Tree-based models and their interpretation

Anna Gottard

29/05/2019

CART with rpart

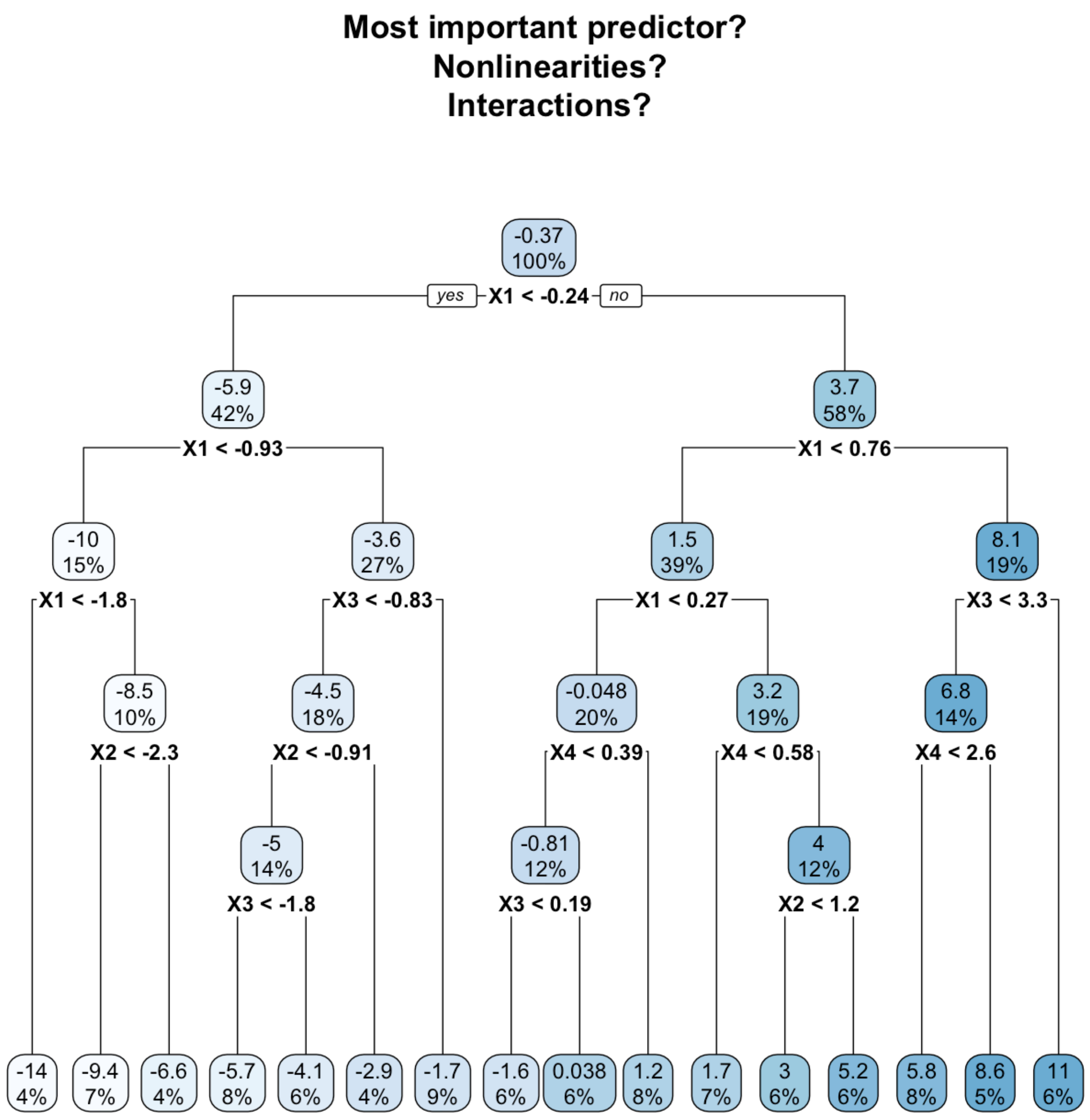
Let us start briefly on how to **grow a tree** with the CART algorithm

Several packages run the CART algorithm, **rpart** is the most used

Suppose we have a response Y and four predictors X_1, X_2, X_3 and X_4, continuous

```
library(rpart)
T0 <- rpart(Y~X1+X2+X3+X4, control = rpart.control(cp = 0))
```

```
rpart.plot::rpart.plot(T0, main="Most important predictor?\n Nonlinearities?\n Int
eractions?", type = 2)
```



```
# to see all: summary(T0)

names(T0)
```

```
##  [1] "frame"               "where"               "call"
##  [4] "terms"               "cptable"             "method"
##  [7] "parms"               "control"             "functions"
## [10] "numresp"             "splits"              "variable.importance"
## [13] "y"                   "ordered"
```

```
T0$frame
```

```
##        var   n  wt         dev         yval  complexity ncompete nsurrogate
## 1       X1 200 200 7787.989049  -0.37229396 0.577484482        3          3
## 2       X1  84  84 1478.326262  -5.94488508 0.110604291        3          3
## 4       X1  30  30  352.755558 -10.24119166 0.028100886        3          3
## 8  <leaf>   9   9   69.584451 -14.36691820 0.000000000        0          0
## 9       X2  21  21   64.321714  -8.47302314 0.004511910        3          3
## 18 <leaf>  14  14    6.989832  -9.38770116 0.000000000        0          0
## 19 <leaf>   7   7   22.193175  -6.64366710 0.000000000        0          0
## 5       X3  54  54  264.185695  -3.55804810 0.012553935        3          3
## 10      X2  36  36  124.826200  -4.50950851 0.003786420        3          1
## 20      X3  27  27   80.284943  -5.03204329 0.002056752        3          3
## 40 <leaf>  16  16   18.953791  -5.67068626 0.000000000        0          0
## 41 <leaf>  11  11   45.313187  -4.10310806 0.000000000        0          0
## 21 <leaf>   9   9   15.052658  -2.94190418 0.000000000        0          0
## 11 <leaf>  18  18   41.589587  -1.65512727 0.000000000        0          0
## 3       X1 116 116 1812.219969   3.66303065 0.141667204        3          3
## 6       X1  78  78  420.418177   1.51043316 0.025618827        3          3
## 12      X4  40  40  104.874323  -0.04842569 0.004979975        3          2
## 24      X3  25  25   47.794629  -0.81115750 0.002136899        3          3
## 48 <leaf>  13  13   16.366814  -1.59504482 0.000000000        0          0
## 49 <leaf>  12  12   14.785666   0.03805375 0.000000000        0          0
## 25 <leaf>  15  15   18.295706   1.22279401 0.000000000        0          0
## 13      X4  38  38  116.024707   3.15133720 0.006237319        3          2
## 26 <leaf>  14  14   12.583885   1.67099728 0.000000000        0          0
## 27      X2  24  24   54.864650   4.01486882 0.003906680        3          3
## 54 <leaf>  13  13   16.415747   2.97916484 0.000000000        0          0
## 55 <leaf>  11  11    8.023720   5.23888261 0.000000000        0          0
## 7       X3  38  38  288.499157   8.08152024 0.018593992        3          3
## 14      X4  27  27   92.085286   6.83550988 0.006155940        3          3
## 28 <leaf>  17  17   25.330394   5.81350374 0.000000000        0          0
## 29 <leaf>  10  10   18.812501   8.57292032 0.000000000        0          0
## 15 <leaf>  11  11   51.604064  11.13990928 0.000000000        0          0
```

```
# Variable importance
T0$variable.importance
```

```
##       X1       X4       X2       X3
## 7068.945 5249.693 5085.050 4910.052
```

```
# In percentage
round(100*T0$variable.importance/sum(T0$variable.importance),2)
```

```
##    X1    X4    X2    X3
## 31.68 23.53 22.79 22.00
```

Data generating process:

Linear, diamond graph:

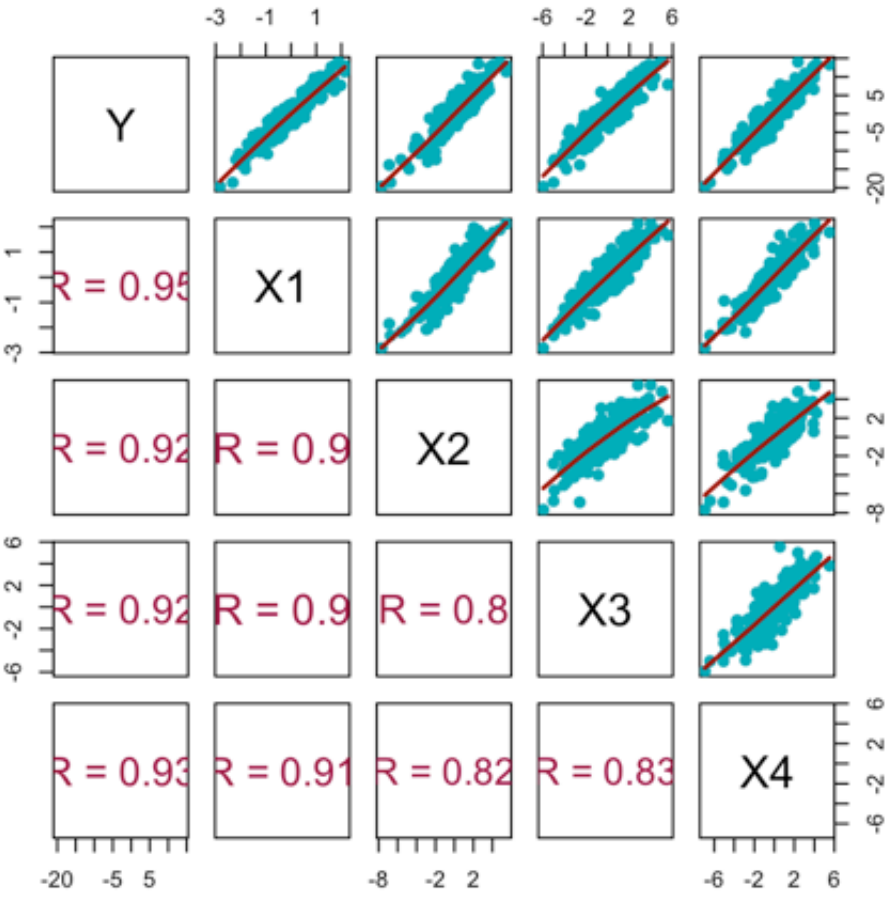
X1t <- rnorm(n)

X2t <- 2*X1t + rnorm(n)

X3t <- 2*X1t + rnorm(n)

