Outline

Random Forests

The following misclassification errors compare “Random Forests” with single trees. RF are closely related to bagged trees.

TABLE 2
Test set misclassification error (%)

Data set	**Forest**	**Single tree**
Breast cancer	2.9	5.9
Ionosphere	5.5	11.2
Diabetes	24.2	25.3
Glass	22.0	30.4
Soybean	5.7	8.6
Letters	3.4	12.4
Satellite	8.6	14.8
Shuttle $\times 10^3$	7.0	62.0
DNA	3.9	6.2
Digit	6.2	17.1

from Breiman: “Statistical Modelling: the two cultures”.

Random Forests (Breiman, 2001) are widely believed to be the best “off-the-shelf” classifiers for high-dimensional data.

Similar to bagged decision trees with a few key differences:

- For each splitpoint, the search is not over all p variables but just over *mtry* variables (where e.g. *mtry* $= \lfloor p/3 \rfloor$)
- No pruning necessary. Trees can be grown until each node contains just very few observations (1 or 5).

Bagged decision trees can be seen as a special case of Random Forests (for *mtry*=p), if trees are not pruned, e.g. always grown to maximal depth.

Advantages of RF over bagged decision trees

- better prediction (in general).
- almost no parameter tuning necessary with RF (although it still helps to vary the value of *mtry*). Tree depth needs to be chosen carefully with bagging, while we can always grow trees without pruning with RF.

Random Forests are implemented in package `randomForest`. Looking at the Boston Housing data again (and at the help page for `randomForest` first).

```
library(randomForest)
library(MASS)
data(Boston)

y <- Boston[,14]
x <- Boston[,1:13]

?randomForest
```

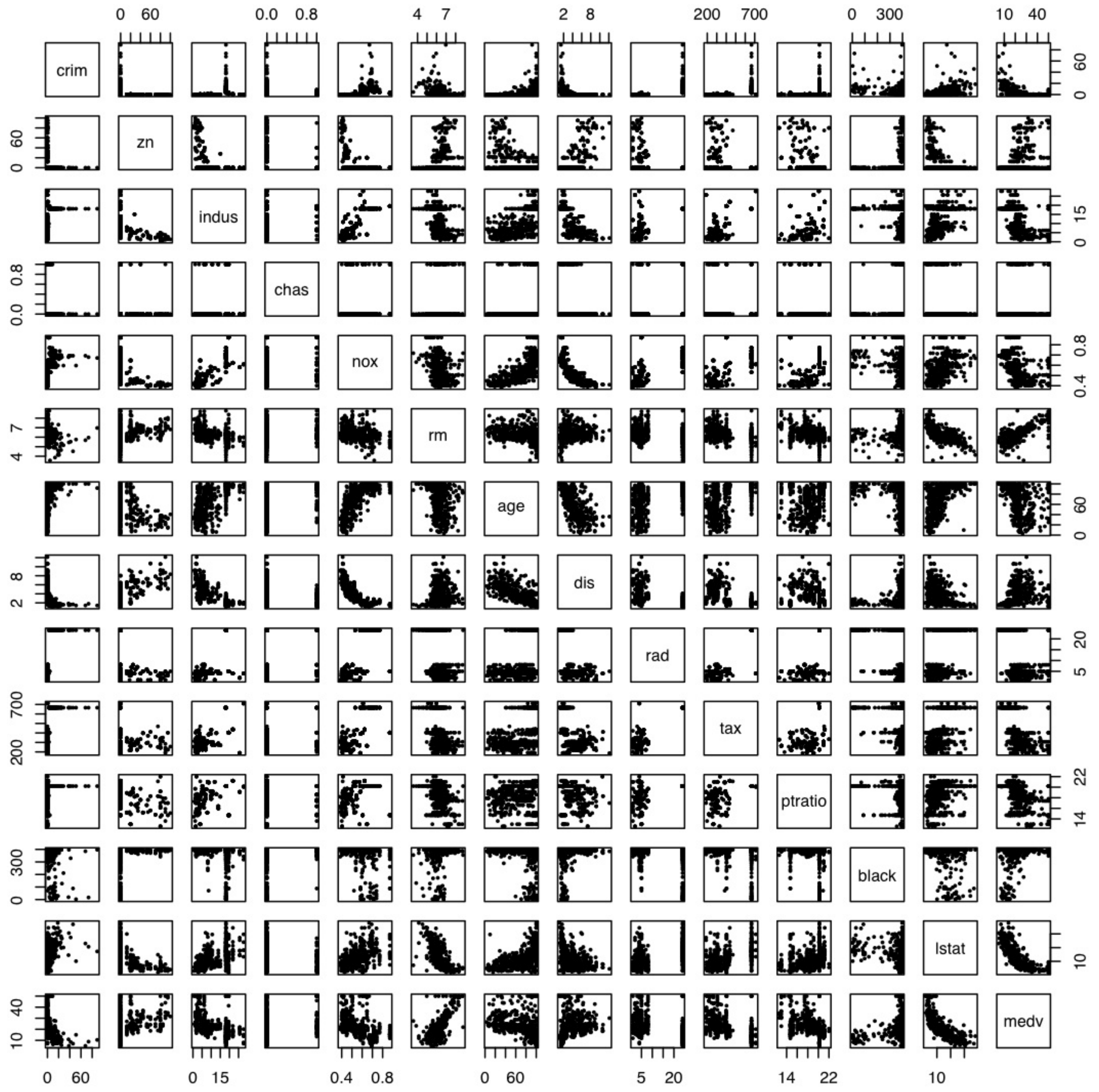
```
> randomForest          package:randomForest          R Documentation

Classification and Regression with Random Forest

Description:
     'randomForest' implements Breiman's random forest algorithm (based
     on Breiman and Cutler's original Fortran code) for classification
     and regression.  It can also be used in unsupervised mode for
     assessing proximities among data points.

Usage:
     ## S3 method for class 'formula':
     randomForest(formula, data=NULL, ..., subset, na.action=na.fail)
     ## Default S3 method:
     randomForest(x, y=NULL,  xtest=NULL, ytest=NULL, ntree=500,
                  mtry=if (!is.null(y) && !is.factor(y))
                  max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
                  replace=TRUE, classwt=NULL, cutoff, strata,
                  sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x
                  nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
                  importance=FALSE, localImp=FALSE, nPerm=1,
                  proximity=FALSE, oob.prox=proximity,
                  norm.votes=TRUE, do.trace=FALSE,
                  keep.forest=!is.null(y) && is.null(xtest), corr.bias=FAI
                  keep.inbag=FALSE, ...)
```

