CART practice

Mathias Bourel

Install and load rpart package. Another package to make tree is the tree package.

Install partykit package (this package makes better picture and have a more suitable interfase)

Look at rpart.control to see the different parameters:

```
rm(list=ls())
library(rpart)
library(partykit)
```

```
## Loading required package: grid

## Loading required package: libcoin

## Loading required package: mvtnorm
```

```
?rpart
?rpart.control
#rpart.control(minsplit = 20, minbucket = round(minsplit/3), cp = 0.01,
#               maxcompete = 4, maxsurrogate = 5, usesurrogate = 2, xval = 10,
#               surrogatestyle = 0, maxdepth = 30)
```

CLASSIFICATION TREES

We use Iris data

```
attach(iris)
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

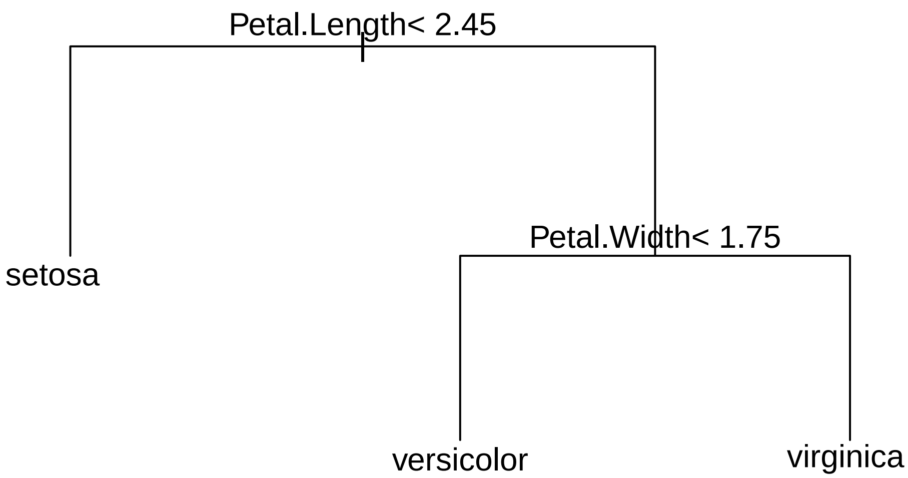
CART

```
model=rpart(Species~.,data=iris)
model
```

```
## n= 150
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
##   2) Petal.Length< 2.45 50   0 setosa (1.00000000 0.00000000 0.00000000) *
##   3) Petal.Length>=2.45 100  50 versicolor (0.00000000 0.50000000 0.50000000)
##     6) Petal.Width< 1.75 54   5 versicolor (0.00000000 0.90740741 0.09259259) *
##     7) Petal.Width>=1.75 46   1 virginica (0.00000000 0.02173913 0.97826087) *
```

Take the time to try to understand what is displayed.

```r
plot(model,margin=0.1)
text(model)
```




```r
model.cl=rpart(Species~.,cp=0.001,minsplit=5,iris)
summary(model.cl)
```

```
## Call:
## rpart(formula = Species ~ ., data = iris, cp = 0.001, minsplit = 5)
##   n= 150
##
##      CP nsplit rel error xerror       xstd
## 1 0.500      0      1.00   1.18 0.05017303
## 2 0.440      1      0.50   0.62 0.06031031
## 3 0.020      2      0.06   0.10 0.03055050
## 4 0.010      3      0.04   0.10 0.03055050
## 5 0.001      4      0.03   0.08 0.02751969
##
## Variable importance
##  Petal.Width Petal.Length Sepal.Length  Sepal.Width
##           34           32           21           14
##
## Node number 1: 150 observations,    complexity param=0.5
##   predicted class=setosa      expected loss=0.6666667  P(node) =1
##     class counts:    50    50    50
##    probabilities: 0.333 0.333 0.333
##   left son=2 (50 obs) right son=3 (100 obs)
##   Primary splits:
```

```
##       Petal.Length < 2.45 to the left,  improve=50.00000, (0 missing)
##       Petal.Width  < 0.8  to the left,  improve=50.00000, (0 missing)
##       Sepal.Length < 5.45 to the left,  improve=34.16405, (0 missing)
##       Sepal.Width  < 3.35 to the right, improve=19.03851, (0 missing)
##   Surrogate splits:
##       Petal.Width  < 0.8  to the left,  agree=1.000, adj=1.00, (0 split)
##       Sepal.Length < 5.45 to the left,  agree=0.920, adj=0.76, (0 split)
##       Sepal.Width  < 3.35 to the right, agree=0.833, adj=0.50, (0 split)
##
## Node number 2: 50 observations
##   predicted class=setosa      expected loss=0  P(node) =0.3333333
##     class counts:    50     0     0
##    probabilities: 1.000 0.000 0.000
##
## Node number 3: 100 observations,    complexity param=0.44
##   predicted class=versicolor  expected loss=0.5  P(node) =0.6666667
##     class counts:     0    50    50
##    probabilities: 0.000 0.500 0.500
##   left son=6 (54 obs) right son=7 (46 obs)
##   Primary splits:
##       Petal.Width  < 1.75 to the left,  improve=38.969400, (0 missing)
##       Petal.Length < 4.75 to the left,  improve=37.353540, (0 missing)
##       Sepal.Length < 6.15 to the left,  improve=10.686870, (0 missing)
##       Sepal.Width  < 2.45 to the left,  improve= 3.555556, (0 missing)
##   Surrogate splits:
##       Petal.Length < 4.75 to the left,  agree=0.91, adj=0.804, (0 split)
##       Sepal.Length < 6.15 to the left,  agree=0.73, adj=0.413, (0 split)
##       Sepal.Width  < 2.95 to the left,  agree=0.67, adj=0.283, (0 split)
##
## Node number 6: 54 observations,    complexity param=0.02
##   predicted class=versicolor  expected loss=0.09259259  P(node) =0.36
##     class counts:     0    49     5
##    probabilities: 0.000 0.907 0.093
##   left son=12 (48 obs) right son=13 (6 obs)
##   Primary splits:
##       Petal.Length < 4.95 to the left,  improve=4.4490740, (0 missing)
##       Petal.Width  < 1.35 to the left,  improve=0.9971510, (0 missing)
##       Sepal.Length < 4.95 to the right, improve=0.6894587, (0 missing)
##       Sepal.Width  < 2.65 to the right, improve=0.2500139, (0 missing)
##
## Node number 7: 46 observations
##   predicted class=virginica   expected loss=0.02173913  P(node) =0.3066667
##     class counts:     0     1    45
##    probabilities: 0.000 0.022 0.978
##
## Node number 12: 48 observations
##   predicted class=versicolor  expected loss=0.02083333  P(node) =0.32
##     class counts:     0    47     1
##    probabilities: 0.000 0.979 0.021
##
## Node number 13: 6 observations,    complexity param=0.01
##   predicted class=virginica   expected loss=0.3333333  P(node) =0.04
##     class counts:     0     2     4
##    probabilities: 0.000 0.333 0.667
```