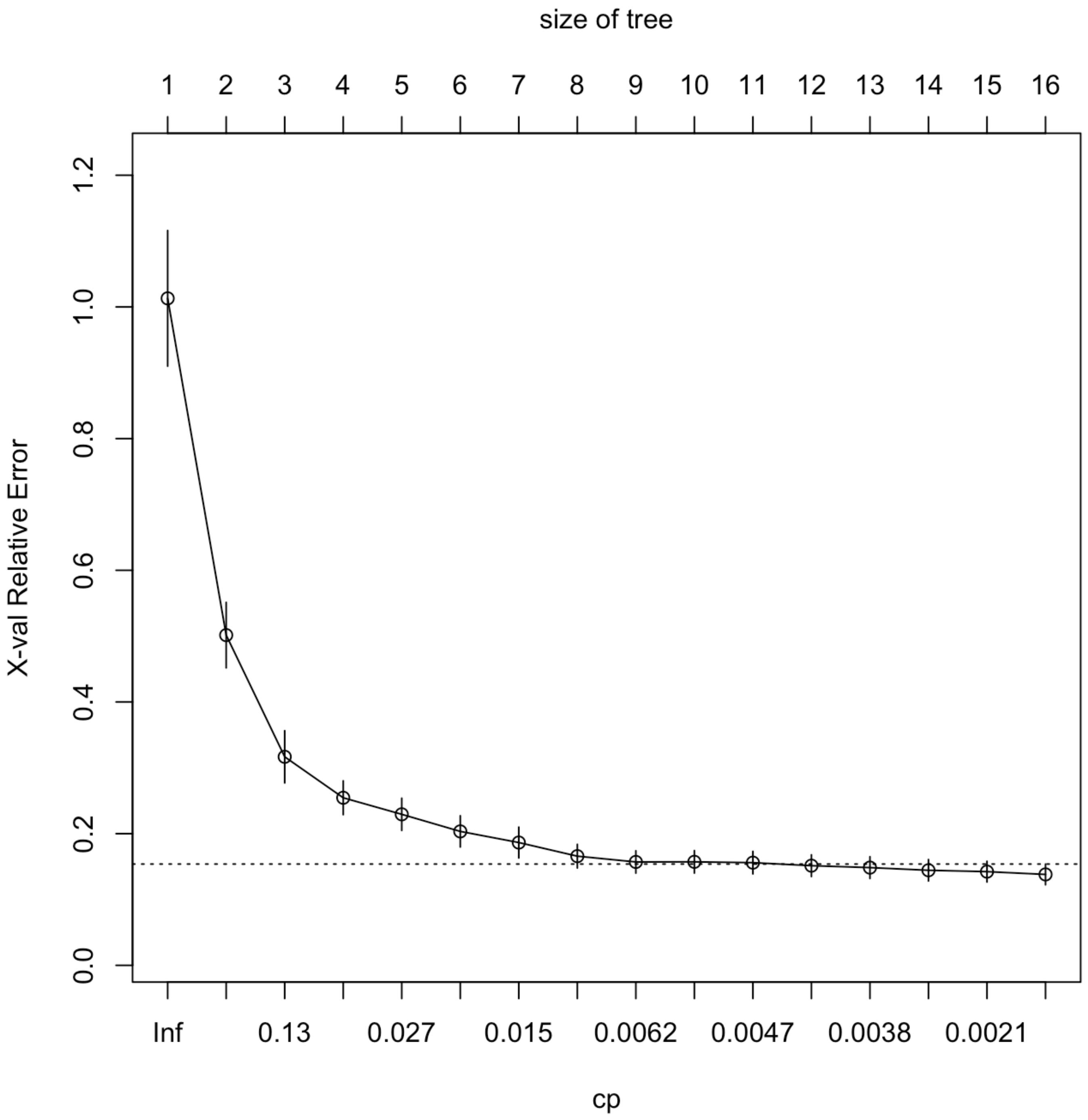
X4t <- 2*X1t + rnorm(n)

Yt <- 2*X2t + 2*X3t + 2*X4t + rnorm(n)



Pruning with crossvalidation

```
plotcp(T0)
```



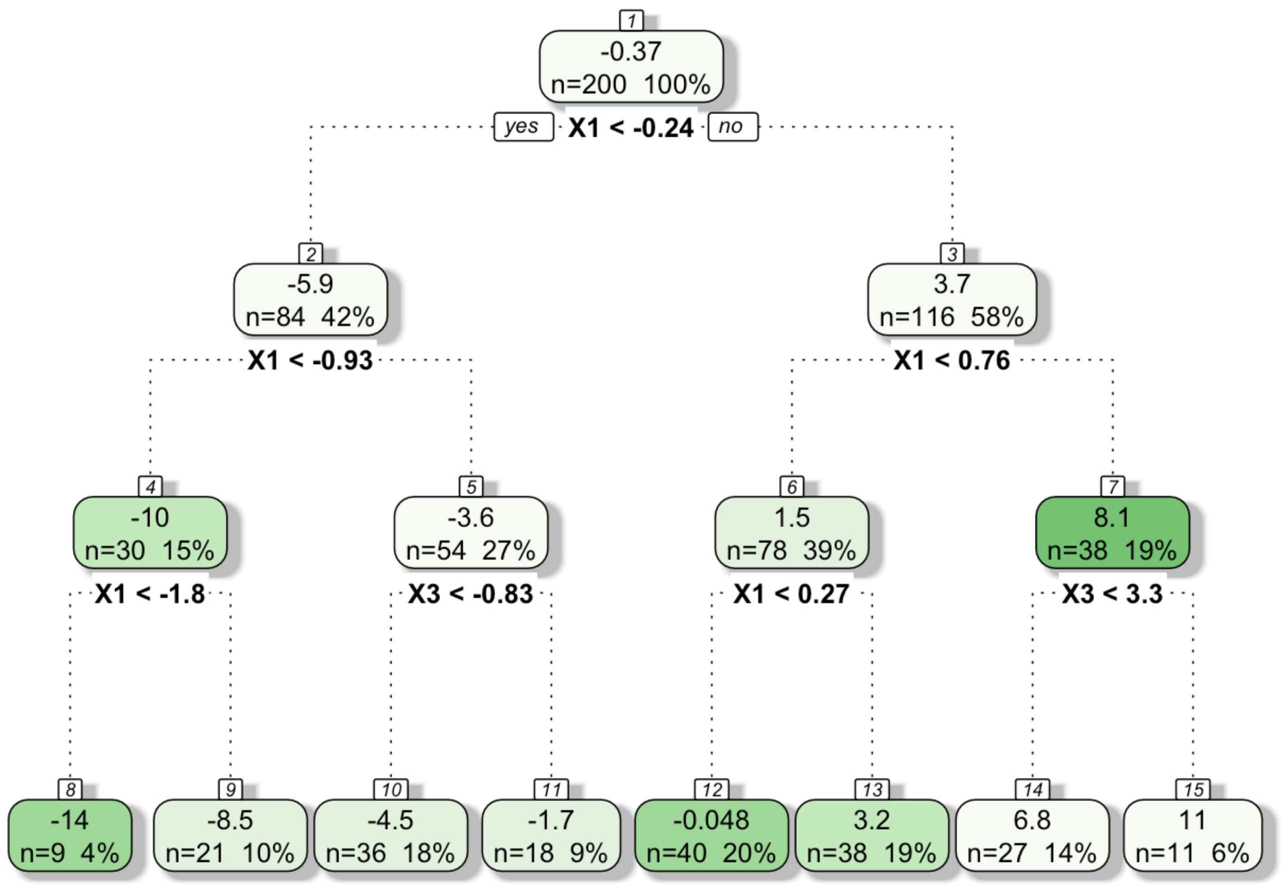
```
printcp(T0) # same as T0$cptable
```

```
##
## Regression tree:
## rpart(formula = Y ~ X1 + X2 + X3 + X4, control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] X1 X2 X3 X4
##
## Root node error: 7788/200 = 38.94
##
## n= 200
##
##            CP nsplit rel error  xerror     xstd
## 1  0.5774845      0  1.000000 1.01300 0.103001
## 2  0.1416672      1  0.422516 0.50151 0.049742
## 3  0.1106043      2  0.280848 0.31672 0.039687
## 4  0.0281009      3  0.170244 0.25451 0.025727
## 5  0.0256188      4  0.142143 0.22937 0.024489
## 6  0.0185940      5  0.116524 0.20343 0.023762
## 7  0.0125539      6  0.097930 0.18650 0.023468
## 8  0.0062373      7  0.085376 0.16581 0.017975
## 9  0.0061559      8  0.079139 0.15702 0.016962
## 10 0.0049800      9  0.072983 0.15719 0.017115
## 11 0.0045119     10  0.068003 0.15596 0.017149
## 12 0.0039067     11  0.063491 0.15136 0.016508
## 13 0.0037864     12  0.059585 0.14844 0.016546
## 14 0.0021369     13  0.055798 0.14439 0.016314
## 15 0.0020568     14  0.053661 0.14230 0.015795
## 16 0.0000000     15  0.051604 0.13809 0.015751
```

```
T0$cptable[which.min(T0$cptable[,"xerror"]),"CP"]
```

```
## [1] 0
```

```
T1 <- prune.rpart(T0,0.008)
rattle::fancyRpartPlot(T1)
```



Rattle 2019-May-28 23:05:15 annagottard

```
T0.pred<- predict(T0, data.test)
T1.pred<- predict(T1, data.test)

# # accuracy in the test set
mean((data.test$Yt - T0.pred)^2)
```

```
## [1] 209.4589
```

```
mean((data.test$Yt - T1.pred)^2)
```

```
## [1] 209.5773
```

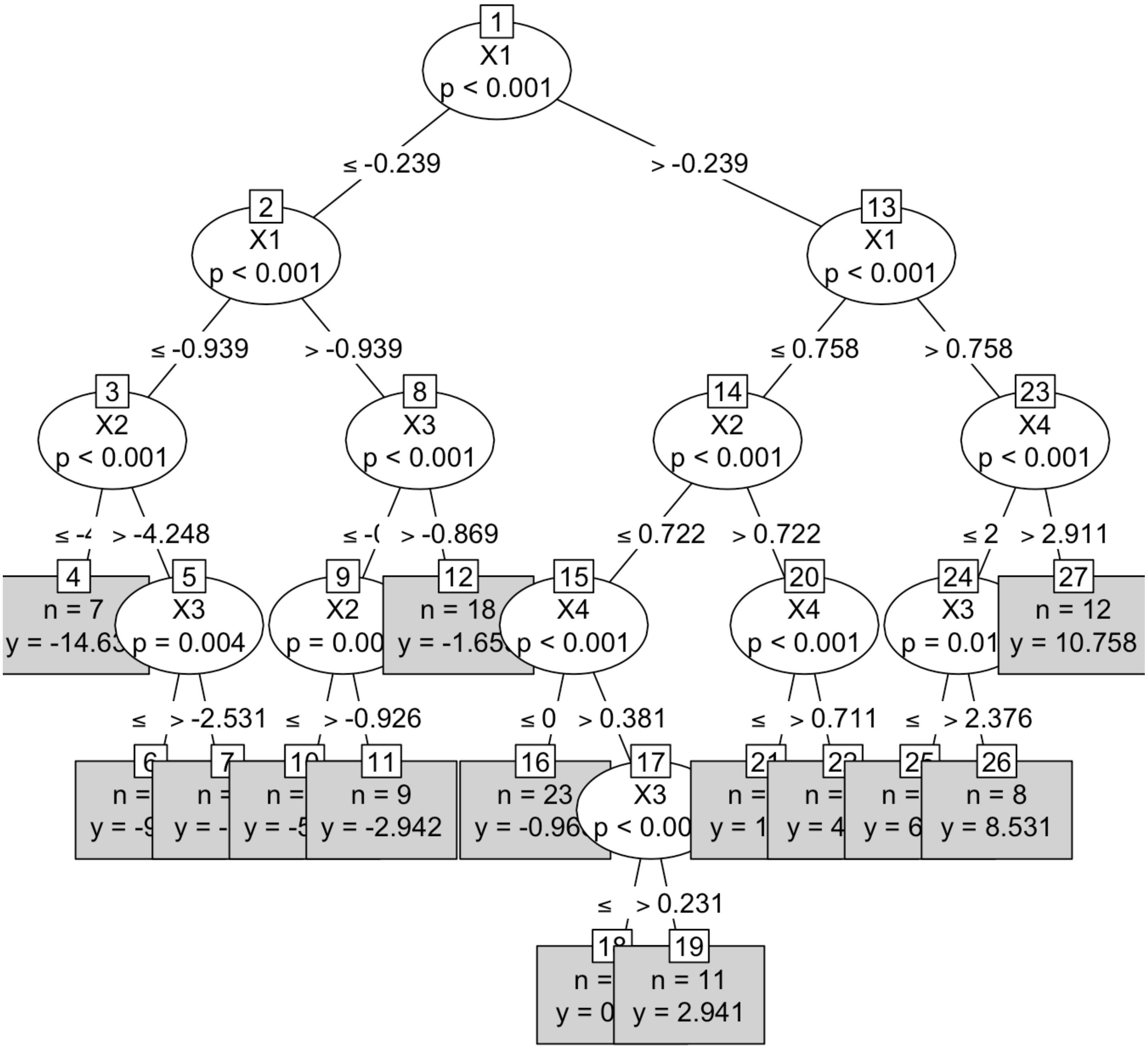
Conditional inference tree

We use the package *party*

Notice that party does not compute variable importance for a single tree but only for conditional inference random forest.