Boston Housing data, again.



```
> rf <- randomForest(x,y)
> print(rf)
>
Call:
 randomForest(x = x, y = y)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 10.26161
                    % Var explained: 87.84
```

Can plot the predicted values (out-of-bag estimation) vs. true values by

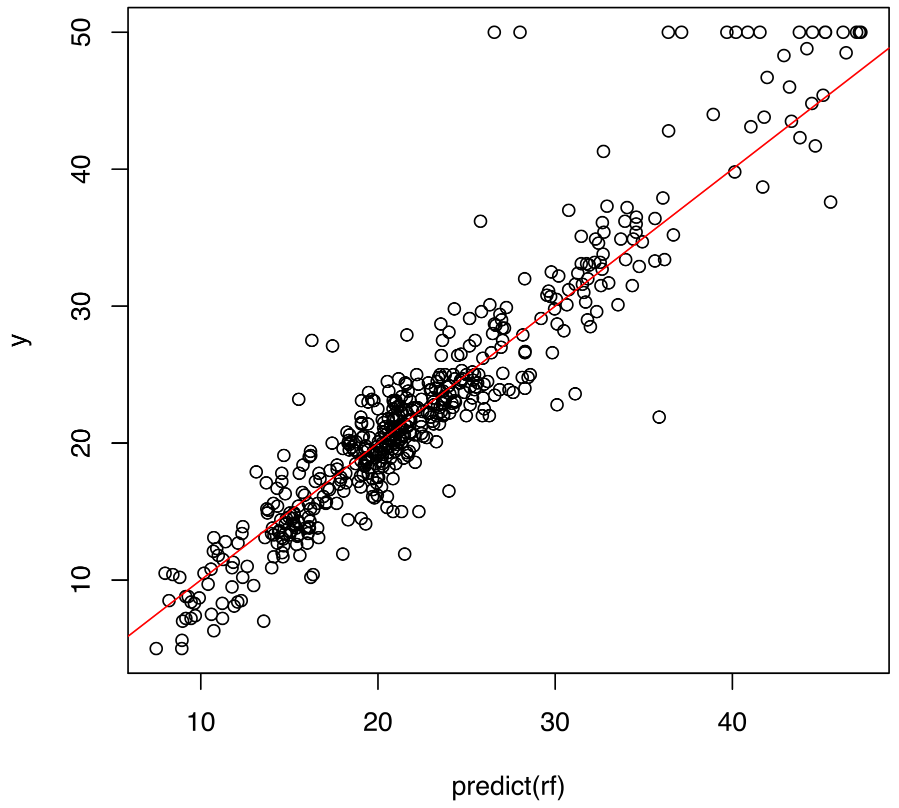
```
> plot( predict(rf), y)
> abline(c(0,1),col=2)
```