```
##   left son=26 (3 obs) right son=27 (3 obs)
##   Primary splits:
##       Petal.Width  < 1.55 to the right, improve=1.3333330, (0 missing)
##       Sepal.Width  < 2.65 to the right, improve=0.6666667, (0 missing)
##       Petal.Length < 5.35 to the left,  improve=0.6666667, (0 missing)
##       Sepal.Length < 6.05 to the left,  improve=0.1666667, (0 missing)
##   Surrogate splits:
##       Sepal.Length < 6.5  to the right, agree=0.833, adj=0.667, (0 split)
##       Sepal.Width  < 2.65 to the right, agree=0.833, adj=0.667, (0 split)
##
## Node number 26: 3 observations
##   predicted class=versicolor  expected loss=0.3333333  P(node) =0.02
##     class counts:     0     2     1
##    probabilities: 0.000 0.667 0.333
##
## Node number 27: 3 observations
##   predicted class=virginica   expected loss=0  P(node) =0.02
##     class counts:     0     0     3
##    probabilities: 0.000 0.000 1.000
```

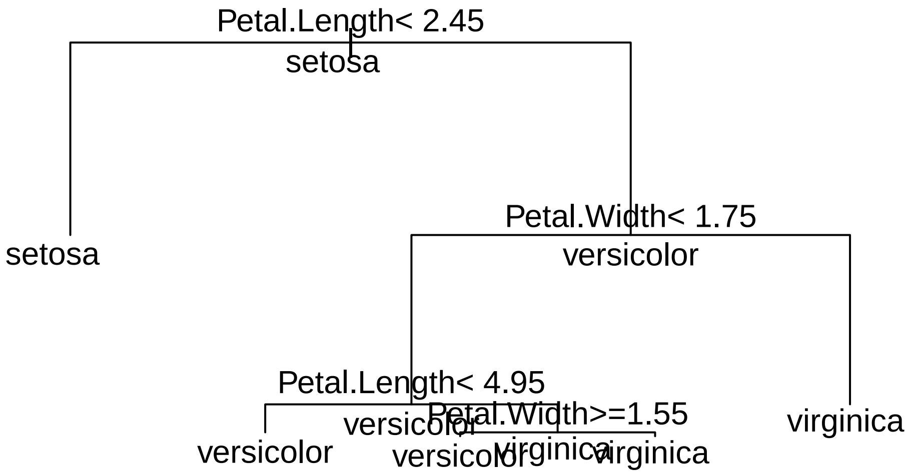
```
model.cl
```