No need to prune!

```
library(party)
cT0 <- ctree(Y~X1+X2+X3+X4, data=data.train)
plot(cT0, type="simple")
```



```
# accuracy in the test set
mean((predict(cT0,data.test[,-1], OOB=TRUE) - data.test$Yt)^2)
```

```
## [1] 209.2108
```

Ensample methods

Random Forests with packacge randomForest

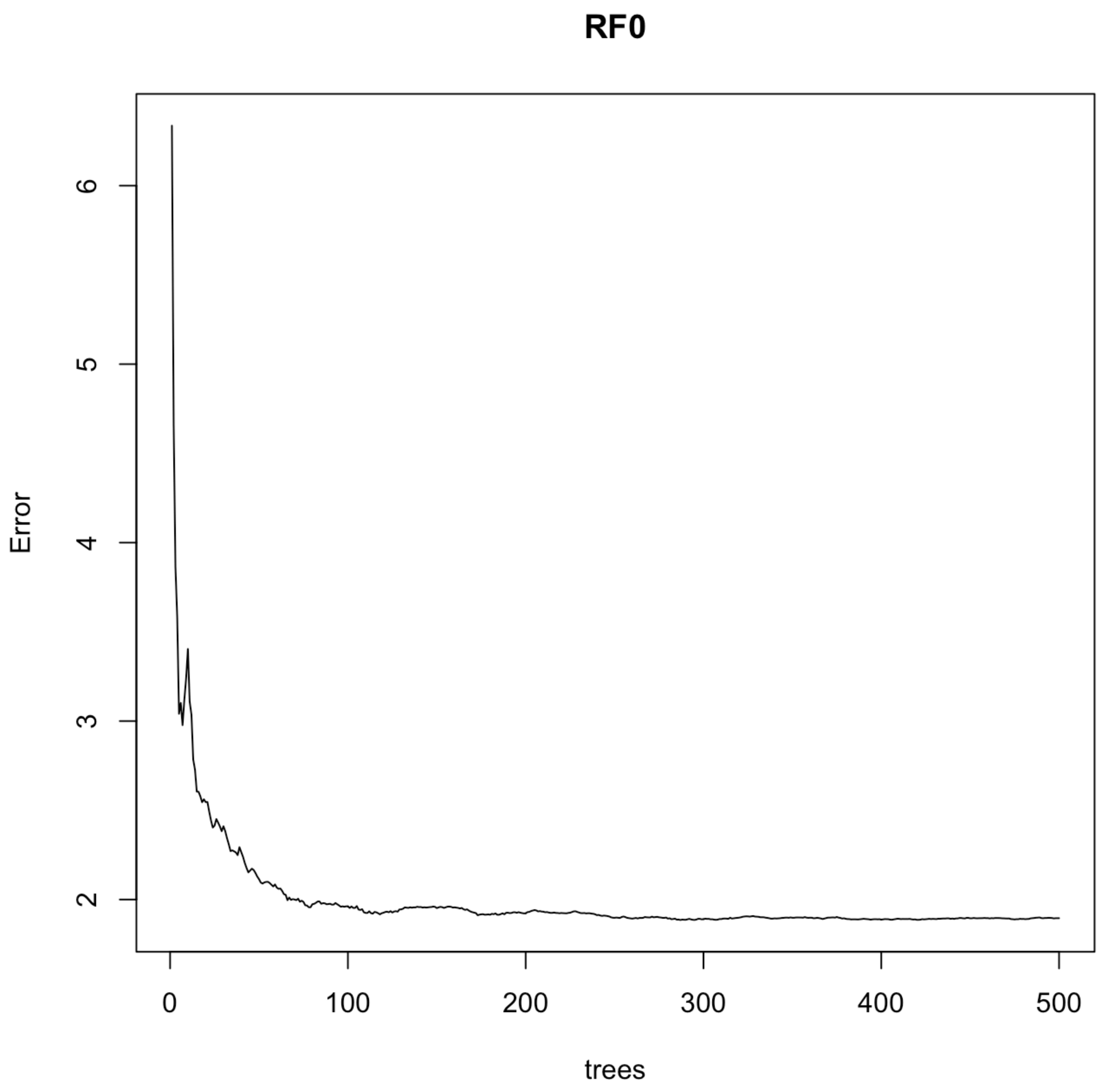
```
library(randomForest)
set.seed(123)
RF0 <- randomForest(Y~X1+X2+X3+X4, data=data.train)
print(RF0)
```

```
##
## Call:
##  randomForest(formula = Y ~ X1 + X2 + X3 + X4, data = data.train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 1.895498
##                     % Var explained: 95.13
```

```
importance(RF0)
```

```
##    IncNodePurity
## X1      2014.614
## X2      1872.000
## X3      1902.646
## X4      1852.613
```

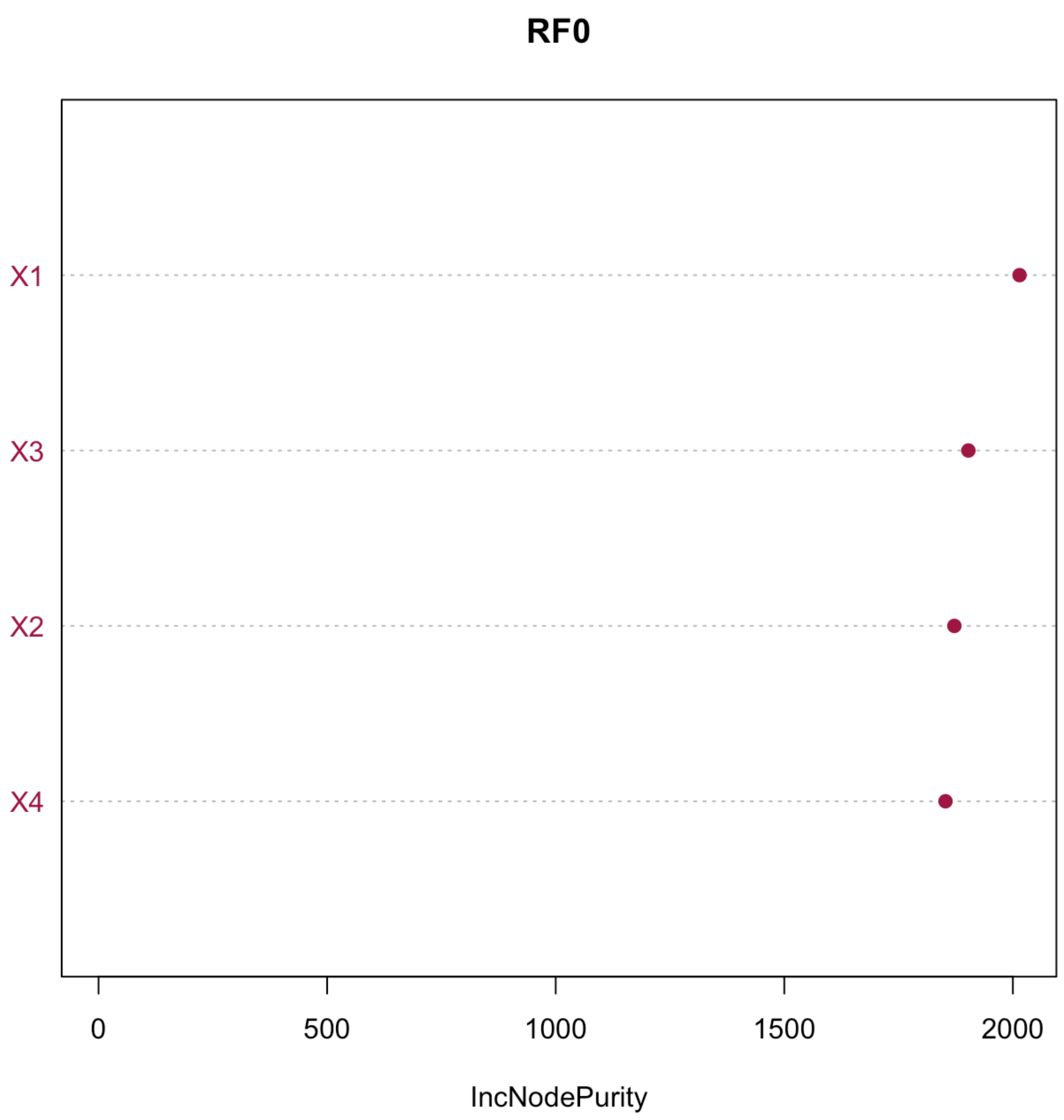
```
plot(RF0)
```



```
100*importance(RF0)/sum(importance(RF0))
```

```
##    IncNodePurity
## X1      26.36283
## X2      24.49661
## X3      24.89763
## X4      24.24292
```

```
varImpPlot(RF0,pch = 19, color="#A20045")
```



```
# accuracy in the test set
mean((predict(RF0,data.test) - data.test$Yt)^2)
```

```
## [1] 207.0088
```

Conditional Inference Forests with

package party

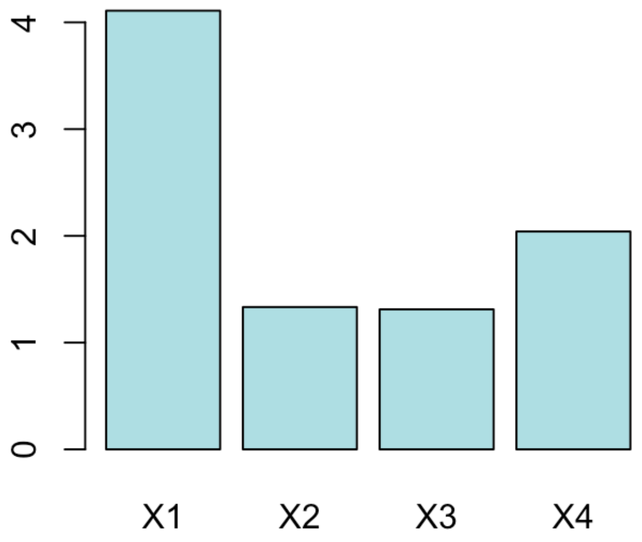
```
library(party)
set.seed(123)
RF1 <- cforest(Y~X1+X2+X3+X4, data=data.train, controls=cforest_unbiased(ntree=500
, mtry=3)) # we are kind
party::varimp(RF1)
```

```
##        X1        X2        X3        X4
## 33.899033  3.070796  2.167062  7.505852
```

```
party::varimp(RF1, conditional = TRUE)
```

```
##       X1       X2       X3       X4
## 4.110277 1.364559 1.320161 2.015668
```

```
barplot(party::varimp(RF1, conditional = TRUE), col = "#b2dfe3")
```



```
# accuracy
mean((predict(RF1,data.test, OOB=TRUE) - data.test$Yt)^2)
```

```
## [1] 205.1967
```