Same if treating the training data as new data

```
> plot( predict(rf,newdata=x), y)
```

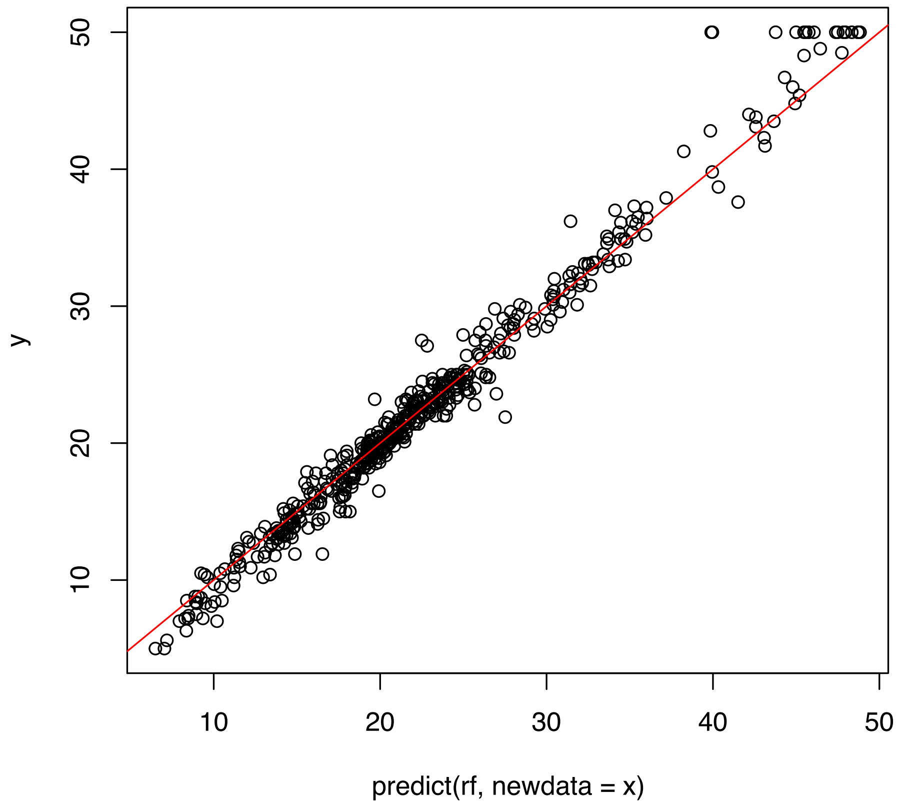
Out-of-bag error.

```
> plot( predict(rf), y)
> abline(c(0,1),col=2)
```



Training error.

```
> plot( predict(rf,newdata=x), y)
> abline(c(0,1),col=2)
```



Try `mtry` 2

```
> (rf <- randomForest(x,y,mtry=2))
Call:
 randomForest(x = x, y = y, mtry = 2)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 2

          Mean of squared residuals: 12.17176
                    % Var explained: 85.58
```

Try `mtry` 4

```
> (rf <- randomForest(x,y,mtry=4))
Call:
 randomForest(x = x, y = y, mtry = 4)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 10.01574
                    % Var explained: 88.14
```

And `mtry` 8 and 10.

```
> (rf <- randomForest(x,y,mtry=8))
Call:
 randomForest(x = x, y = y, mtry = 8)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 8

          Mean of squared residuals: 9.552806
                    % Var explained: 88.68

> > (rf <- randomForest(x,y,mtry=10))
Call:
 randomForest(x = x, y = y, mtry = 10)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 10

          Mean of squared residuals: 9.774435
                    % Var explained: 88.42
```

Choice of `mtry` makes little difference but is the only real tuning parameter.

