10 - Trees

R Code for analyzing CART regression trees

SYS 4582/6018 | Spring 2019

10-trees_demo.pdf

Contents

1 Trees Intro

1.1 Required R Packages

We will be using the R packages of:

- `rpart` for classification and regression trees (CART)
- `rpart.plot` for `prp()` which allows more plotting control for trees
- `randomForest` for `randomForest()` function
- `ISLR` for Hitters baseball data
- `tidyverse` for data manipulation and visualization

```
library(ISLR)
library(rpart)
library(rpart.plot)
library(randomForest)
library(tidyverse)
```

1.2 Baseball Salary Data

The goal is to build models to predict the (log) salary of baseball players

```
head(bball)
```

```
#>   AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI CWalks
#> 2   479  130    18   66  72    76     3   1624   457     63   224  266    263
#> 3   496  141    20   65  78    37    11   5628  1575    225   828  838    354
#> 4   321   87    10   39  42    30     2    396   101     12    48   46     33
#> 5   594  169     4   74  51    35    11   4408  1133     19   501  336    194
#> 6   185   37     1   23   8    21     2    214    42      1    30    9     24
#> 8   323   81     6   26  32     8     2    341    86      6    32   34      8
#>   League Division PutOuts Assists Errors     Y NewLeague
#> 2      A        W     880      82     14 6.174         A
#> 3      N        E     200      11      3 6.215         N
#> 4      N        E     805      40      4 4.516         N
#> 5      A        W     282     421     25 6.620         A
#> 6      N        E      76     127      7 4.248         A
#> 8      N        W     143     290     19 4.317         N
```

2 Regression Tree

2.1 Build Tree

```
################################################################
#-- Regression Trees in R
# trees are in many packages: rpart, tree, party, ...
# there are also many packages to display tree results
#
# Formulas: you don't need to specify interactions as the tree does this
#  naturally.
################################################################
#-- Build Tree
library(rpart)
tree = rpart(Y~., data=bball)
summary(tree, cp=1)
```

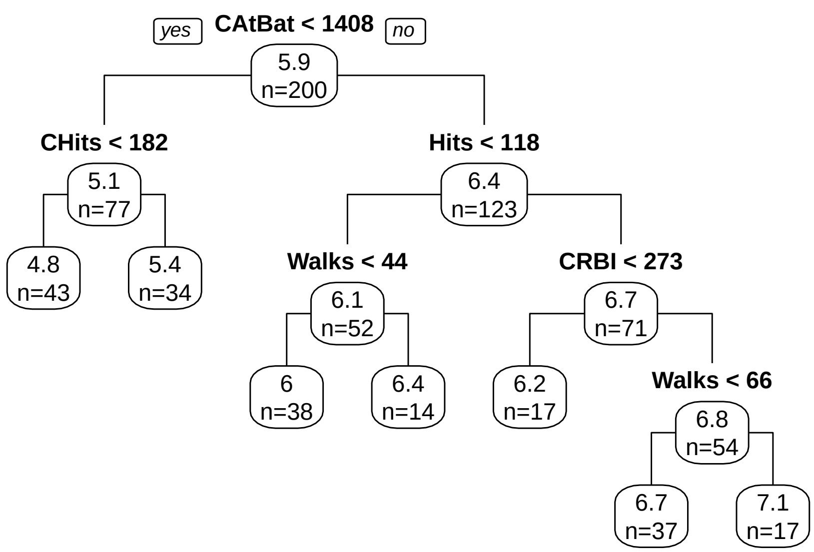
```
#> Call:
#> rpart(formula = Y ~ ., data = bball)
#>   n= 200
#>
```

```
#>        CP nsplit rel error xerror    xstd
#> 1 0.59845      0    1.0000 1.0038 0.07401
#> 2 0.06632      1    0.4015 0.4337 0.05064
#> 3 0.05009      2    0.3352 0.4164 0.06062
#> 4 0.03374      3    0.2851 0.3772 0.06031
#> 5 0.01442      4    0.2514 0.3509 0.05445
#> 6 0.01296      5    0.2370 0.3618 0.05360
#> 7 0.01000      6    0.2240 0.3667 0.05395
#>
#> Variable importance
#> CAtBat  CHits  CRuns   CRBI CWalks  Years   Hits  AtBat  Walks   Runs    RBI
#>     17     17     16     15     15     11      2      2      2      1      1
#> CHmRun
#>      1
#>
#> Node number 1: 200 observations
#>   mean=5.905, MSE=0.7495
```

```
length(unique(tree$where))       # number of leaf nodes
```

```
#> [1] 7
```

```
#-- Plot Tree
library(rpart.plot)    # for prp() which allows more plotting control
prp(tree, type=1, extra=1, branch=1)
```




```
# rpart() functions can also plot (just not as good):
#   plot(tree, uniform=TRUE)
#   text(tree, use.n=TRUE, xpd=TRUE)
```

