Classification and tree-based methods

Introduction

Terminology

- We will consider Classification and regression trees (CART) as well as the related Random Forests
- These are methods of supervised learning (*"labelled"* training data) for:
  - Classification
  - Regression
- We focus on classification

Classification

- Given a feature vector x and a qualitative response Y taking values in the set C, the classification task is to build a function $f(x)$ that takes as input the feature vector x and predicts its value for Y; i.e. $f(x) \in C$
- Often: interested in estimating the probabilities that X belongs to each category in C

Many methods for classification:

- Logistic regression
- Classification (and regression) trees
- Random Forest
- Nearest Neighbours
- Naive Bayes
- Support Vector Machines (SVM)
- Neural Networks
- ...

Classification trees

One limitation of logistic regression (and multinomial regression) is the need for a statistical model, with e.g. the assumption logit-linear relationships, which typically is the biggest challenge - the relationship between the probability of the outcome and the explanatory variables may be anything but linear on the (arbitrary) logit-scale.

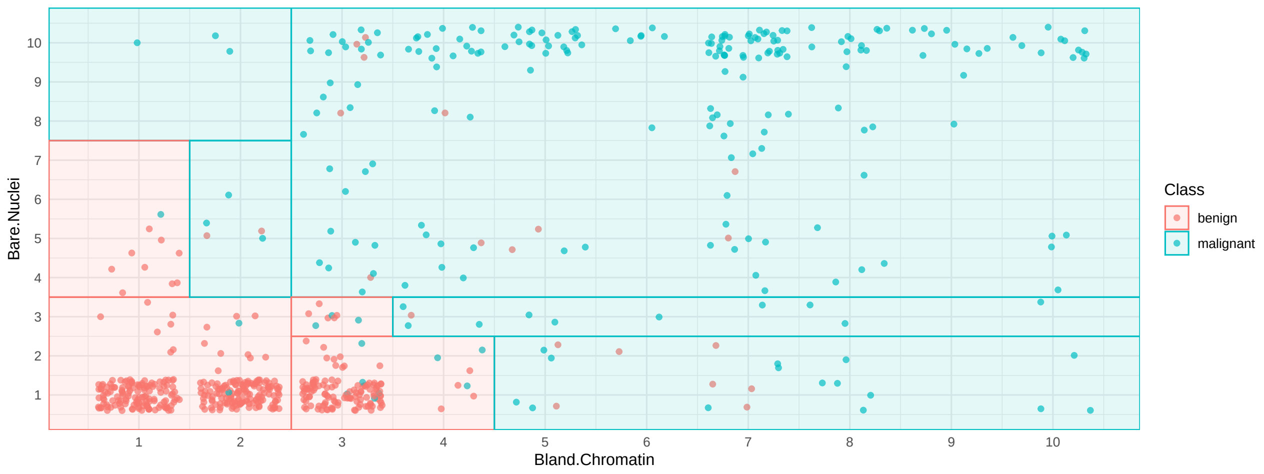
A very generic, assumption free, adaptable and flexible class of models are the classification and regression trees. The seminal CART-book (Classification And Regression Trees) from 1984 by Beirman et al layed much of the foundation for their success.

Partition concept with small example

```
#' Plot uses non-standard package installable from r-universe.dev:
#' install.packages('parttree', repos = 'https://grantmcdermott.r-universe.dev')
BCfull <- na.omit(OneR::breastcancer) |>
  as_tibble(.name_repair = "universal")
```

```
New names:
* `Clump Thickness` -> `Clump.Thickness`
* `Uniformity of Cell Size` -> `Uniformity.of.Cell.Size`
* `Uniformity of Cell Shape` -> `Uniformity.of.Cell.Shape`
* `Marginal Adhesion` -> `Marginal.Adhesion`
* `Single Epithelial Cell Size` -> `Single.Epithelial.Cell.Size`
* `Bare Nuclei` -> `Bare.Nuclei`
* `Bland Chromatin` -> `Bland.Chromatin`
* `Normal Nucleoli` -> `Normal.Nucleoli`
```