Variable “importance”

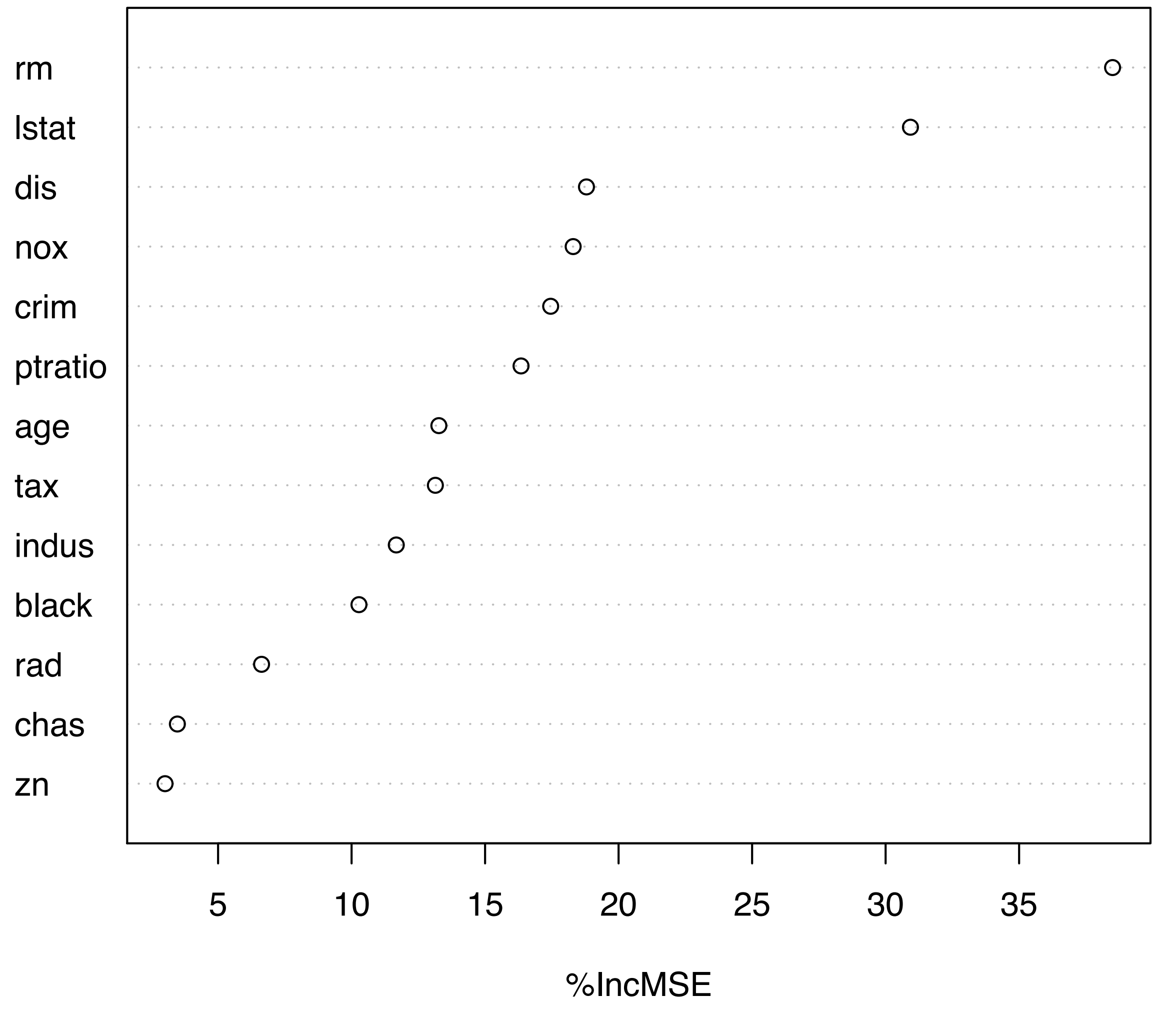
Despite the better predictive performance, single trees seem to have an edge over tree ensembles in terms of interpretability.
How do you interpret a forest of trees ?
Idea: denote by $\hat{e}$ the out-of bag estimate of the loss when using the original data samples.
For each variable $k \in \{1, \dots, p\}$,