BART

I prefer the package bartMachine, quite fast as written in Java

Someway treaky to install, but afterthat works very well

```
options(java.parameters="-Xmx5000m")
library(bartMachine)
set_bart_machine_num_cores(2)
```

```
## bartMachine now using 2 cores.
```

```
YY <- data.train[,1]
XX <- data.train[,-1]


bart1 <- bartMachine(XX, YY, seed=123)
```

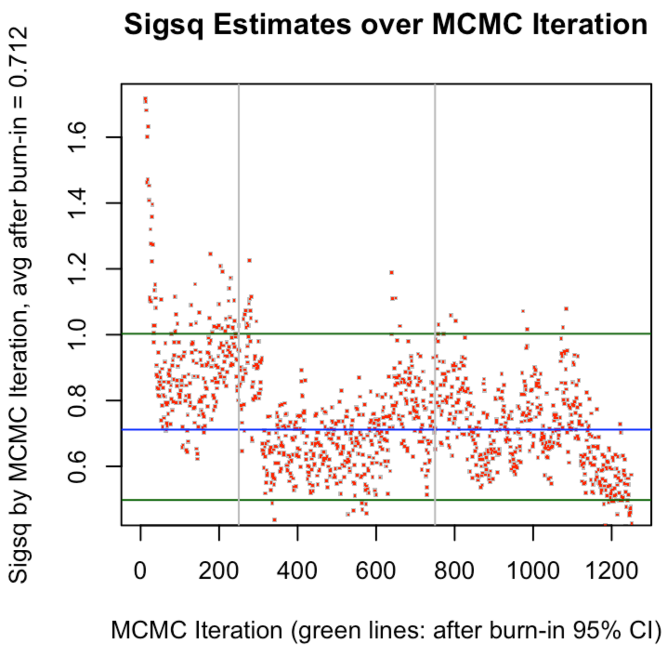
```
## bartMachine initializing with 50 trees...
## bartMachine vars checked...
## bartMachine java init...
## bartMachine factors created...
## bartMachine before preprocess...
## bartMachine after preprocess... 5 total features...
## bartMachine sigsq estimated...
## bartMachine training data finalized...
## Now building bartMachine for regression ...
## evaluating in sample data...done
```

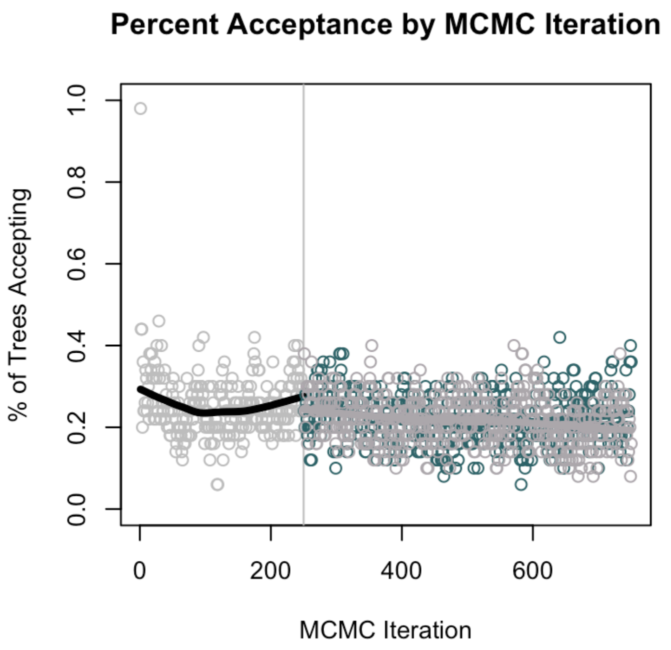
```
bart1
```

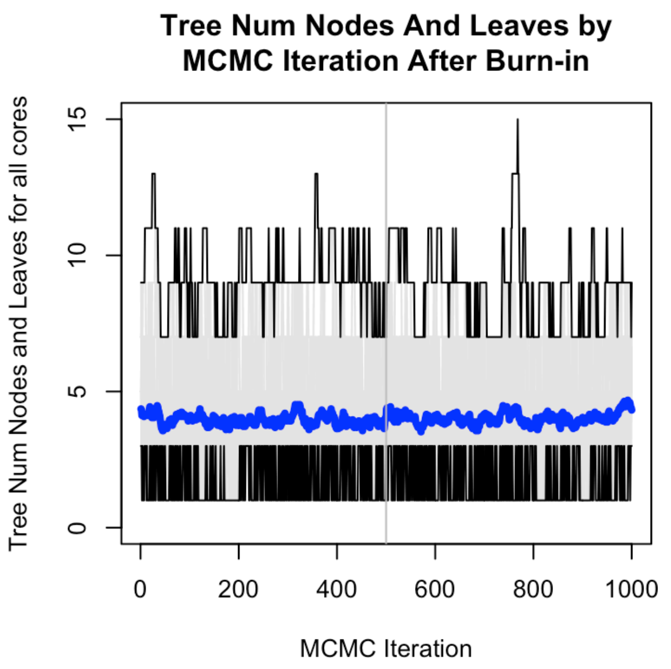
```
## bartMachine v1.2.3 for regression
##
## training data n = 200 and p = 4
## built in 1.8 secs on 2 cores, 50 trees, 250 burn-in and 1000 post. samples
##
## sigsq est for y beforehand: 1.041
## avg sigsq estimate after burn-in: 0.7116
##
## in-sample statistics:
##  L1 = 107.02
##  L2 = 95.6
##  rmse = 0.69
##  Pseudo-Rsq = 0.9877
## p-val for shapiro-wilk test of normality of residuals: 0.00361
## p-val for zero-mean noise: 0.05291
```

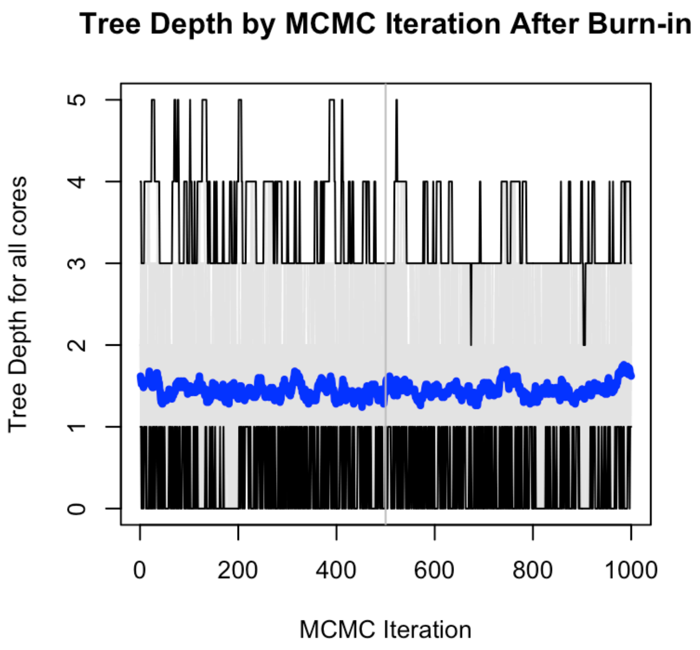
Check for convergence

```
plot_convergence_diagnostics(bart1)
```







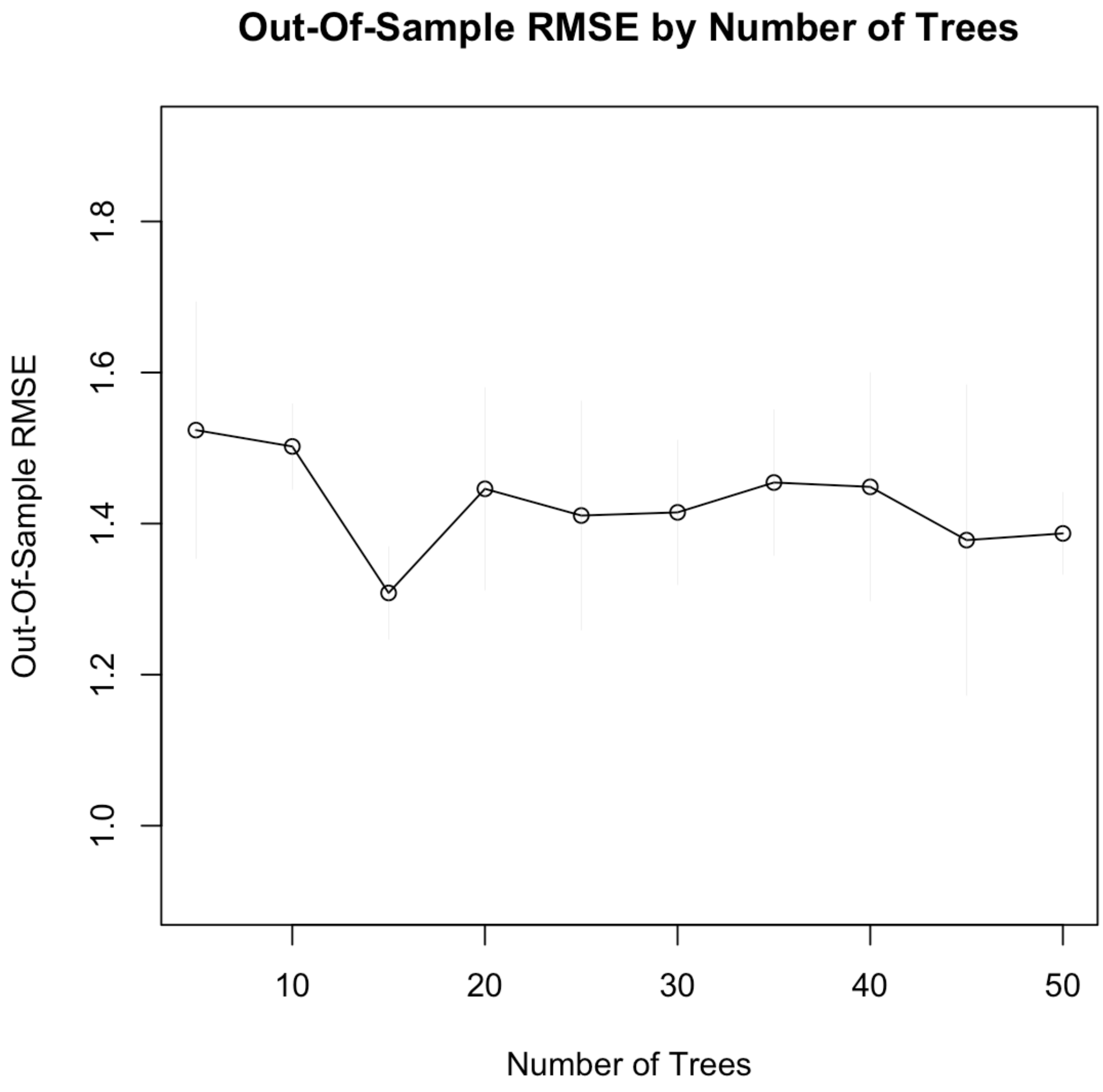


Check for assumptions

```
check_bart_error_assumptions(bart1)
```

```
# how many trees
rmse_by_num_trees(bart1,  tree_list=c(seq(5, 50, by=5)), num_replicates=5)
```

```
## num_trees = ..5..5..5..5..5..10..10..10..10..10..15..15..15..15..15..20..20..20
..20..20..25..25..25..25..25..30..30..30..30..30..35..35..35..35..35..40..40..40..
40..40..45..45..45..45..45..50..50..50..50..50
```