```
## n= 150
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
##    2) Petal.Length< 2.45 50   0 setosa (1.00000000 0.00000000 0.00000000) *
##    3) Petal.Length>=2.45 100  50 versicolor (0.00000000 0.50000000 0.50000000)
##      6) Petal.Width< 1.75 54   5 versicolor (0.00000000 0.90740741 0.09259259)
##       12) Petal.Length< 4.95 48   1 versicolor (0.00000000 0.97916667 0.02083333) *
##       13) Petal.Length>=4.95 6   2 virginica (0.00000000 0.33333333 0.66666667)
##         26) Petal.Width>=1.55 3   1 versicolor (0.00000000 0.66666667 0.33333333) *
##         27) Petal.Width< 1.55 3   0 virginica (0.00000000 0.00000000 1.00000000) *
##      7) Petal.Width>=1.75 46   1 virginica (0.00000000 0.02173913 0.97826087) *
```

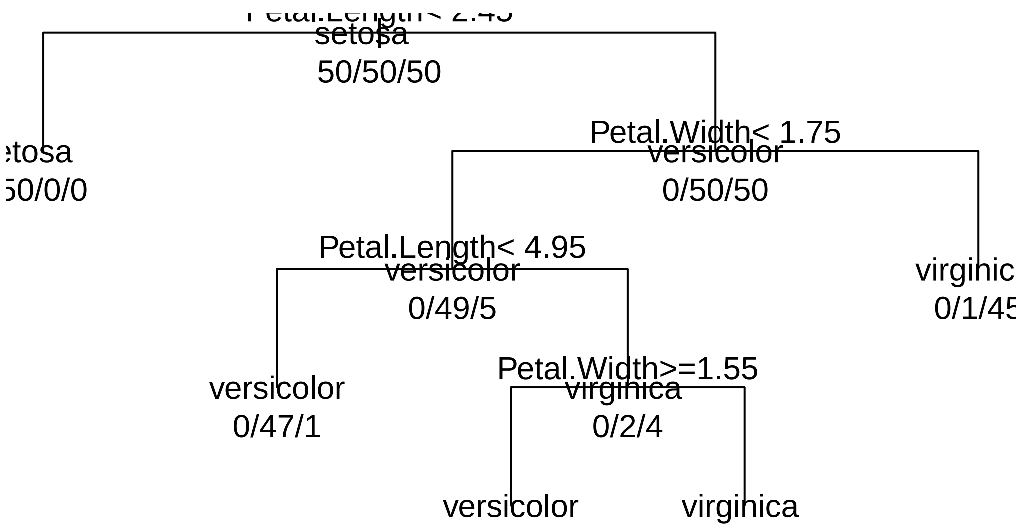
Take the time to try to understand what is displayed.

rpart uses a default cp value of 0.01 if you don't specify one in prune.

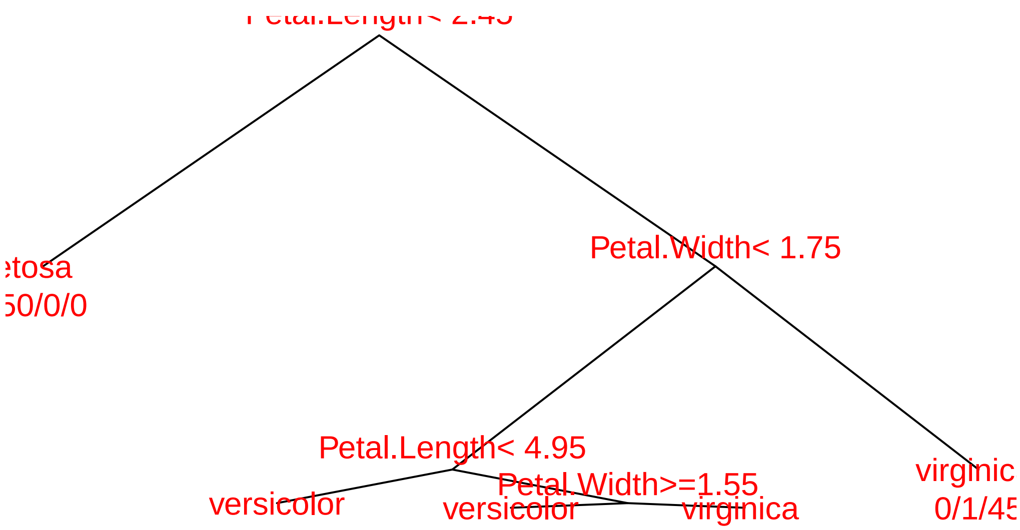
```
# x11()
# par(mfrow=c(3,2))
plot(model.cl,margin=0.1)
text(model.cl,all=T)
```



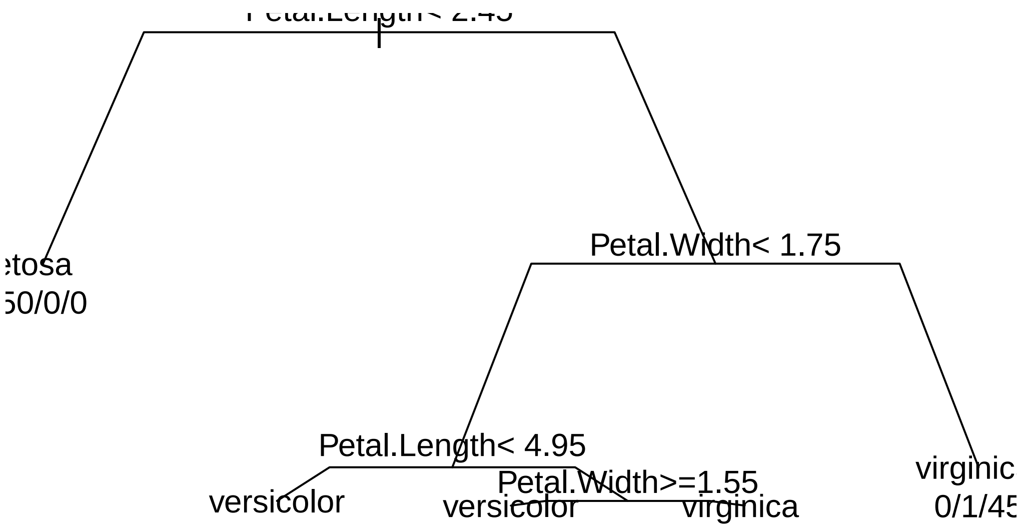

```r
plot(model.cl, uniform=T)
text(model.cl, use.n=T, all=T)
```



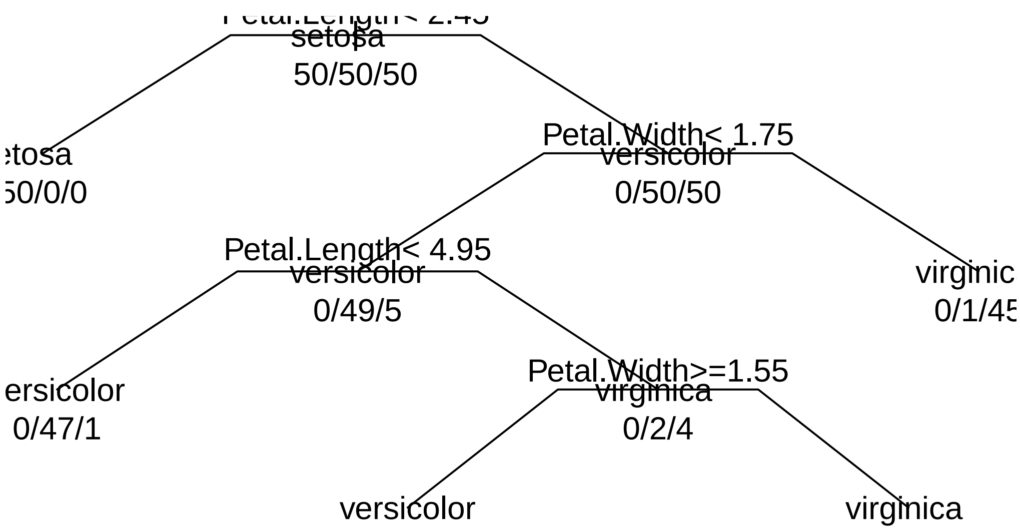

```r
plot(model.cl, branch=0)
text(model.cl, use.n=T,col="red")
```



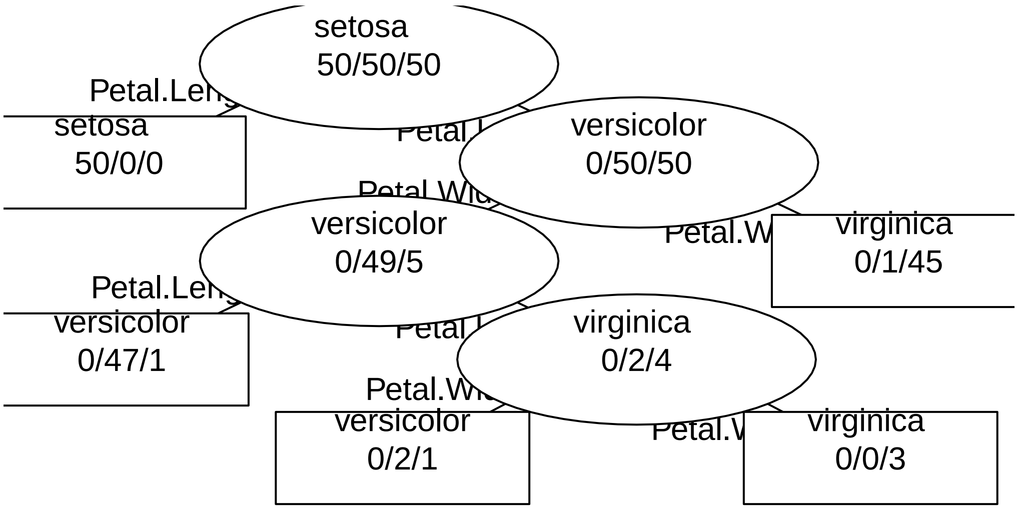
```
plot(model.cl, branch=.7)
text(model.cl, use.n=T)
```