- permute randomly the k-th predictor variable to generate a new set of samples $(\tilde{X}_1, Y_1), \dots, (\tilde{X}_n, Y_n)$.
- compute the out-of-bag estimate $\hat{e}_k$ of the prediction error with these new samples.

A measure of importance of variable k is then $\hat{e}_k - \hat{e}$, the increase in error rate due to random permutation of the k-th variable.

Example for Boston Housing data.

```
rf <- randomForest(x,y,importance=TRUE)
varImpPlot(rf)
```



Random Forests can be seen as an adaptive nearest neighbour technique. Let $P(x, x_i) \in [0, 1]$ be the proportion of trees for which an observation x falls into the same final leaf node as the original observation x_i. If every leaf node contains the same number of observations, the prediction of Random Forests (in regression mode) at predictor x is

$$\hat{Y}^{RF}(x) = \frac{\sum_{i=1}^{n} P(x, x_i) Y_i}{\sum_{i=1}^{n} P(x, x_i)},$$

which is a weighted (adaptive) nearest neighbour scheme and the weights are proportional to the proximities $P(x, x_i)$.
If the nodes contain different number of original observations, $P(x, x_i)$ is the weighted proportion of trees where x and x_i fall into the same leaf node, and weights are inversely proportional for each tree to the number of samples in the leaf node where x_i falls into.
For classification, the prediction will be the weighted majority vote, where again weights are proportional to the proximities $P(x, x_i)$.

Can visualize weights $P(x_i, x_j)$ for example by MDS.
Use Glass dataset as example.

```
> library(MASS)
> data(Glass)
> Glass[1:10,]
        RI    Na   Mg   Al    Si    K   Ca Ba   Fe Type
1  1.52101 13.64 4.49 1.10 71.78 0.06 8.75  0 0.00    1
2  1.51761 13.89 3.60 1.36 72.73 0.48 7.83  0 0.00    1
3  1.51618 13.53 3.55 1.54 72.99 0.39 7.78  0 0.00    1
4  1.51766 13.21 3.69 1.29 72.61 0.57 8.22  0 0.00    1
5  1.51742 13.27 3.62 1.24 73.08 0.55 8.07  0 0.00    1
6  1.51596 12.79 3.61 1.62 72.97 0.64 8.07  0 0.26    1
7  1.51743 13.30 3.60 1.14 73.09 0.58 8.17  0 0.00    1
8  1.51756 13.15 3.61 1.05 73.24 0.57 8.24  0 0.00    1
9  1.51918 14.04 3.58 1.37 72.08 0.56 8.30  0 0.00    1
10 1.51755 13.00 3.60 1.36 72.99 0.57 8.40  0 0.11    1
```

Try to predict glass type, based on chemical composition.

```
> X <- Glass[,-10]
> Y <- Glass[,10]
> rf <- randomForest(X,Y,ntree=500,proximity=TRUE,oob.prox=TRUE)
```

Calculate the proximities $P(x_i, x_j)$ based on out-of-bag observations.

