genes involved in stress response, cell death, muscle disorders and metabolism.

BMC Genomics, 12(548).