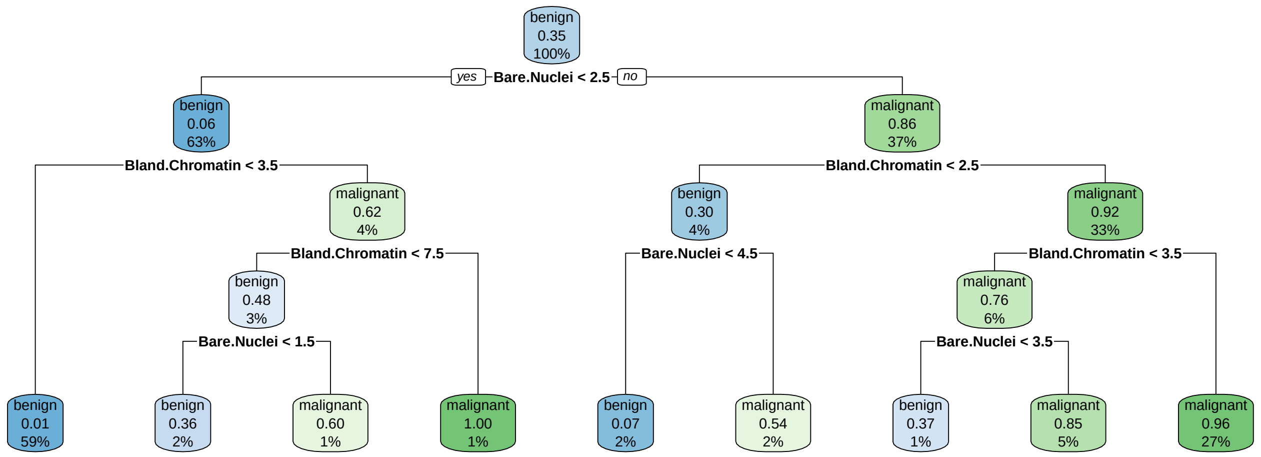
```
BC_first_tree <- rpart::rpart(Class ~ Bland.Chromatin + Bare.Nuclei, data = BCfull, cp = 0)
BCfull |>
  ggplot(aes(x = Bland.Chromatin, y = Bare.Nuclei, color = Class)) +
  geom_jitter(alpha=0.7) +
  parttree::geom_parttree(data = BC_first_tree, aes(fill=Class), alpha = 0.1) +
  theme_minimal() +
  scale_y_continuous(n.breaks = 10) +
  scale_x_continuous(n.breaks = 10)
```



```
library(rpart.plot)
```

```
Loading required package: rpart
```

```
rpart.plot(BC_first_tree, roundint = FALSE)
```



---

rpart

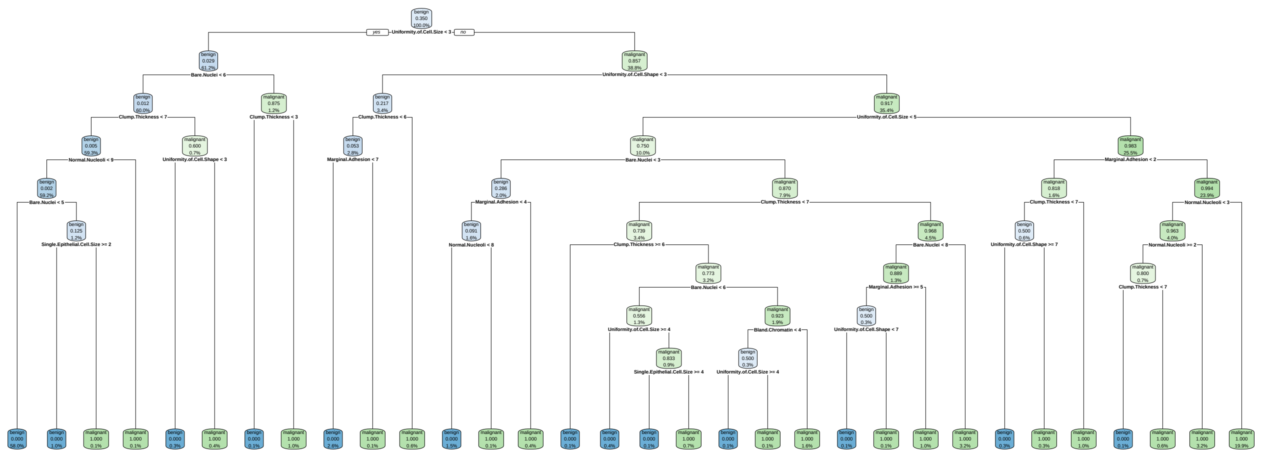
For plotting the `rpart.plot` package is excellent: http://www.milbo.org/rpart-plot/prp.pdf