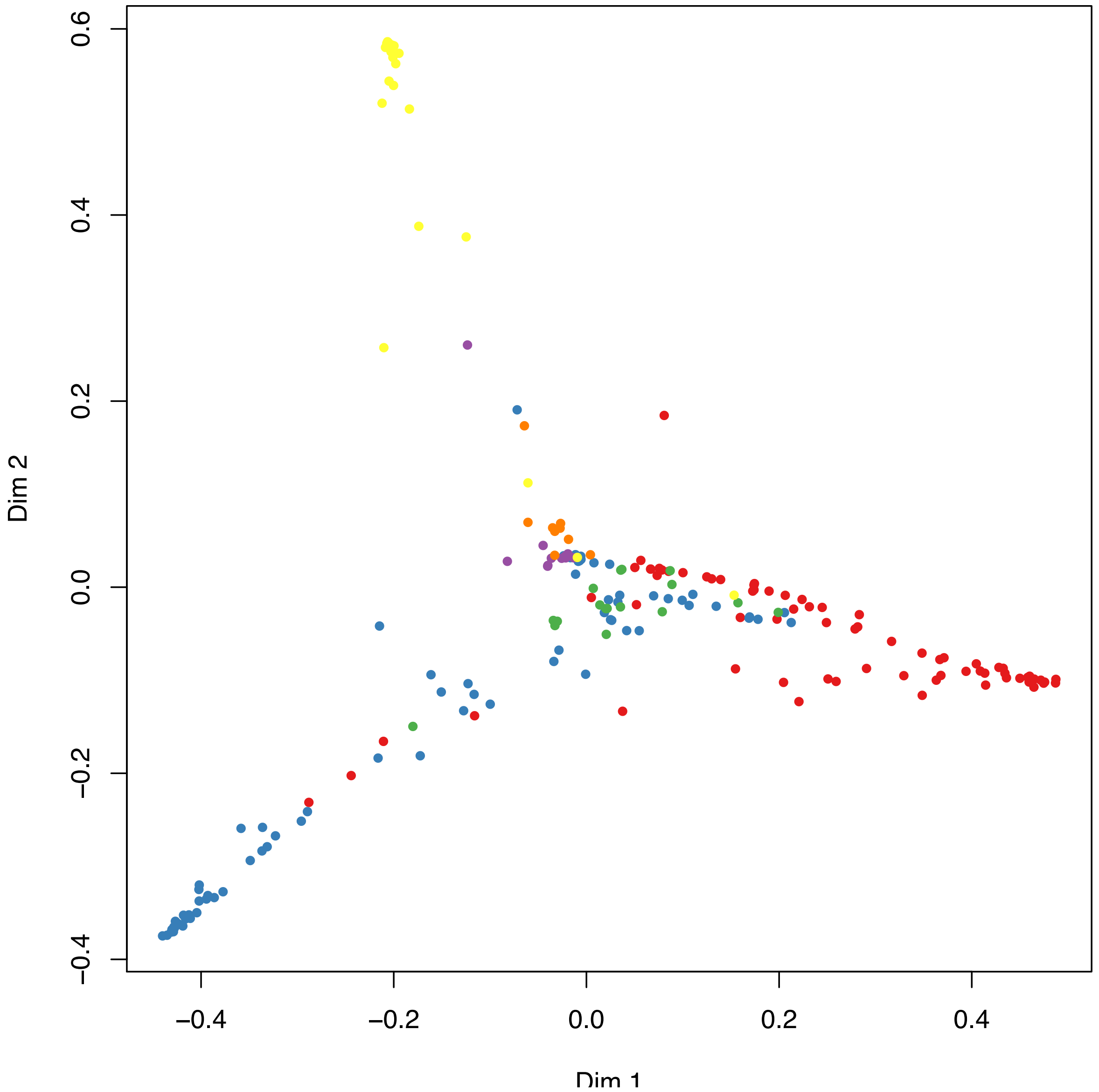
```
> rf

Call:
 randomForest(x = X, y =  Y, proximity = TRUE, oob.prox = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 20.09%
Confusion matrix:
   1  2 3 4 5  6 class.error
1 63  6 1 0 0  0   0.1000000
2  9 60 2 2 2  1   0.2105263
3  7  3 7 0 0  0   0.5882353
4  0  3 0 9 0  1   0.3076923
5  0  2 0 0 7  0   0.2222222
6  1  3 0 0 0 25   0.1379310
```

Visualize proximities $P(x_i, x_j)$ for $i, j = 1, \ldots, n$ by MDS, using as distance matrix $D = 1 - P$.

```
> MDSplot(rf,Y)
```


Dim 2
Dim 1