Let's run BART with 20 trees

```
bart1 <- bartMachine(XX, YY, num_trees=20, seed=123)
```

```
## bartMachine initializing with 20 trees...
## bartMachine vars checked...
## bartMachine java init...
## bartMachine factors created...
## bartMachine before preprocess...
## bartMachine after preprocess... 5 total features...
## bartMachine sigsq estimated...
## bartMachine training data finalized...
## Now building bartMachine for regression ...
## evaluating in sample data...done
```

```
bart1
```

```
## bartMachine v1.2.3 for regression
##
## training data n = 200 and p = 4
## built in 0.4 secs on 2 cores, 20 trees, 250 burn-in and 1000 post. samples
##
## sigsq est for y beforehand: 1.041
## avg sigsq estimate after burn-in: 0.98346
##
## in-sample statistics:
##  L1 = 130.85
##  L2 = 144.7
##  rmse = 0.85
##  Pseudo-Rsq = 0.9814
## p-val for shapiro-wilk test of normality of residuals: 0.50648
## p-val for zero-mean noise: 0.13787
```

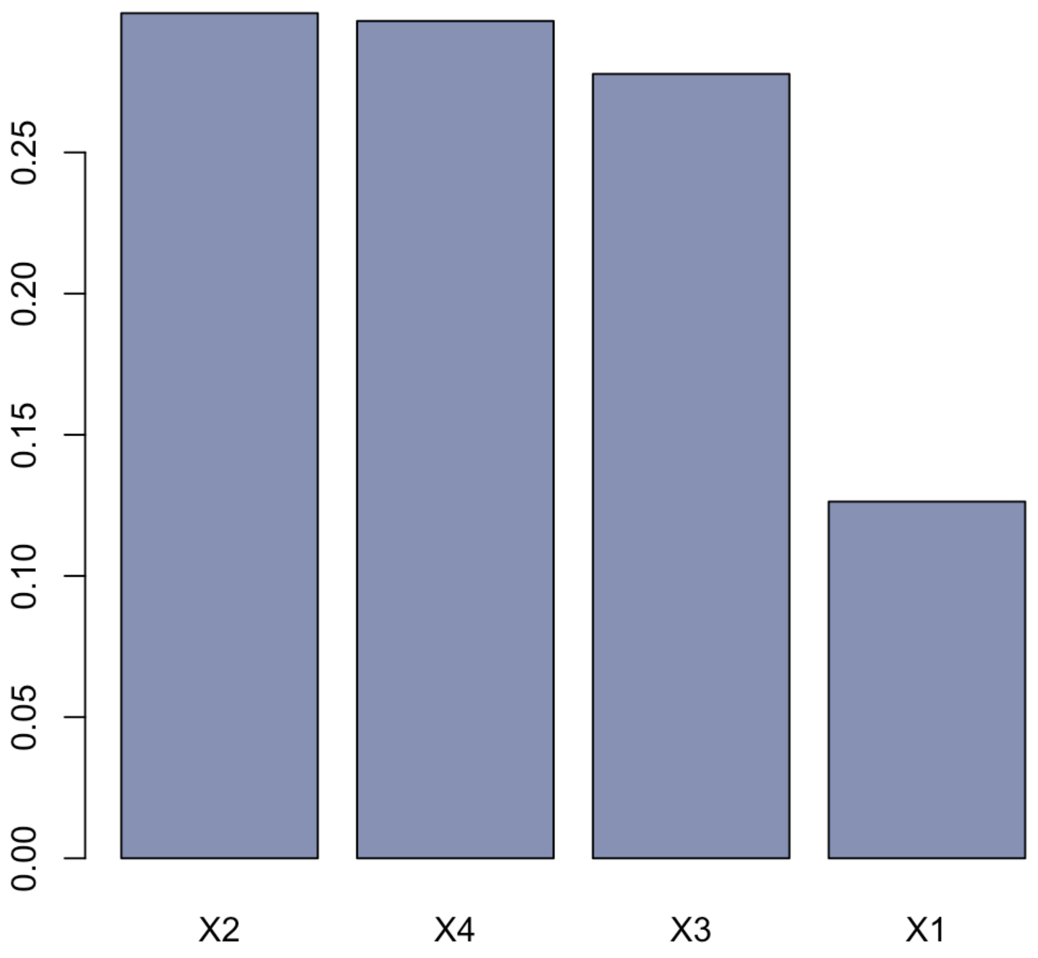
```
# variable importance
pVI1<-var_selection_by_permute(bart1, num_reps_for_avg=20, plot=FALSE, num_permute
_samples=20)
```

```
## avg....................null....................
```

```
pVI1$var_true_props_avg
```

```
##        X2        X4        X3        X1
## 0.2993218 0.2965521 0.2777908 0.1263353
```

```
barplot(pVI1$var_true_props_avg, col = "#8A92B8")
```



Prediction

```
XXt <- data.frame(data.test[,-1])
names(XXt) <- names(XX)
YYt <- data.frame(data.test[,1])
pred<- bart_predict_for_test_data(bart1, Xtest = XXt, YYt)
mean((pred$y_hat - data.test$Yt)^2)
```

```
## [1] 40.19839
```

```
colMeans((pred$e)^2)
```

```
## data.test...1.
##       40.19839
```

Data from Michela Baccini (miR) - ask her

```
datMB <- read.table("Dati_Anna.txt", header = TRUE)
dim(datMB)
```

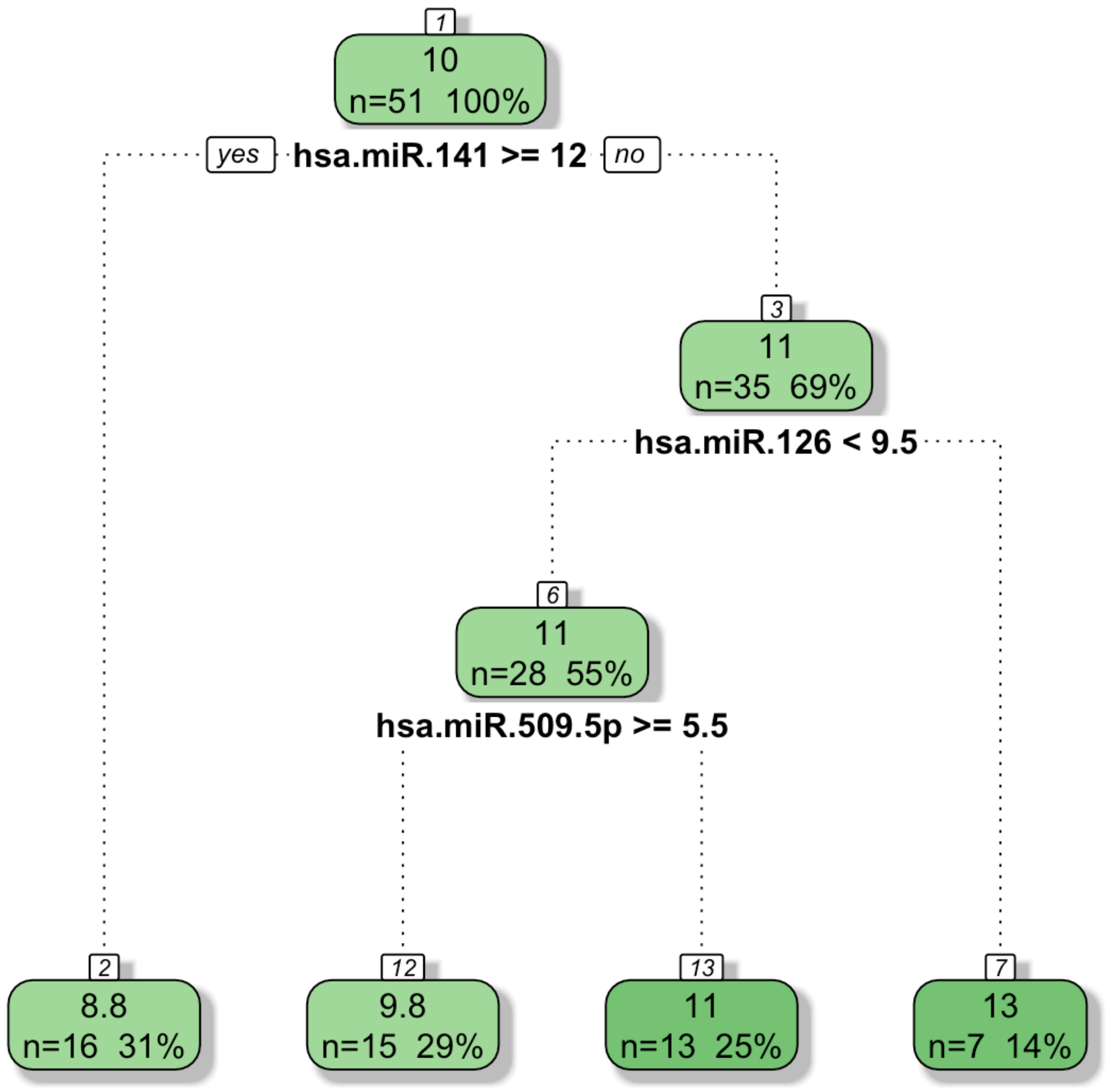
```
## [1] 76 25
```

```
set.seed(123)
test <- sample(1:dim(datMB)[1], 25)
```

Let’s start from the simple rpart and see:

```
library(rpart)
MB.T0 <- rpart(X4629~., data=datMB[-test,])

rattle::fancyRpartPlot(MB.T0)
```



Rattle 2019-May-28 23:06:37 annagottard

```
MB.T0$cptable
```

```
##           CP nsplit rel error   xerror      xstd
## 1 0.25106440      0 1.0000000 1.024336 0.1883597
## 2 0.16370621      1 0.7489356 1.082159 0.1836525
## 3 0.09258786      2 0.5852294 1.231326 0.2294923
## 4 0.01000000      3 0.4926415 1.099139 0.2074706
```

```
MB.T0$variable.importance
```

```
##     hsa.miR.141     hsa.miR.126     hsa.miR.424      hsa.miR.25 
##       66.035874       33.654010       22.518613       19.354772 
##  hsa.miR.509.5p    hsa.miR.376c     hsa.miR.92a     hsa.miR.506 
##       19.033809       16.936127       16.128977       10.248974 
##     hsa.miR.10a     hsa.miR.144    hsa.miR.1260     hsa.miR.155 
##        9.677386        9.677386        9.615432        8.784835 
##    hsa.miR.181d hsa.miR.193a.5p     hsa.miR.203 
##        7.320696        7.320696        4.807716
```