For illustration purposes we set parameters to get a big tree which is clearly over-fitting the data

```r
library(rpart)
set.seed(42)
BC_rpart <- rpart(Class ~ ., data = BCfull, cp = 0, minsplit = 2, minbucket = 1)

BC_rpart
```

```
n= 683

node), split, n, loss, yval, (yprob)
      * denotes terminal node

  1) root 683 239 benign (0.650073206 0.349926794)
    2) Uniformity.of.Cell.Size< 2.5 418  12 benign (0.971291866 0.028708134)
      4) Bare.Nuclei< 5.5 410   5 benign (0.987804878 0.012195122)
        8) Clump.Thickness< 6.5 405   2 benign (0.995061728 0.004938272)
         16) Normal.Nucleoli< 9 404   1 benign (0.997524752 0.002475248)
           32) Bare.Nuclei< 4.5 396   0 benign (1.000000000 0.000000000) *
           33) Bare.Nuclei>=4.5 8   1 benign (0.875000000 0.125000000)
             66) Single.Epithelial.Cell.Size>=1.5 7   0 benign (1.000000000 0.000000000) *
             67) Single.Epithelial.Cell.Size< 1.5 1   0 malignant (0.000000000 1.000000000) *
         17) Normal.Nucleoli>=9 1   0 malignant (0.000000000 1.000000000) *
        9) Clump.Thickness>=6.5 5   2 malignant (0.400000000 0.600000000)
         18) Uniformity.of.Cell.Shape< 2.5 2   0 benign (1.000000000 0.000000000) *
         19) Uniformity.of.Cell.Shape>=2.5 3   0 malignant (0.000000000 1.000000000) *
      5) Bare.Nuclei>=5.5 8   1 malignant (0.125000000 0.875000000)
       10) Clump.Thickness< 2.5 1   0 benign (1.000000000 0.000000000) *
       11) Clump.Thickness>=2.5 7   0 malignant (0.000000000 1.000000000) *
    3) Uniformity.of.Cell.Size>=2.5 265  38 malignant (0.143396226 0.856603774)
      6) Uniformity.of.Cell.Shape< 2.5 23   5 benign (0.782608696 0.217391304)
       12) Clump.Thickness< 5.5 19   1 benign (0.947368421 0.052631579)
         24) Marginal.Adhesion< 7 18   0 benign (1.000000000 0.000000000) *
         25) Marginal.Adhesion>=7 1   0 malignant (0.000000000 1.000000000) *
       13) Clump.Thickness>=5.5 4   0 malignant (0.000000000 1.000000000) *
      7) Uniformity.of.Cell.Shape>=2.5 242  20 malignant (0.082644628 0.917355372)
       14) Uniformity.of.Cell.Size< 4.5 68  17 malignant (0.250000000 0.750000000)
         28) Bare.Nuclei< 2.5 14   4 benign (0.714285714 0.285714286)
           56) Marginal.Adhesion< 3.5 11   1 benign (0.909090909 0.090909091)
            112) Normal.Nucleoli< 7.5 10   0 benign (1.000000000 0.000000000) *
            113) Normal.Nucleoli>=7.5 1   0 malignant (0.000000000 1.000000000) *
           57) Marginal.Adhesion>=3.5 3   0 malignant (0.000000000 1.000000000) *
         29) Bare.Nuclei>=2.5 54   7 malignant (0.129629630 0.870370370)
           58) Clump.Thickness< 6.5 23   6 malignant (0.260869565 0.739130435)
            116) Clump.Thickness>=5.5 1   0 benign (1.000000000 0.000000000) *
            117) Clump.Thickness< 5.5 22   5 malignant (0.227272727 0.772727273)
              234) Bare.Nuclei< 6 9   4 malignant (0.444444444 0.555555556)
                468) Uniformity.of.Cell.Size>=3.5 3   0 benign (1.000000000 0.000000000) *
                469) Uniformity.of.Cell.Size< 3.5 6   1 malignant (0.166666667 0.833333333)
                  938) Single.Epithelial.Cell.Size>=3.5 1   0 benign (1.000000000 0.000000000) *
                  939) Single.Epithelial.Cell.Size< 3.5 5   0 malignant (0.000000000 1.000000000) *
              235) Bare.Nuclei>=6 13   1 malignant (0.076923077 0.923076923)
                470) Bland.Chromatin< 3.5 2   1 benign (0.500000000 0.500000000)
                  940) Uniformity.of.Cell.Size>=3.5 1   0 benign (1.000000000 0.000000000) *
                  941) Uniformity.of.Cell.Size< 3.5 1   0 malignant (0.000000000 1.000000000) *
                471) Bland.Chromatin>=3.5 11   0 malignant (0.000000000 1.000000000) *
           59) Clump.Thickness>=6.5 31   1 malignant (0.032258065 0.967741935)
            118) Bare.Nuclei< 7.5 9   1 malignant (0.111111111 0.888888889)
              236) Marginal.Adhesion>=4.5 2   1 benign (0.500000000 0.500000000)
                472) Uniformity.of.Cell.Shape< 7 1   0 benign (1.000000000 0.000000000) *
                473) Uniformity.of.Cell.Shape>=7 1   0 malignant (0.000000000 1.000000000) *
              237) Marginal.Adhesion< 4.5 7   0 malignant (0.000000000 1.000000000) *
            119) Bare.Nuclei>=7.5 22   0 malignant (0.000000000 1.000000000) *
       15) Uniformity.of.Cell.Size>=4.5 174   3 malignant (0.017241379 0.982758621)
         30) Marginal.Adhesion< 1.5 11   2 malignant (0.181818182 0.818181818)
           60) Clump.Thickness< 7 4   2 benign (0.500000000 0.500000000)
            120) Uniformity.of.Cell.Shape>=6.5 2   0 benign (1.000000000 0.000000000) *
            121) Uniformity.of.Cell.Shape< 6.5 2   0 malignant (0.000000000 1.000000000) *
           61) Clump.Thickness>=7 7   0 malignant (0.000000000 1.000000000) *
         31) Marginal.Adhesion>=1.5 163   1 malignant (0.006134969 0.993865031)
           62) Normal.Nucleoli< 2.5 27   1 malignant (0.037037037 0.962962963)
            124) Normal.Nucleoli>=1.5 5   1 malignant (0.200000000 0.800000000)
              248) Clump.Thickness< 6.5 1   0 benign (1.000000000 0.000000000) *
              249) Clump.Thickness>=6.5 4   0 malignant (0.000000000 1.000000000) *
            125) Normal.Nucleoli< 1.5 22   0 malignant (0.000000000 1.000000000) *
           63) Normal.Nucleoli>=2.5 136   0 malignant (0.000000000 1.000000000) *
```

```
rpart.plot(BC_rpart, digits = 3)
```