```
plot(model.cl, branch=.4, uniform=T, compress=T)
text(model.cl, all=T,use.n=T)
```



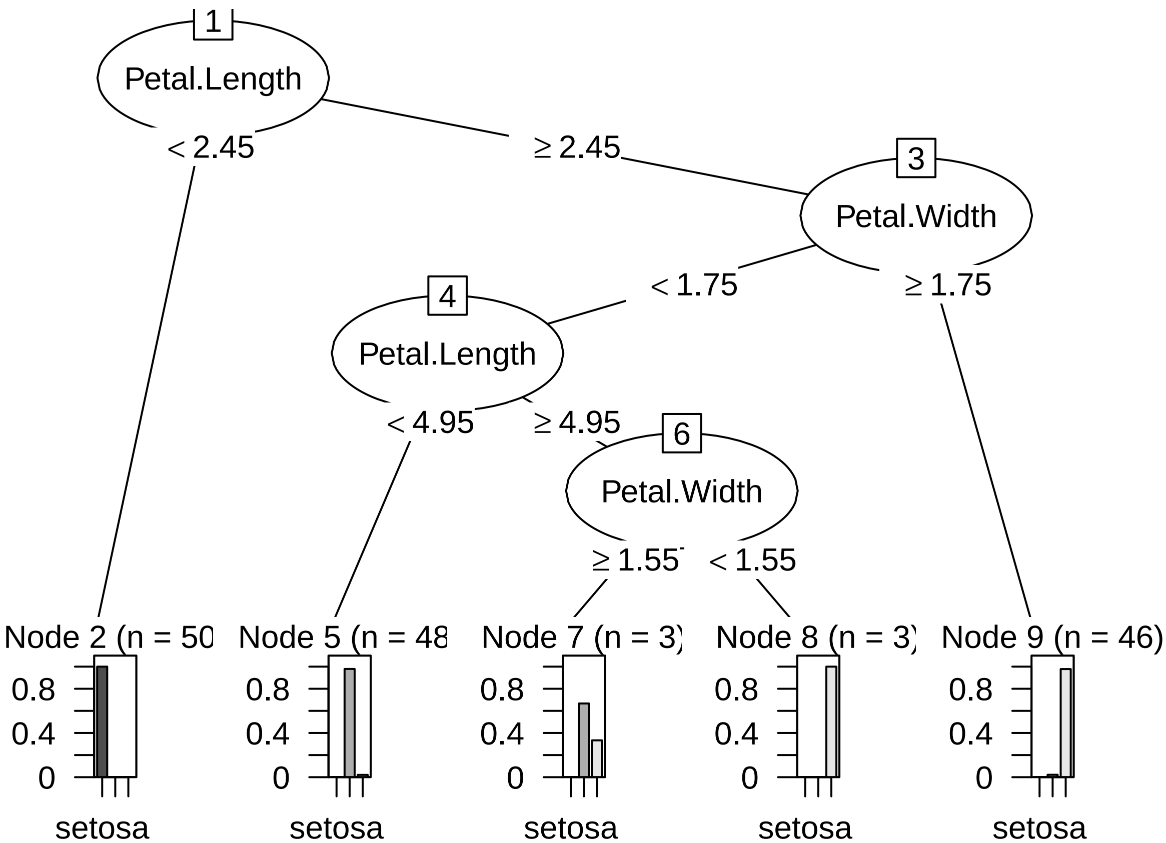

```
plot(model.cl, branch=.2, uniform=T, compress=T, margin=.1)
text(model.cl, all=T, use.n=T, fancy=T)
```



try to understand the different graphical parameters

another way to see the output

```
rparty.tree = as.party(model.cl)
plot(rparty.tree)
```



Classification error over the train sample

```
pred=predict(model.cl,type="class",iris)

table(pred,true=iris[,5])
```

```
##              true
## pred          setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         49         2
##   virginica       0          1        48
```

```r
error=3/150
error
```

```
## [1] 0.02
```

```r
#or
error = mean(pred!= iris[,5])
error
```

```
## [1] 0.02
```

Classification error over newdata

```r
newdata = rbind(c(5,3.45,1,0.2),c(5.8,3.5,5.11,2))
dimnames(newdata)=list(NULL, c("Sepal.Length","Sepal.Width","Petal.Length","Petal.Width"))
newdatas=data.frame(newdata)
pred=predict(model.cl,type="class",newdata=newdatas)
pred
```

```
##         1         2
##    setosa virginica
## Levels: setosa versicolor virginica
```

```r
pred=predict(model.cl,type="class",newdata=newdatas)
pred
```

```
##         1         2
##    setosa virginica
## Levels: setosa versicolor virginica
```

```r
pred=predict(model.cl,type="prob",newdata=newdatas)
pred
```

```
##   setosa versicolor virginica
## 1      1 0.00000000 0.0000000
## 2      0 0.02173913 0.9782609
```

```r
pred=predict(model.cl,type="vector",newdata=newdatas)
pred
```

```
## 1 2
## 1 3
```

Classification error over a test sample Divide the dataset 30 times randomly in train and test samples. For each split, fit a classification tree with the train sample and compute its prediction over the test sample. At the end, compute the mean, with standard deviation, of these errors.