2.2 Evaluate Tree

```
#- mean squared error function
mse <- function(yhat, y){
  yhat = as.matrix(yhat)
  apply(yhat, 2, function(f) mean((f-y)^2))
}


mse(predict(tree), bball$Y)              # training error
```

```
#> [1] 0.1679
```

```r
mse(predict(tree, X.test), Y.test)      # testing error
```

```
#> [1] 0.3301
```

Build a more complex tree

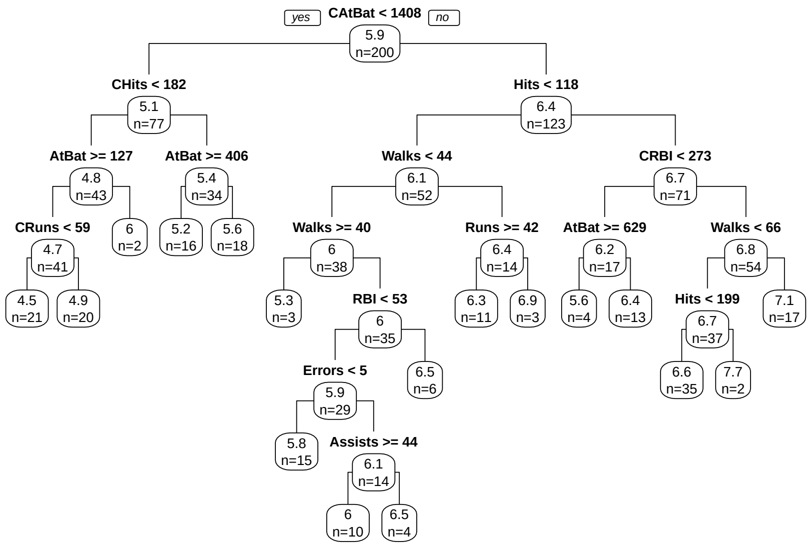
```r
#-- More complex tree
# see ?rpart.control() for details
# xval: number of cross-validations
# minsplit: min obs to still allow a split
# cp: complexity parameter

tree2 = rpart(Y~., data=bball, xval=0, minsplit=5, cp=0.005)
summary(tree2, cp=1)
```

```
#> Call:
#> rpart(formula = Y ~ ., data = bball, xval = 0, minsplit = 5,
#>     cp = 0.005)
#>   n= 200
#>
#>           CP nsplit rel error
#> 1  0.598450      0    1.0000
#> 2  0.066317      1    0.4015
#> 3  0.050093      2    0.3352
#> 4  0.033745      3    0.2851
#> 5  0.022051      4    0.2514
#> 6  0.014423      5    0.2293
#> 7  0.014007      6    0.2149
#> 8  0.013687      7    0.2009
#> 9  0.012957      8    0.1872
#> 10 0.011339      9    0.1743
#> 11 0.010736     10    0.1629
#> 12 0.009488     12    0.1415
#> 13 0.006548     13    0.1320
#> 14 0.006112     14    0.1254
#> 15 0.005530     15    0.1193
#> 16 0.005000     16    0.1138
#>
#> Variable importance
#>   CHits  CAtBat   CRuns    CRBI  CWalks   Years    Hits   AtBat    Runs   Walks
#>      16      16      16      14      14      10       3       3       2       2
#>     RBI  CHmRun PutOuts
#>       2       1       1
#>
#> Node number 1: 200 observations
#>   mean=5.905, MSE=0.7495
```

```r
length(unique(tree2$where))
```

```
#> [1] 17
```

```r
prp(tree2, type=1, extra=1, branch=1)
```




```
mse(predict(tree2), bball$Y)              # training error
```