** Prediction**

```
MB.T0.pred<- predict(MB.T0, newdata=datMB[test,])

# # accuracy in the test set
mean((datMB[test,1] - MB.T0.pred)^2)
```

```
## [1] 4.22041
```

** Random Forests **

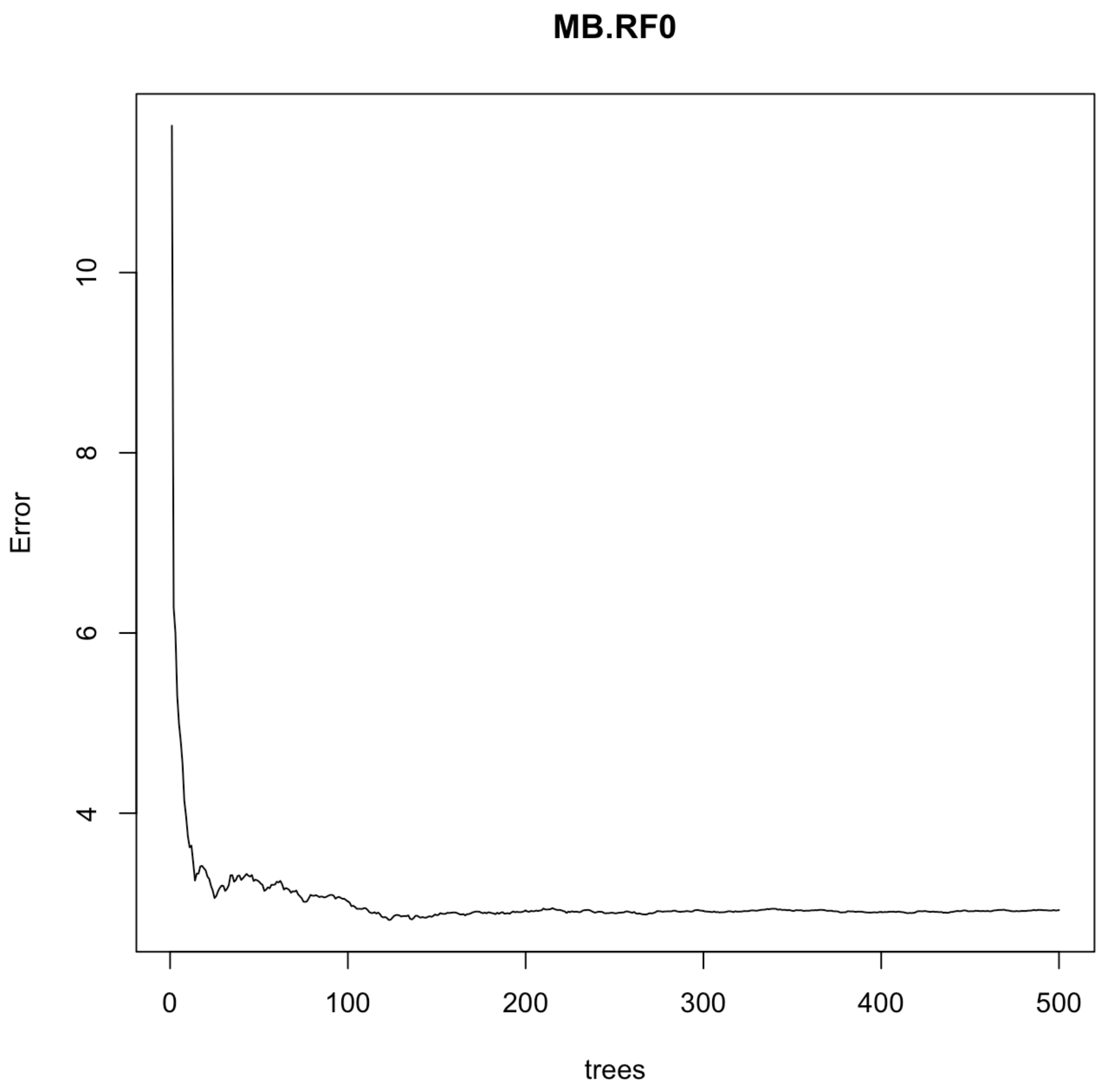
```
library(randomForest)
set.seed(123)
MB.RF0 <- randomForest(X4629~., data=datMB[-test,])
print(MB.RF0)
```

```
## 
## Call:
##  randomForest(formula = X4629 ~ ., data = datMB[-test, ]) 
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 8
## 
##           Mean of squared residuals: 2.925534
##                     % Var explained: 27.42
```

```
importance(MB.RF0)
```

```
##                   IncNodePurity
## hsa.miR.141           19.976129
## hsa.miR.203            6.354123
## hsa.miR.18a            7.590843
## hsa.miR.506            7.502267
## hsa.miR.23a            5.966101
## hsa.miR.1260          11.234315
## hsa.miR.381            6.881142
## hsa.miR.509.5p         9.496275
## hsa.miR.424           12.054569
## hsa.miR.27a           11.449221
## hsa.miR.25             3.485821
## hsa.miR.10a            6.656576
## hsa.miR.23b           10.122205
## hsa.miR.144            5.085693
## hsa.miR.155            4.790652
## hsa.miR.148a           4.653961
## hsa.miR.193a.5p        3.726950
## hsa.miR.223            6.855876
## hsa.miR.376c          14.276534
## hsa.miR.92a            8.709091
## hsa.miR.29a            2.348920
## hsa.miR.126           16.112391
## hsa.miR.1246           3.463398
## hsa.miR.181d           4.900801
```

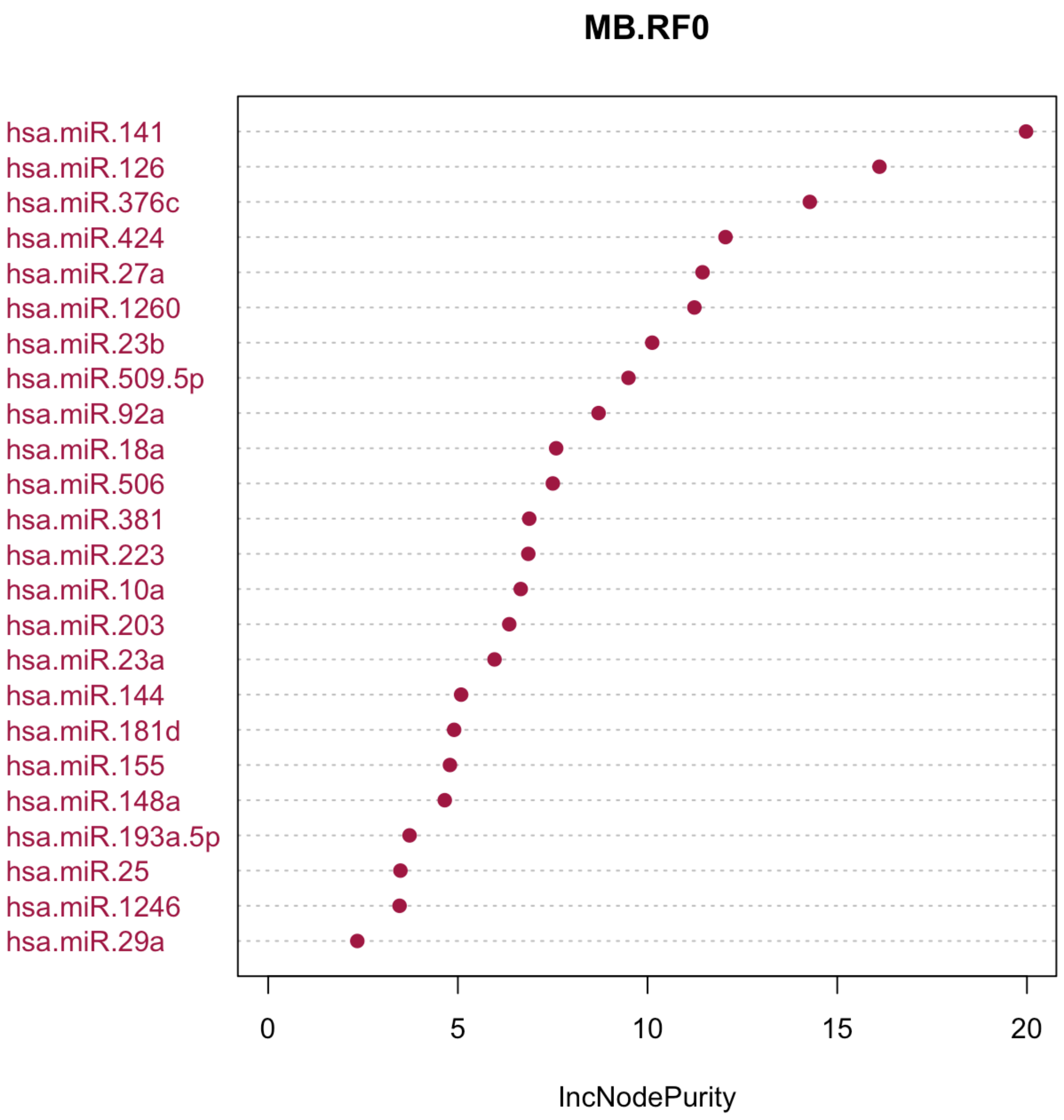
```
plot(MB.RF0)
```



```
100*importance(MB.RF0)/sum(importance(MB.RF0))
```

```
##                  IncNodePurity
## hsa.miR.141         10.313249
## hsa.miR.203          3.280498
## hsa.miR.18a          3.918990
## hsa.miR.506          3.873260
## hsa.miR.23a          3.080171
## hsa.miR.1260         5.800037
## hsa.miR.381          3.552587
## hsa.miR.509.5p       4.902724
## hsa.miR.424          6.223517
## hsa.miR.27a          5.910988
## hsa.miR.25           1.799655
## hsa.miR.10a          3.436648
## hsa.miR.23b          5.225879
## hsa.miR.144          2.625635
## hsa.miR.155          2.473311
## hsa.miR.148a         2.402741
## hsa.miR.193a.5p      1.924144
## hsa.miR.223          3.539542
## hsa.miR.376c         7.370669
## hsa.miR.92a          4.496318
## hsa.miR.29a          1.212697
## hsa.miR.126          8.318484
## hsa.miR.1246         1.788078
## hsa.miR.181d         2.530179
```

```
varImpPlot(MB.RF0,pch = 19, color="#A20045")
```



```
# accuracy in the test set
mean((predict(MB.RF0,datMB[test,]) - datMB[test,1])^2)
```

```
## [1] 3.056108
```

Binary response: a classification problem

Diabetes in Pima Indian Women

A population of women who were at least 21 years old, of Pima Indian heritage and living near Phoenix, Arizona, was tested for diabetes according to World Health Organization criteria.

The data were collected by the US National Institute of Diabetes and Digestive and Kidney Diseases.

Data set 532 complete records devided in:

Pima.tr : 200 randomly selected subjects Pima.te : the remaining 332 subjects.

```
dat.train <- MASS::Pima.tr # training data
dat.test <- MASS::Pima.te # training data
```

#These data frames contains the following columns:

npreg number of pregnancies.

glu plasma glucose concentration in an oral glucose tolerance test.

bp diastolic blood pressure (mm Hg).

skin triceps skin fold thickness (mm).

bmi body mass index (weight in kg/(height in m)^2).

ped diabetes pedigree function.