```r
K=30
  error.cl=NULL
    n = nrow(iris)
    for(k in 1:K)   {
        smp=sample(n,round(n/3))
        learn=iris[-smp,]
        test=iris[smp,]
```

```r
            model.cl.learn=rpart(Species~.,cp=0.001,minsplit=5,data=learn)
            pred.cl=predict(model.cl.learn,type="class",test)
            error.cl[k] = mean(pred.cl!= test[,5])
        }

mean.error.cl=mean(error.cl)
sd.error.cl=sd(error.cl)
mean.error.cl
```

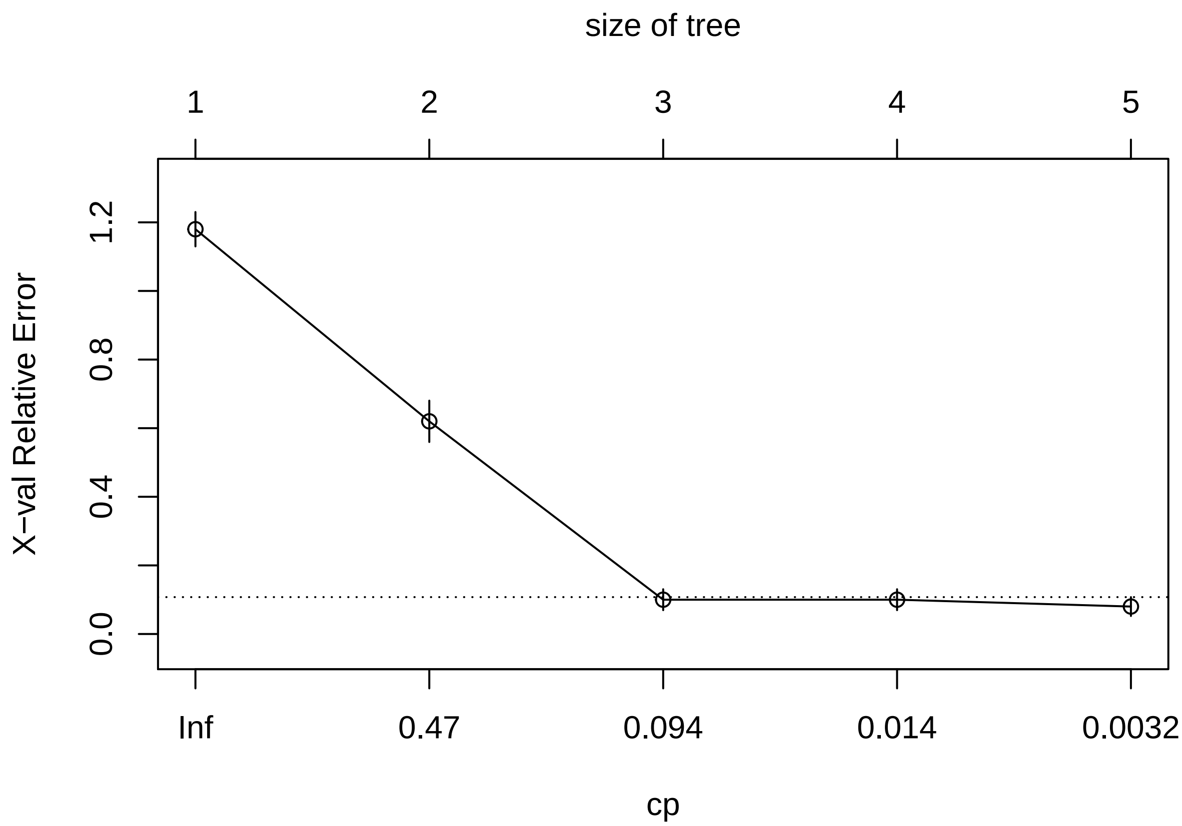
```
## [1] 0.052
```

```r
sd.error.cl
```

```
## [1] 0.02657455
```

Pruning

```r
plotcp(model.cl)
```




```r
printcp(model.cl)
```

```
##
## Classification tree:
## rpart(formula = Species ~ ., data = iris, cp = 0.001, minsplit = 5)
##
## Variables actually used in tree construction:
## [1] Petal.Length Petal.Width
##
## Root node error: 100/150 = 0.66667
##
```

```
## n= 150
##
##       CP nsplit rel error xerror     xstd
## 1 0.500      0      1.00   1.18 0.050173
## 2 0.440      1      0.50   0.62 0.060310
## 3 0.020      2      0.06   0.10 0.030551
## 4 0.010      3      0.04   0.10 0.030551
## 5 0.001      4      0.03   0.08 0.027520
```

According to the criterion of least error by cross validation we are left with the tree 5.

According to criterion 1-SE: with what tree do we stay?

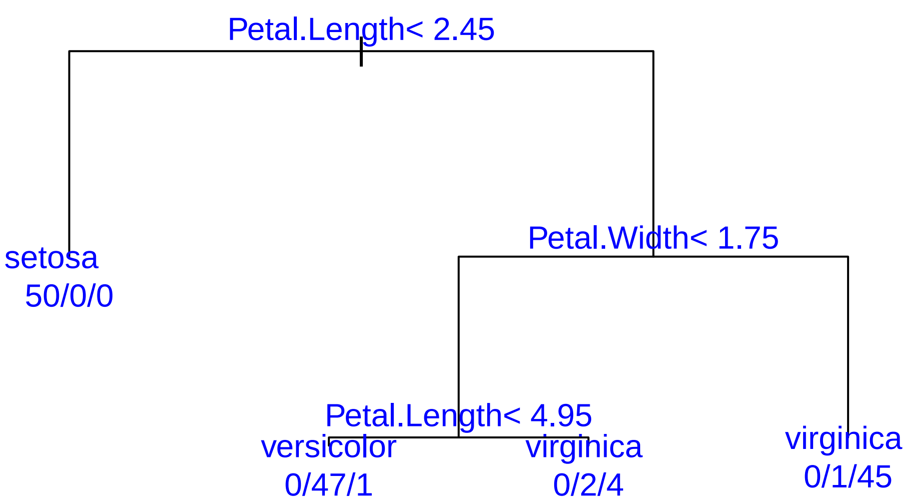
1-SE Rule: We are left with the simplest tree that has a minor error at the lowest error by VC + 1se (xerror + xstd).

BE CAREFUL: THESE RESULTS MAY VARY FOR THE DIFFERENT STEPS BECAUSE WHEN CALCULATING THE ERROR VALIDATION CROSSED, WE ARE SEPARATING THE SAMPLE RANDOMLY AND IT CAN BRING DIFFERENT RESULTS.

Contents of the printcp table: Remember that this table is normalized so that the error in the root node is 1 cp: cost-complexity parameter (as we have said before it is alpha / error by resolution in root node) nsplit: number of divisions (nsplit + 1 = number of terminal nodes) rel error: relative error xerror: error by cross validation xstd: standard deviation (it helps us analyze the 1-SE rule)

Let see (as an exercise) what happen if I choose to prune the tree to keep the sub tree with cp = 0.010.