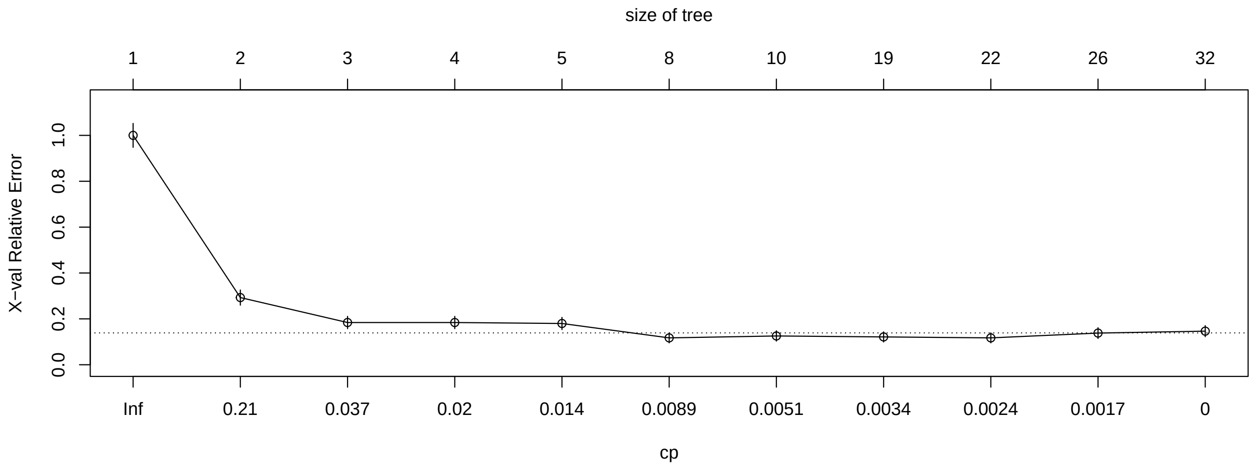


Complexity?