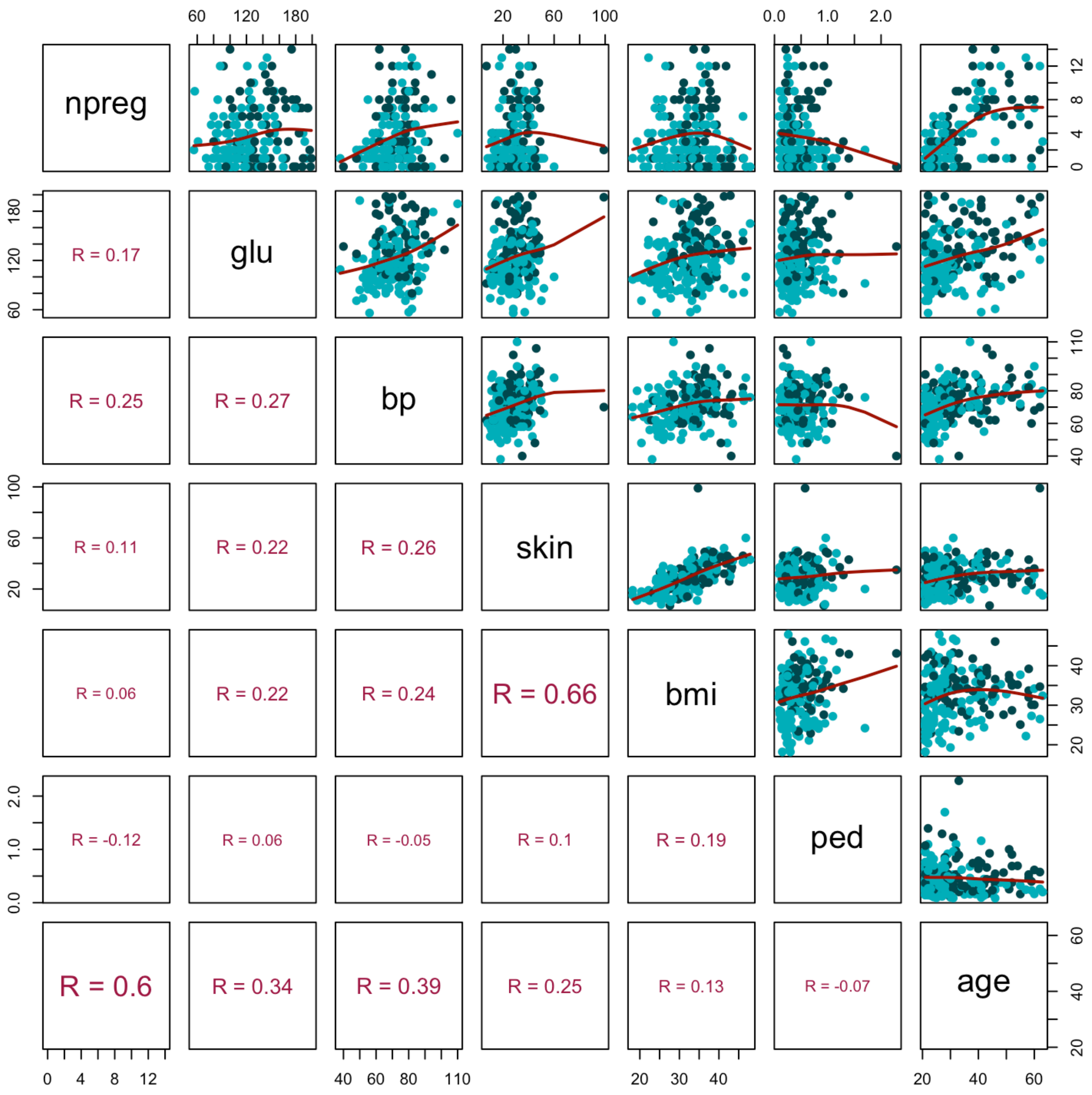
age age in years.

type Yes or No, for diabetic according to WHO criteria.

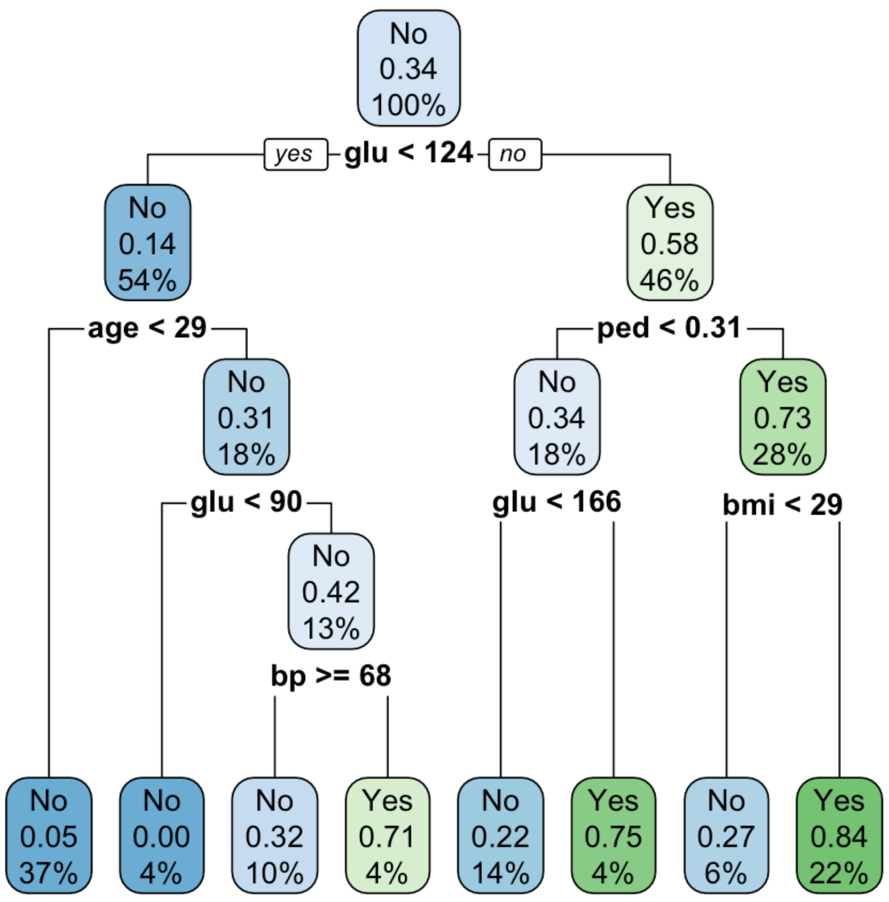
CART for classificarion

Let us start briefly on how to **grow a tree** with the CART algorithm

Several packages run the CART algorithm, **rpart** is the most used



```
library(rpart)
T2 <- rpart(type~.,dat.train)
rpart.plot::rpart.plot(T2, type = 2)
```