```
model.cl.pod=prune(model.cl,cp=0.010)
plot(model.cl.pod,margin=0.1)
text(model.cl.pod, use.n=T,col="blue")
```



Classification error of the prunned tree

```
K=30
    error.cl.pod=NULL
    n = nrow(iris)
    for(k in 1:K)   {
        smp=sample(n,round(n/3))
        learn=iris[-smp,]
            test=iris[smp,]
            model.cl.pod.learn=rpart(Species~.,cp=0.010,learn)
            pred.cl.pod.learn=predict(model.cl.pod.learn,type="class",test)
```

```r
            error.cl.pod[k] = error = mean(pred.cl.pod.learn!= test[,5])}

mean.error.cl.pod=mean(error.cl.pod)
sd.error.cl.pod=sd(error.cl.pod)
mean.error.cl.pod
```

```
## [1] 0.06533333
```

```r
sd.error.cl.pod
```

```
## [1] 0.02569494
```

```r
mean.error.cl
```

```
## [1] 0.052
```

```r
sd.error.cl
```

```
## [1] 0.02657455
```

2- Regression trees

We use airquality data

```r
data=airquality
attach(data)
model.rg=rpart(Ozone~Solar.R+Wind+Temp,data=data)
model.rg
```

```
## n=116 (37 observations deleted due to missingness)
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 116 125143.1000 42.12931
##    2) Temp< 82.5 79  42531.5900 26.54430
##      4) Wind>=7.15 69  10919.3300 22.33333
##        8) Solar.R< 79.5 18    777.1111 12.22222 *
##        9) Solar.R>=79.5 51   7652.5100 25.90196
##         18) Temp< 77.5 33   2460.9090 21.18182 *
##         19) Temp>=77.5 18   3108.4440 34.55556 *
##      5) Wind< 7.15 10  21946.4000 55.60000 *
##    3) Temp>=82.5 37  22452.9200 75.40541
##      6) Temp< 87.5 20  12046.9500 62.95000
##       12) Wind>=8.9 7    617.7143 45.57143 *
##       13) Wind< 8.9 13   8176.7690 72.30769 *
##      7) Temp>=87.5 17   3652.9410 90.05882 *
```

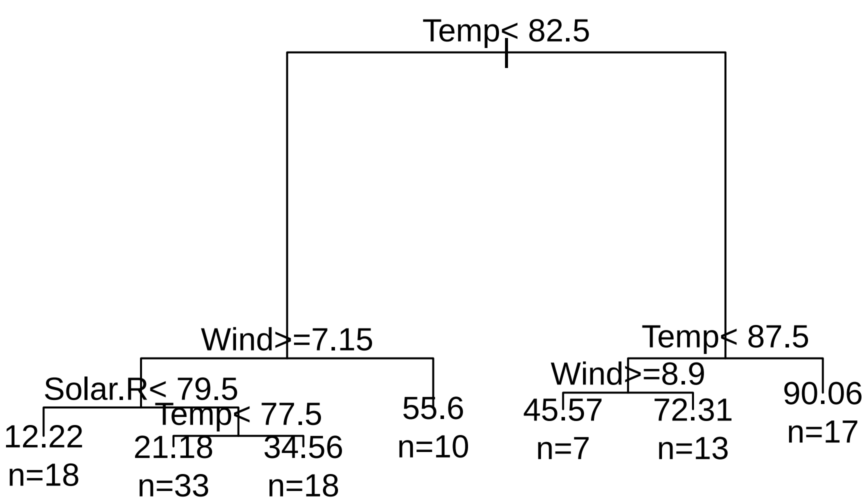
```r
summary(model.rg)
```

```
## Call:
## rpart(formula = Ozone ~ Solar.R + Wind + Temp, data = data)
##   n=116 (37 observations deleted due to missingness)
##
##           CP nsplit rel error    xerror      xstd
## 1 0.48071820      0 1.0000000 1.0156539 0.1689319
## 2 0.07723849      1 0.5192818 0.6305631 0.1799594
```

```
## 3 0.05396246      2 0.4420433 0.5983591 0.1747386
## 4 0.02598999      3 0.3880808 0.5306661 0.1536703
## 5 0.01989493      4 0.3620909 0.5399725 0.1566407
## 6 0.01664620      5 0.3421959 0.5230867 0.1559437
## 7 0.01000000      6 0.3255497 0.4898162 0.1404016
##
## Variable importance
##    Temp    Wind Solar.R
##      66      31       2
##
## Node number 1: 116 observations,    complexity param=0.4807182
##   mean=42.12931, MSE=1078.819
##   left son=2 (79 obs) right son=3 (37 obs)
##   Primary splits:
##       Temp    < 82.5  to the left,  improve=0.4807182, (0 missing)
##       Wind    < 6.6   to the right, improve=0.4042669, (0 missing)
##       Solar.R < 153   to the left,  improve=0.2108002, (5 missing)
##   Surrogate splits:
##       Wind < 6.6   to the right, agree=0.776, adj=0.297, (0 split)
##
## Node number 2: 79 observations,    complexity param=0.07723849
##   mean=26.5443, MSE=538.3746
##   left son=4 (69 obs) right son=5 (10 obs)
##   Primary splits:
##       Wind    < 7.15  to the right, improve=0.2272631, (0 missing)
##       Temp    < 77.5  to the left,  improve=0.2248966, (0 missing)
##       Solar.R < 153   to the left,  improve=0.1044972, (2 missing)
##
## Node number 3: 37 observations,    complexity param=0.05396246
##   mean=75.40541, MSE=606.8356
##   left son=6 (20 obs) right son=7 (17 obs)
##   Primary splits:
##       Temp    < 87.5  to the left,  improve=0.3007639, (0 missing)
##       Wind    < 10.6  to the right, improve=0.2739298, (0 missing)
##       Solar.R < 273.5 to the right, improve=0.1145269, (3 missing)
##   Surrogate splits:
##       Wind < 6.6   to the right, agree=0.676, adj=0.294, (0 split)
##
## Node number 4: 69 observations,    complexity param=0.01989493
##   mean=22.33333, MSE=158.2512
##   left son=8 (18 obs) right son=9 (51 obs)
##   Primary splits:
##       Solar.R < 79.5  to the left,  improve=0.22543670, (1 missing)
##       Temp    < 77.5  to the left,  improve=0.21455360, (0 missing)
##       Wind    < 10.6  to the right, improve=0.04850548, (0 missing)
##   Surrogate splits:
##       Temp < 63.5  to the left,  agree=0.794, adj=0.222, (1 split)
##       Wind < 16.05 to the right, agree=0.750, adj=0.056, (0 split)
##
## Node number 5: 10 observations
##   mean=55.6, MSE=2194.64
##
## Node number 6: 20 observations,    complexity param=0.02598999
##   mean=62.95, MSE=602.3475
```