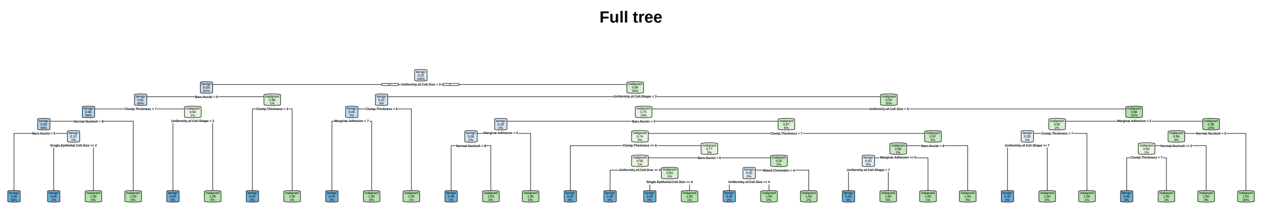
```r
printcp(BC_rpart)
```

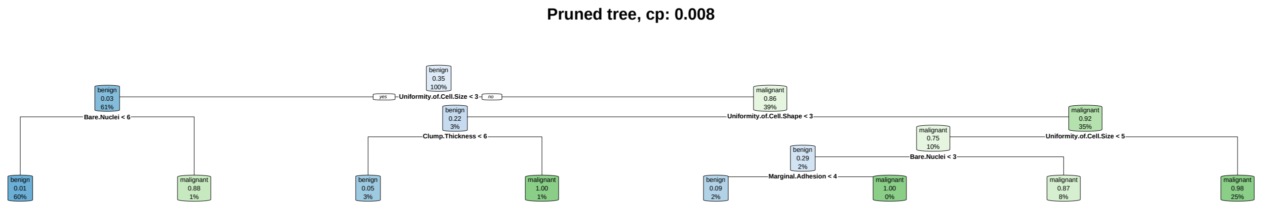
```

Classification tree:
rpart(formula = Class ~ ., data = BCfull, cp = 0, minsplit = 2,
    minbucket = 1)

Variables actually used in tree construction:
[1] Bare.Nuclei                 Bland.Chromatin
[3] Clump.Thickness             Marginal.Adhesion
[5] Normal.Nucleoli             Single.Epithelial.Cell.Size
[7] Uniformity.of.Cell.Shape    Uniformity.of.Cell.Size

Root node error: 239/683 = 0.34993

n= 683

          CP nsplit rel error  xerror     xstd
1  0.7907950      0 1.0000000 1.00000 0.052153
2  0.0543933      1 0.2092050 0.29289 0.033164
3  0.0251046      2 0.1548117 0.18410 0.026845
4  0.0167364      3 0.1297071 0.18410 0.026845
5  0.0125523      4 0.1129707 0.17992 0.026559
6  0.0062762      7 0.0753138 0.11715 0.021682
7  0.0041841      9 0.0627615 0.12552 0.022408
8  0.0027894     18 0.0251046 0.12134 0.022049
9  0.0020921     21 0.0167364 0.11715 0.021682
10 0.0013947     25 0.0083682 0.13808 0.023448
11 0.0000000     31 0.0000000 0.14644 0.024111
```

```r
plotcp(BC_rpart)
```



```r
cp_value <- 0.008 ## from printcp/plotcp output
BC_rpart_prune <- prune(BC_rpart, cp = cp_value)

par(mfrow = c(2,1))
rpart.plot(BC_rpart, main = "Full tree")
rpart.plot(BC_rpart_prune, main = paste("Pruned tree, cp:", cp_value))
```





```r
par(mfrow = c(1,1))
```

Prediction

```r
BC_pred <- as_tibble(predict(BC_rpart_prune, type = "prob")) |>
  mutate(pred_class = ifelse(benign > .5, "benign", "malignant"))

BCfull |> select(Class) |> bind_cols(BC_pred) |>
  count(Class, pred_class)
```

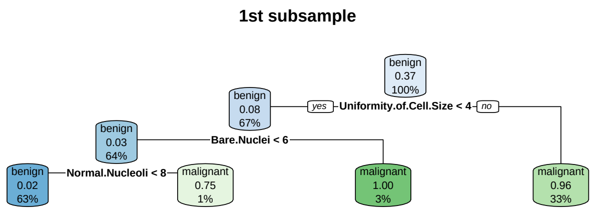
```
# A tibble: 4 x 3
  Class     pred_class      n
  <fct>     <chr>       <int>
1 benign    benign        433
2 benign    malignant      11
```

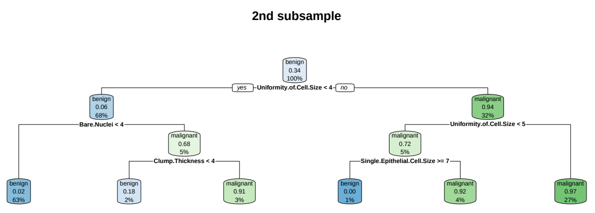
```
3 malignant benign        7
4 malignant malignant   232
```