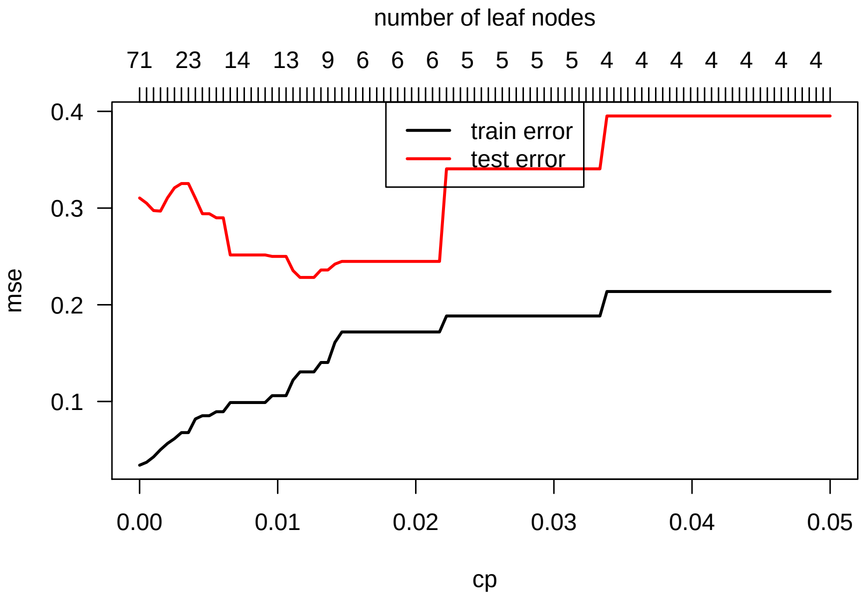
```
#> [1] 0.08528
```

```
mse(predict(tree2, X.test), Y.test)       # testing error
```

```
#> [1] 0.2942
```

Now, fit a set of tree for range of `cp` values.

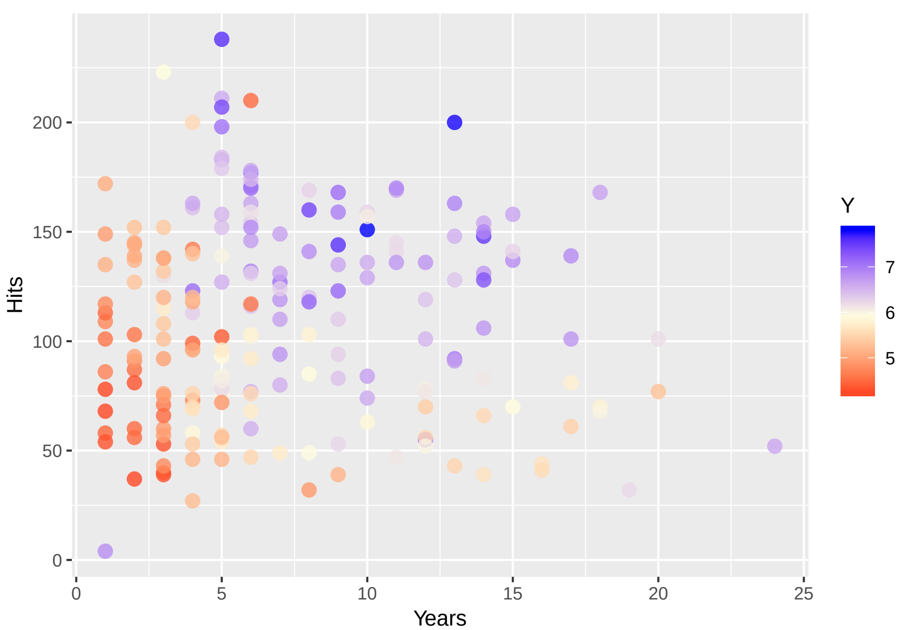
```
cp = seq(.05,0,length=100)   # cp is like a penalty on the tree size
for(i in 1:length(cp)){
  if(i == 1){train.error = test.error = nleafs = numeric(length(cp))}
  tree.fit = rpart(Y~.,data=bball, xval=0, minsplit=5, cp=cp[i])
  train.error[i] = mse(predict(tree.fit),bball$Y)            # training error
  test.error[i] = mse(predict(tree.fit,X.test),Y.test)    # testing error
  nleafs[i] = length(unique(tree.fit$where))
}

plot(range(cp),range(train.error,test.error),typ='n',xlab="cp",ylab="mse",las=1)
lines(cp,train.error,col="black",lwd=2)
lines(cp,test.error,col="red",lwd=2)
legend("top",c('train error','test error'),col=c("black","red"),lwd=2)
axis(3,at=cp,labels=nleafs)
mtext("number of leaf nodes",3,line=2.5)
```



2.3 Regression Tree example with 2 dimensions only

Consider the two variables `Years` and `Hits` and their relationship to `Y`.

```
####################################################################
#-- Regression Tree Examples for 2D
####################################################################
library(ggplot2)

#-- 2D plot (using only Years and Hits)
p2D = ggplot(bball) + #scale_size_area(max_size=5) +
            scale_color_gradient2(midpoint=mean(bball$Y),mid="lightyellow",low="red",high="
p2D +
  geom_point(aes(x=Years, y=Hits, color=Y), alpha=.8, size=3)
```