```
##   left son=12 (7 obs) right son=13 (13 obs)
##   Primary splits:
##       Wind    < 8.9   to the right, improve=0.269982600, (0 missing)
##       Solar.R < 217.5 to the left,  improve=0.058145680, (3 missing)
##       Temp    < 85.5  to the right, improve=0.007674142, (0 missing)
##
## Node number 7: 17 observations
##   mean=90.05882, MSE=214.8789
##
## Node number 8: 18 observations
##   mean=12.22222, MSE=43.17284
##
## Node number 9: 51 observations,    complexity param=0.0166462
##   mean=25.90196, MSE=150.0492
##   left son=18 (33 obs) right son=19 (18 obs)
##   Primary splits:
##       Temp    < 77.5  to the left,  improve=0.27221870, (0 missing)
##       Wind    < 10.6  to the right, improve=0.09788213, (0 missing)
##       Solar.R < 255   to the right, improve=0.03603008, (1 missing)
##   Surrogate splits:
##       Wind < 10.6  to the right, agree=0.667, adj=0.056, (0 split)
##
## Node number 12: 7 observations
##   mean=45.57143, MSE=88.2449
##
## Node number 13: 13 observations
##   mean=72.30769, MSE=628.9822
##
## Node number 18: 33 observations
##   mean=21.18182, MSE=74.573
##
## Node number 19: 18 observations
##   mean=34.55556, MSE=172.6914
```

```r
plot(model.rg,margin=0.1)
text(model.rg, use.n=T)
```



Compute pseudo-R2 (deviance root node - deviance tree)/desviance root node where do you read deviance for model.rg?

Predictions

```r
predict(model.rg)
```

```
##        1        2        3        4        6        7        8        9
## 21.18182 21.18182 21.18182 21.18182 21.18182 21.18182 21.18182 12.22222
##       11       12       13       14       15       16       17       18
## 55.60000 21.18182 21.18182 21.18182 12.22222 21.18182 21.18182 12.22222
##       19       20       21       22       23       24       28       29
## 21.18182 12.22222 12.22222 21.18182 12.22222 21.18182 12.22222 34.55556
##       30       31       38       40       41       44       47       48
## 55.60000 21.18182 34.55556 90.05882 45.57143 34.55556 21.18182 21.18182
##       49       50       51       62       63       64       66       67
## 12.22222 21.18182 21.18182 72.30769 45.57143 34.55556 72.30769 45.57143
##       68       69       70       71       73       74       76       77
## 90.05882 90.05882 90.05882 90.05882 21.18182 34.55556 12.22222 55.60000
##       78       79       80       81       82       85       86       87
## 34.55556 72.30769 72.30769 45.57143 55.60000 72.30769 72.30769 34.55556
##       88       89       90       91       92       93       94       95
## 45.57143 90.05882 72.30769 72.30769 34.55556 55.60000 12.22222 12.22222
##       96       97       98       99      100      101      104      105
## 72.30769 72.30769 72.30769 90.05882 90.05882 90.05882 45.57143 34.55556
##      106      108      109      110      111      112      113      114
## 34.55556 12.22222 55.60000 21.18182 34.55556 34.55556 21.18182 12.22222
##      116      117      118      120      121      122      123      124
## 34.55556 55.60000 72.30769 90.05882 90.05882 90.05882 90.05882 90.05882
##      125      126      127      128      129      130      131      132
## 90.05882 90.05882 90.05882 72.30769 45.57143 34.55556 34.55556 21.18182
##      133      134      135      136      137      138      139      140
## 21.18182 34.55556 21.18182 55.60000 12.22222 21.18182 55.60000 21.18182
##      141      142      143      144      145      146      147      148
## 12.22222 21.18182 34.55556 21.18182 12.22222 34.55556 12.22222 12.22222
##      149      151      152      153
## 55.60000 21.18182 21.18182 21.18182
```

```r
newdata = rbind(c(315,12,60),c(270,9,65))
dimnames(newdata)=list(NULL, c("Solar.R","Wind","Temp"))
newdata=data.frame(newdata)
pred=predict(model.rg,newdata)
pred
```