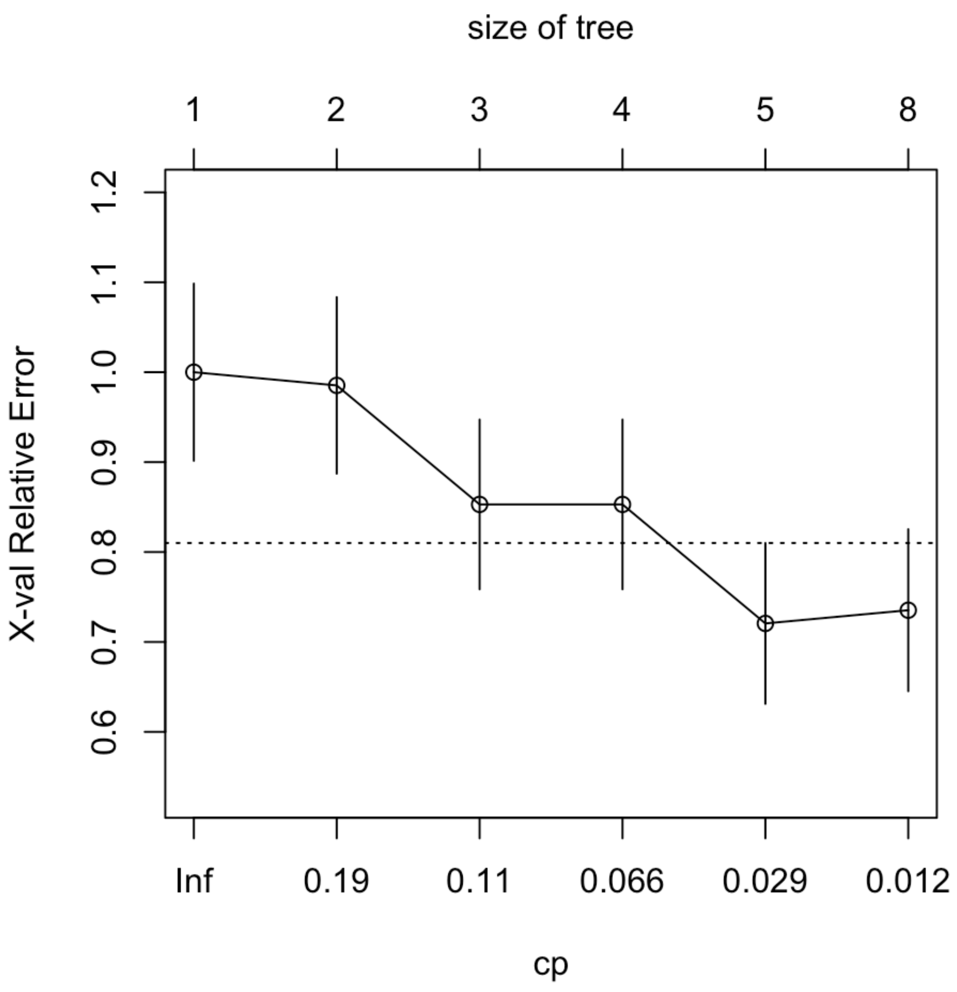


```
summary(T2)
```

```
## Call:
## rpart(formula = type ~ ., data = dat.train)
##   n= 200
##
##           CP nsplit rel error    xerror       xstd
## 1 0.22058824      0 1.0000000 1.0000000 0.09851844
## 2 0.16176471      1 0.7794118 0.9852941 0.09816108
## 3 0.07352941      2 0.6176471 0.8529412 0.09437007
## 4 0.05882353      3 0.5441176 0.8529412 0.09437007
## 5 0.01470588      4 0.4852941 0.7205882 0.08944635
## 6 0.01000000      7 0.4411765 0.7352941 0.09005477
##
## Variable importance
##   glu   age   bmi    bp   ped npreg  skin
##    38    13    11    11    11     8     8
##
## Node number 1: 200 observations,    complexity param=0.2205882
##   predicted class=No    expected loss=0.34  P(node) =1
##     class counts:   132    68
##    probabilities: 0.660 0.340
##   left son=2 (109 obs) right son=3 (91 obs)
##   Primary splits:
##       glu   < 123.5  to the left,  improve=19.624700, (0 missing)
##       age   < 28.5   to the left,  improve=15.016410, (0 missing)
##       npreg < 6.5    to the left,  improve=10.465630, (0 missing)
##       bmi   < 27.35  to the left,  improve= 9.727105, (0 missing)
##       skin  < 22.5   to the left,  improve= 8.201159, (0 missing)
##   Surrogate splits:
##       age   < 30.5   to the left,  agree=0.685, adj=0.308, (0 split)
##       bp    < 77     to the left,  agree=0.650, adj=0.231, (0 split)
##       npreg < 6.5    to the left,  agree=0.640, adj=0.209, (0 split)
##       skin  < 32.5   to the left,  agree=0.635, adj=0.198, (0 split)
##       bmi   < 30.85  to the left,  agree=0.575, adj=0.066, (0 split)
##
## Node number 2: 109 observations,    complexity param=0.01470588
##   predicted class=No   expected loss=0.1376147  P(node) =0.545
##     class counts:    94    15
##    probabilities: 0.862 0.138
##   left son=4 (74 obs) right son=5 (35 obs)
##   Primary splits:
##       age   < 28.5   to the left,  improve=3.2182780, (0 missing)
##       npreg < 6.5    to the left,  improve=2.4578310, (0 missing)
##       bmi   < 33.5   to the left,  improve=1.6403660, (0 missing)
##       bp    < 59     to the left,  improve=0.9851960, (0 missing)
##       skin  < 24     to the left,  improve=0.8342926, (0 missing)
##   Surrogate splits:
##       npreg < 4.5    to the left,  agree=0.798, adj=0.371, (0 split)
##       bp    < 77     to the left,  agree=0.734, adj=0.171, (0 split)
##       skin  < 36.5   to the left,  agree=0.725, adj=0.143, (0 split)
##       bmi   < 38.85  to the left,  agree=0.716, adj=0.114, (0 split)
##       glu   < 66     to the right, agree=0.688, adj=0.029, (0 split)
##
## Node number 3: 91 observations,    complexity param=0.1617647
##   predicted class=Yes  expected loss=0.4175824  P(node) =0.455
##     class counts:    38    53
##    probabilities: 0.418 0.582
##   left son=6 (35 obs) right son=7 (56 obs)
##   Primary splits:
##       ped  < 0.3095 to the left,  improve=6.528022, (0 missing)
##       bmi  < 28.65  to the left,  improve=6.473260, (0 missing)
##       skin < 19.5   to the left,  improve=4.778504, (0 missing)
##       glu  < 166    to the left,  improve=4.104532, (0 missing)
##       age  < 39.5   to the left,  improve=3.607390, (0 missing)
##   Surrogate splits:
##       glu   < 126.5  to the left,  agree=0.670, adj=0.143, (0 split)
##       bp    < 93     to the right, agree=0.659, adj=0.114, (0 split)
##       bmi   < 27.45  to the left,  agree=0.659, adj=0.114, (0 split)
##       npreg < 9.5    to the right, agree=0.648, adj=0.086, (0 split)
##       skin  < 20.5   to the left,  agree=0.637, adj=0.057, (0 split)
##
```

```
## Node number 4: 74 observations
##   predicted class=No   expected loss=0.05405405  P(node) =0.37
##     class counts:    70     4
##    probabilities: 0.946 0.054 
## 
## Node number 5: 35 observations,    complexity param=0.01470588
##   predicted class=No   expected loss=0.3142857  P(node) =0.175
##     class counts:    24    11
##    probabilities: 0.686 0.314 
##   left son=10 (9 obs) right son=11 (26 obs)
##   Primary splits:
##       glu  < 90     to the left,  improve=2.3934070, (0 missing)
##       bmi  < 33.4   to the left,  improve=1.3714290, (0 missing)
##       bp   < 68     to the right, improve=0.9657143, (0 missing)
##       ped  < 0.334  to the left,  improve=0.9475564, (0 missing)
##       skin < 39.5   to the right, improve=0.7958592, (0 missing)
##   Surrogate splits:
##       ped < 0.1795 to the left,  agree=0.8, adj=0.222, (0 split)
## 
## Node number 6: 35 observations,    complexity param=0.05882353
##   predicted class=No   expected loss=0.3428571  P(node) =0.175
##     class counts:    23    12
##    probabilities: 0.657 0.343 
##   left son=12 (27 obs) right son=13 (8 obs)
##   Primary splits:
##       glu   < 166    to the left,  improve=3.438095, (0 missing)
##       ped   < 0.2545 to the right, improve=1.651429, (0 missing)
##       skin  < 25.5   to the left,  improve=1.651429, (0 missing)
##       npreg < 3.5    to the left,  improve=1.078618, (0 missing)
##       bp    < 73     to the right, improve=1.078618, (0 missing)
##   Surrogate splits:
##       bp < 94.5   to the left,  agree=0.8, adj=0.125, (0 split)
## 
## Node number 7: 56 observations,    complexity param=0.07352941
##   predicted class=Yes  expected loss=0.2678571  P(node) =0.28
##     class counts:    15    41
##    probabilities: 0.268 0.732 
##   left son=14 (11 obs) right son=15 (45 obs)
##   Primary splits:
##       bmi   < 28.65  to the left,  improve=5.778427, (0 missing)
##       age   < 39.5   to the left,  improve=3.259524, (0 missing)
##       npreg < 6.5    to the left,  improve=2.133215, (0 missing)
##       ped   < 0.8295 to the left,  improve=1.746894, (0 missing)
##       skin  < 22     to the left,  improve=1.474490, (0 missing)
##   Surrogate splits:
##       skin < 19.5   to the left,  agree=0.839, adj=0.182, (0 split)
## 
## Node number 10: 9 observations
##   predicted class=No   expected loss=0  P(node) =0.045
##     class counts:     9     0
##    probabilities: 1.000 0.000
##
## Node number 11: 26 observations,    complexity param=0.01470588
##   predicted class=No   expected loss=0.4230769  P(node) =0.13
##     class counts:    15    11
##    probabilities: 0.577 0.423
##   left son=22 (19 obs) right son=23 (7 obs)
##   Primary splits:
##       bp    < 68     to the right, improve=1.6246390, (0 missing)
##       bmi   < 33.4   to the left,  improve=1.6173080, (0 missing)
##       npreg < 6.5    to the left,  improve=0.9423077, (0 missing)
##       skin  < 39.5   to the right, improve=0.6923077, (0 missing)
##       ped   < 0.334  to the left,  improve=0.4923077, (0 missing)
##   Surrogate splits:
##       glu < 94.5   to the right, agree=0.808, adj=0.286, (0 split)
##       ped < 0.2105 to the right, agree=0.808, adj=0.286, (0 split)
##
## Node number 12: 27 observations
##   predicted class=No   expected loss=0.2222222  P(node) =0.135
##     class counts:    21     6
##    probabilities: 0.778 0.222
##
## Node number 13: 8 observations
##   predicted class=Yes  expected loss=0.25  P(node) =0.04
##     class counts:     2     6
##    probabilities: 0.250 0.750
##
## Node number 14: 11 observations
##   predicted class=No   expected loss=0.2727273  P(node) =0.055
##     class counts:     8     3
##    probabilities: 0.727 0.273
##
## Node number 15: 45 observations
##   predicted class=Yes  expected loss=0.1555556  P(node) =0.225
##     class counts:     7    38
##    probabilities: 0.156 0.844
##
## Node number 22: 19 observations
##   predicted class=No   expected loss=0.3157895  P(node) =0.095
##     class counts:    13     6
##    probabilities: 0.684 0.316
##
## Node number 23: 7 observations
##   predicted class=Yes  expected loss=0.2857143  P(node) =0.035
##     class counts:     2     5
##    probabilities: 0.286 0.714
```

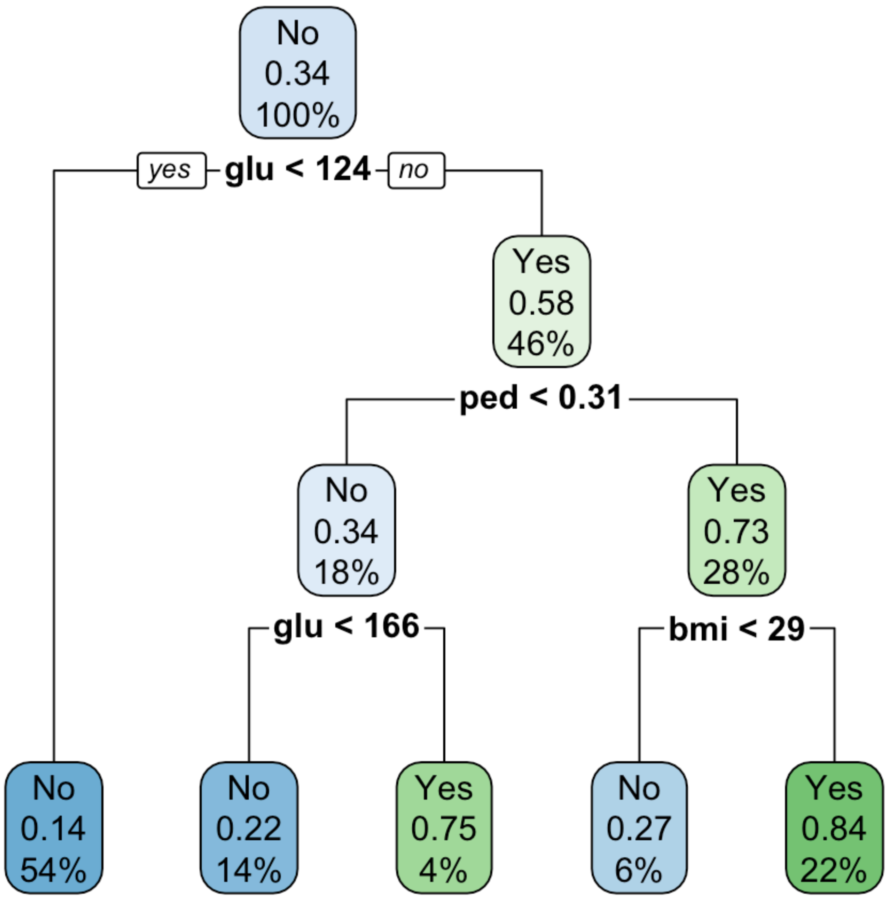
```
plotcp(T2)
```



```
printcp(T2)
```

```
##
## Classification tree:
## rpart(formula = type ~ ., data = dat.train)
##
## Variables actually used in tree construction:
## [1] age bmi bp  glu ped
##
## Root node error: 68/200 = 0.34
##
## n= 200
##
##         CP nsplit rel error  xerror     xstd
## 1 0.220588      0   1.00000 1.00000 0.098518
## 2 0.161765      1   0.77941 0.98529 0.098161
## 3 0.073529      2   0.61765 0.85294 0.094370
## 4 0.058824      3   0.54412 0.85294 0.094370
## 5 0.014706      4   0.48529 0.72059 0.089446
## 6 0.010000      7   0.44118 0.73529 0.090055
```

```
T3 <- prune.rpart(T2,.029)
rpart.plot::rpart.plot(T3, type = 2)
```



```
accuracyT2 <- mean(predict(T2, dat.test, type = "class") ==dat.test$type )
accuracyT3 <- mean(predict(T3, dat.test, type = "class") ==dat.test$type )
c(accuracyT2, accuracyT3)
```

```
## [1] 0.7319277 0.7560241
```