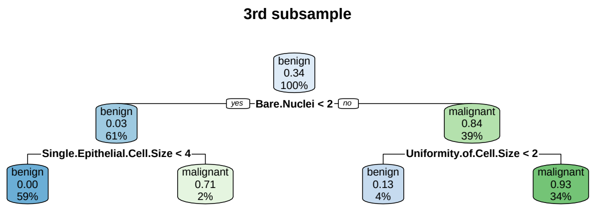
Flexibility

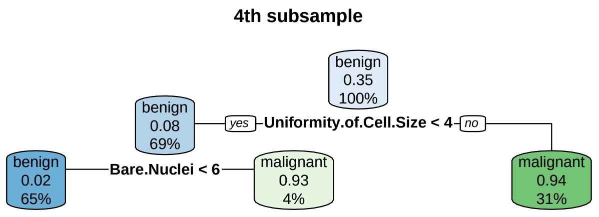
Trees can be very non-robust. In other words, a small change in the data can cause a large change in the final estimated tree

```r
set.seed(123)
N <- nrow(BCfull)
n_rep <- 4
row_index <- replicate(n_rep,
                       sample(N, size = N, replace = TRUE),
                       simplify = FALSE)

BC_resamp <-
  lapply(row_index, function(x) rpart(Class ~ ., data = BCfull, subset = x))

par(mfrow = c(2,2))
rpart.plot(BC_resamp[[1]], main = "1st subsample")
rpart.plot(BC_resamp[[2]], main = "2nd subsample")
rpart.plot(BC_resamp[[3]], main = "3rd subsample")
rpart.plot(BC_resamp[[4]], main = "4th subsample")
```









```r
par(mfrow = c(1,1))
```

Random forests

Random forest tries to remedy this flexibility of tree-based models by an ensemble of trees. The trees created on the previous slide are not identical, but still correlated in the sense that they use the same features/explanatory variables for splits and creating the trees. Furthermore, their predictions are highly correlated with each other.

```r
set.seed(123)
BC_resamp |> lapply( function(x) predict(x, newdata = BCfull, type = "class")) |> set_names(paste0("res
  bind_cols(Class = BCfull$Class) |>
  mutate(row = row_number()) |>
  relocate(row, Class) |>
  slice_sample(n = 10)
```

```
# A tibble: 10 x 6
     row Class     resample1 resample2 resample3 resample4
   <int> <fct>     <fct>     <fct>     <fct>     <fct>
 1   415 benign    benign    benign    benign    benign
 2   463 benign    benign    benign    benign    benign
 3   179 malignant malignant malignant malignant malignant
 4   526 benign    benign    benign    benign    benign
 5   195 malignant malignant malignant malignant malignant
 6   118 benign    benign    benign    malignant benign
 7   299 benign    benign    benign    benign    benign
 8   229 benign    benign    benign    benign    benign
 9   244 malignant malignant malignant malignant malignant
10    14 benign    benign    benign    benign    benign
```

Uncorrelated trees

The success of random forest relies on the simple fact that for uncorrelated quantities, the average is a consistent and unbiased estimator with a variance going to zero as 1/#trees.

Because of the bootstrapping of the data, the non-included observations (called out-of-bag samples) are used in random forests to assess the accuracy of the model.

Package `randomForest`

```r
library(randomForest)
```

```
randomForest 4.7-1.1

Type rfNews() to see new features/changes/bug fixes.


Attaching package: 'randomForest'

The following object is masked from 'package:dplyr':

    combine

The following object is masked from 'package:ggplot2':

    margin
```

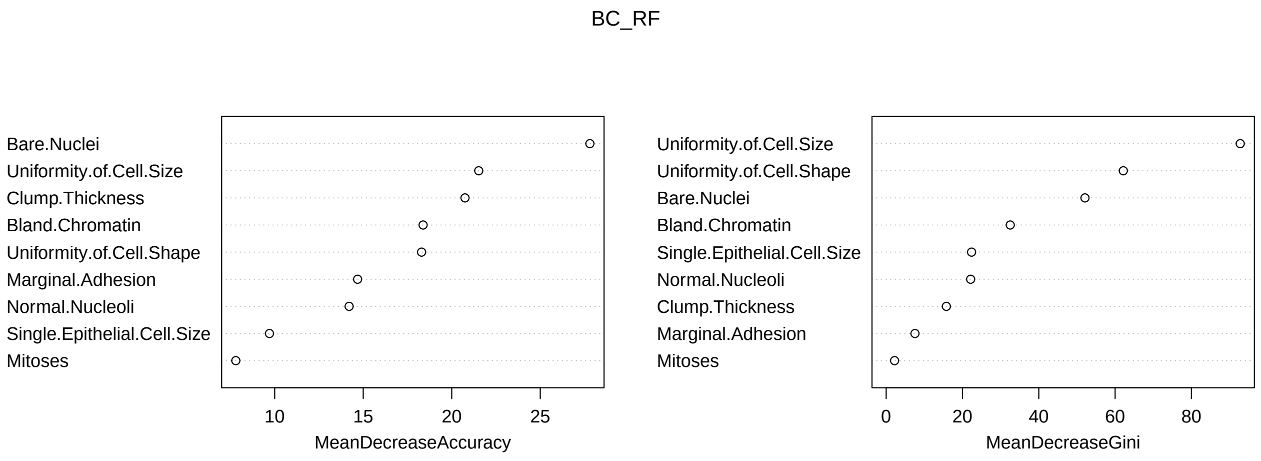
```r
set.seed(1234)

BC_RF <- randomForest(Class ~ ., data = BCfull, importance=TRUE)
BC_RF
```

```

Call:
 randomForest(formula = Class ~ ., data = BCfull, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
```

```
No. of variables tried at each split: 3

        OOB estimate of  error rate: 2.78%
Confusion matrix:
          benign malignant class.error
benign       432        12  0.02702703
malignant      7       232  0.02928870
```