```
##        1        2
## 21.18182 21.18182
```

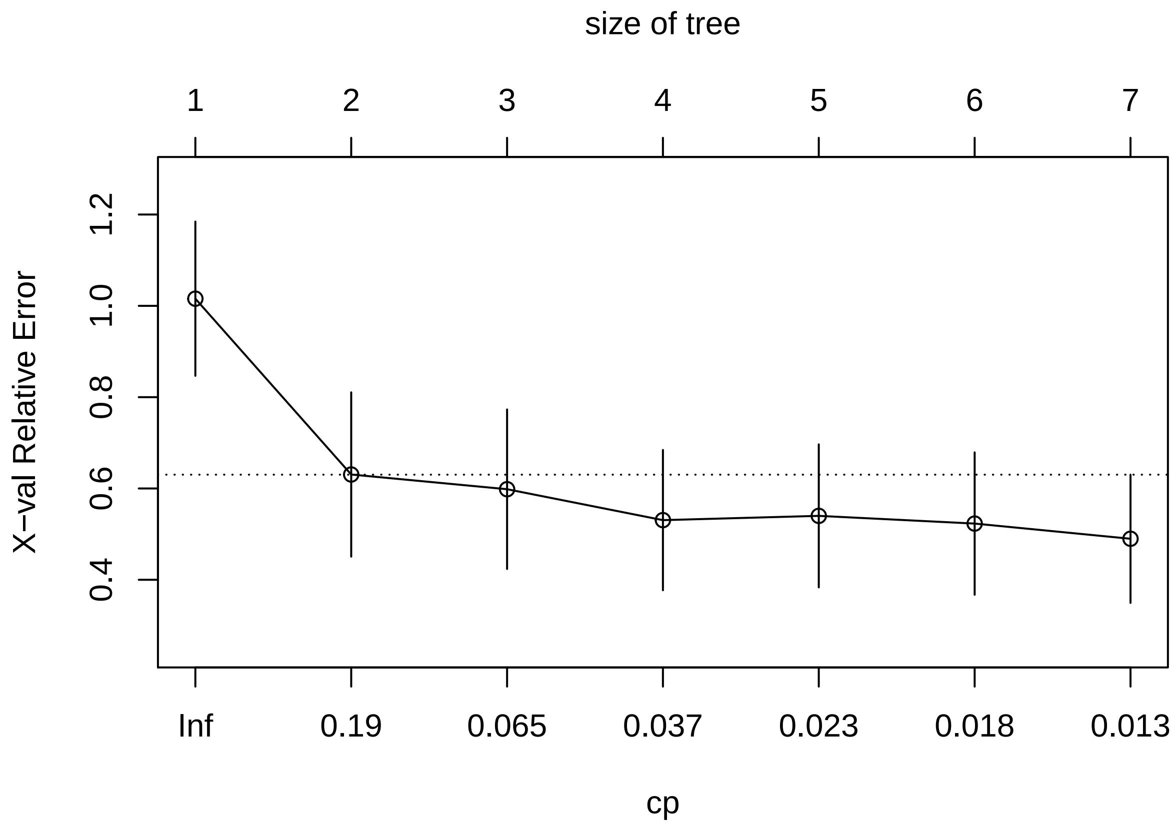
Kept the tree according to the cp criterion (crossvalidation and 1-SE Rule).

```r
printcp(model.rg)
```

```
##
## Regression tree:
## rpart(formula = Ozone ~ Solar.R + Wind + Temp, data = data)
##
## Variables actually used in tree construction:
## [1] Solar.R Temp    Wind
##
## Root node error: 125143/116 = 1078.8
##
```

```
## n=116 (37 observations deleted due to missingness)
##
##          CP nsplit rel error  xerror    xstd
## 1 0.480718      0   1.00000 1.01565 0.16893
## 2 0.077238      1   0.51928 0.63056 0.17996
## 3 0.053962      2   0.44204 0.59836 0.17474
## 4 0.025990      3   0.38808 0.53067 0.15367
## 5 0.019895      4   0.36209 0.53997 0.15664
## 6 0.016646      5   0.34220 0.52309 0.15594
## 7 0.010000      6   0.32555 0.48982 0.14040
```

```r
plotcp(model.rg)
```



Here some ways to select cp:

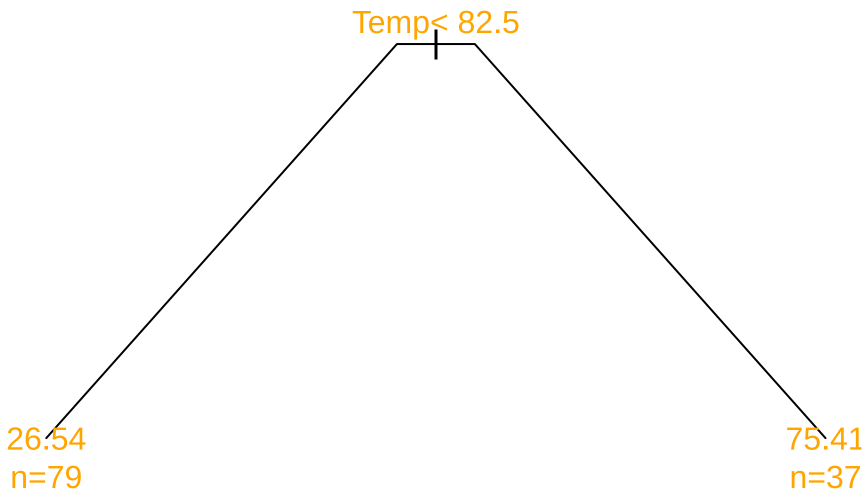
```r
#With crossvalidation
cp.optx = model.rg$cptable[which.min(model.rg$cptable[,"xerror"]),"CP"]
cp.optx
```

```
## [1] 0.01
```

```r
#with 1-SErule
xerror=model.rg$cptable[,4]
xstd <- model.rg$cptable[, 5]
t.opt <- min(seq(along = xerror)[xerror <= min(xerror) + xstd])
cp.opt1SE=model.rg$cptable[t.opt,1]
cp.opt1SE
```

```
## [1] 0.07723849
```

```r
#We kept with the tree which satisfies 1SE rule
model.rg.pr=prune(model.rg,cp=cp.opt1SE)
plot(model.rg.pr,branch=0.1,margin=0.1)
text(model.rg.pr, use.n=T,col="orange")
```



Classification error

```r
K=30
    error.rg.pr=NULL
    n = nrow(data)
    for(k in 1:K)   {
        smp=sample(n,round(n/3))
        learn=data[-smp,]
        learn=learn[,1:4]
        test=data[smp,]
        test=test[,1:4]
            model.rg.pr.learn=rpart(Ozone~Solar.R+Wind+Temp,learn)
            pred= predict (model.rg.pr.learn,test)
            sin.na=na.omit(cbind(test[,1],pred))
            error.rg.pr[k] = sqrt(mean((sin.na[,1]-sin.na[,2])^2))
    }

mean.error.rg.pr=mean(error.rg.pr)
sd.error.rg.pr=sd(error.rg.pr)

mean.error.rg.pr
```

```
## [1] 23.09413
```

```r
sd.error.rg.pr
```

```
## [1] 4.252215
```

The final model is the one build over all data!