Variable importance for RF

```
varImpPlot(BC_RF)
```

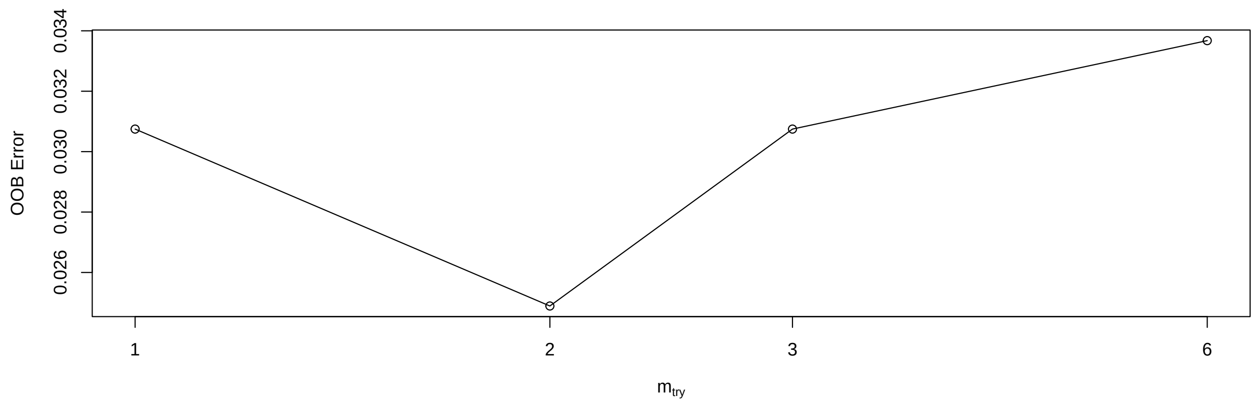


Tuning of algorithm/hyper parameters of RF

Tuning on `mtry` with `ntree` fixed:

```
BC_RF_tune <- tuneRF(y = BCfull$Class, x = select(BCfull, -Class), improve = 0.001)
```

```
mtry = 3  OOB error = 3.07%
Searching left ...
mtry = 2    OOB error = 2.49%
0.1904762 0.001
mtry = 1    OOB error = 3.07%
-0.2352941 0.001
Searching right ...
mtry = 6    OOB error = 3.37%
-0.3529412 0.001
```

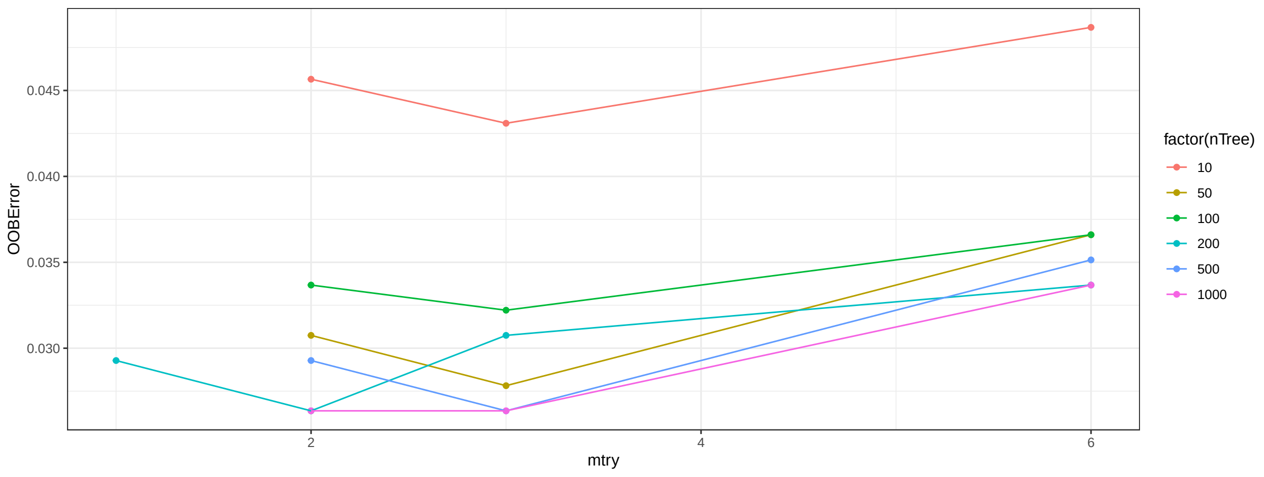


Tuning both `mtry` and `ntree`

```r
tune_one <- function(n){
  output <- tuneRF(y = BCfull$Class, x = select(BCfull, -Class),
                   ntreeTry = n, improve = 0.05, trace = FALSE, plot = FALSE)
  return(bind_cols(nTree = n, output))
}
nTree <- c(10, 50, 100, 200, 500, 1000)
OOB_error <- map(nTree, tune_one)
```

```
-0.05951958 0.05
-0.1295392 0.05
-0.1052632 0.05
-0.3157895 0.05
-0.04545455 0.05
-0.1363636 0.05
0.1428571 0.05
-0.1111111 0.05
-0.2777778 0.05
-0.1111111 0.05
-0.3333333 0.05
0 0.05
-0.2777778 0.05
```

```r
OOB_error |>
  bind_rows() |>
  ggplot(aes(x = mtry, y = OOBError, colour = factor(nTree))) +
  geom_point() + geom_line()
```

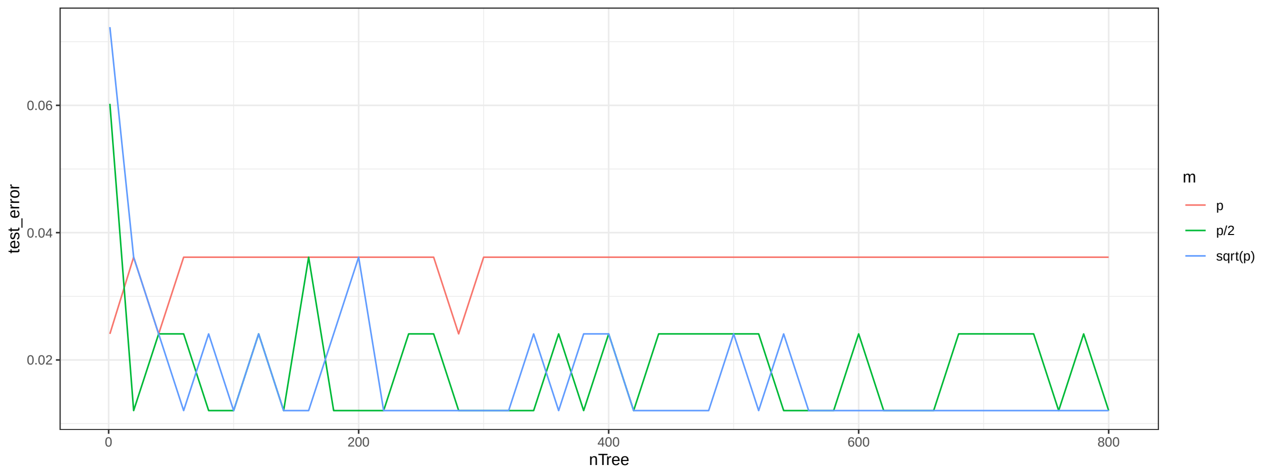


Test and training set for RF

We can also use the `test` arguments of `randomForest` in order to assess the accuracy on a test set while fitting the model.

```r
train_id <- sample(nrow(BCfull), 600)

BC_train <- BCfull[train_id,]
BC_test <- BCfull[-train_id,]

BC_RF <- randomForest(Class ~ ., data = BC_train, importance=TRUE,
                      xtest = select(BC_test, - Class), ytest = BC_test$Class)

RF_test_error <- function(rf){
  rf_conf <- rf$test$confusion
  1 - sum(diag(rf_conf))/sum(rf_conf[,-ncol(rf_conf)])
}

p <- ncol(BC_train)-1

BC_RF_test <-
  expand_grid(
    nTree = c(1, seq(from = 20, to = 800, by = 20)),
    mTry = c(p, p/2, sqrt(p))) |>
  mutate(
    test_error = map2_dbl(.x = nTree, .y = mTry, \(n,m) RF_test_error(
      randomForest(Class ~ ., data = BC_train, importance=FALSE, ntree = n, mtry = m,
                   xtest = select(BC_test, - Class), ytest = BC_test$Class)
      ))
)

BC_RF_test |>
  mutate(
    m = case_when(mTry == p ~ "p",
                  mTry == p/2 ~ "p/2",
                  TRUE ~ "sqrt(p)"),
    ) |>
```

```r
ggplot(aes(nTree, test_error, colour = m)) +
geom_line()
```


test_error
nTree
m
p
p/2
sqrt(p)