Let's fit a tree with the two predictors

```
#-- Fit tree to only Years and Hits
tree3 = rpart(Y~Years+Hits, data=bball)
```

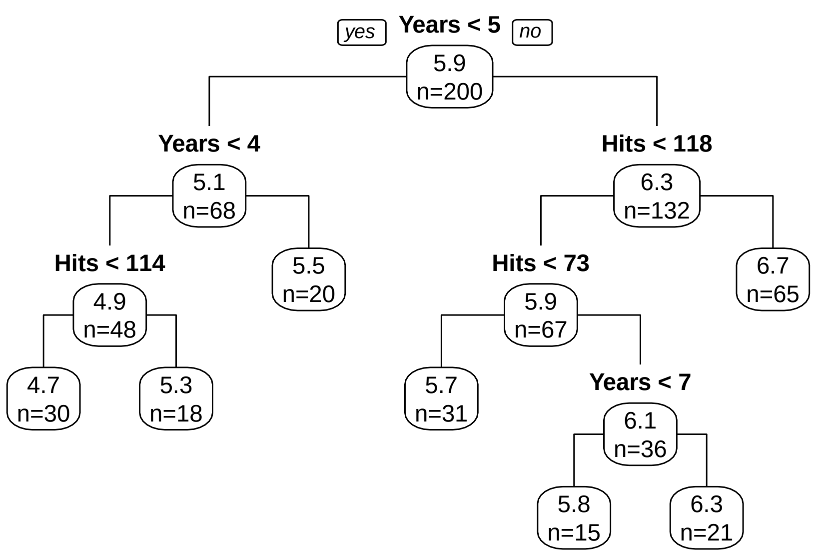
```
summary(tree3,cp=1)
```

```
#> Call:
#> rpart(formula = Y ~ Years + Hits, data = bball)
#>   n= 200
#>
#>         CP nsplit rel error xerror    xstd
#> 1 0.45116      0    1.0000 1.0082 0.07436
#> 2 0.13082      1    0.5488 0.6199 0.06938
#> 3 0.03152      2    0.4180 0.4945 0.06907
#> 4 0.02193      3    0.3865 0.4863 0.07004
#> 5 0.01660      4    0.3646 0.5007 0.07313
#> 6 0.01000      6    0.3314 0.4585 0.07053
#>
#> Variable importance
#> Years  Hits
#>    74    26
#>
#> Node number 1: 200 observations
#>   mean=5.905, MSE=0.7495
```

```
length(unique(tree3$where))                    # number of leaf nodes
```

```
#> [1] 7
```

```
prp(tree3, type=1, extra=1, branch=1)
```



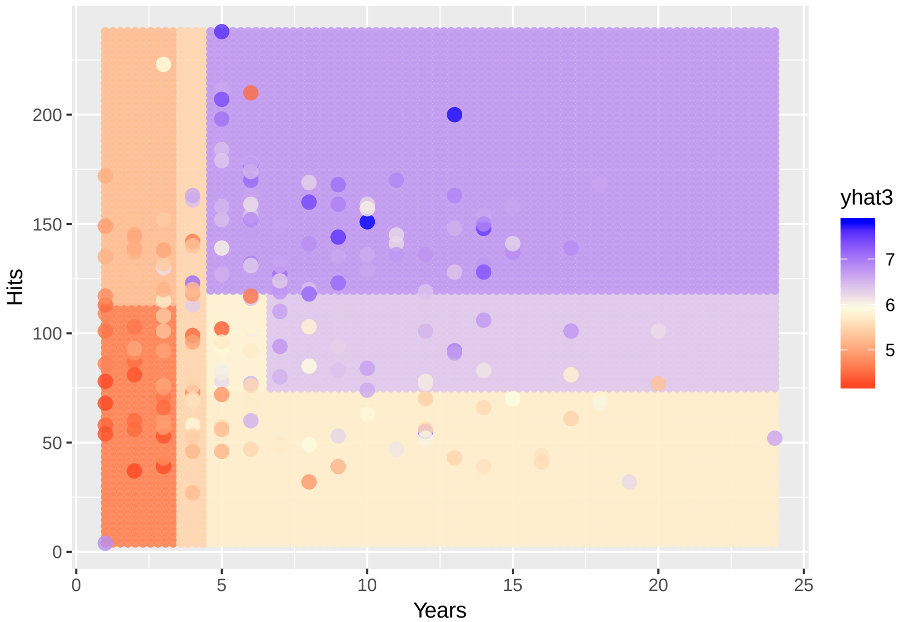

```
mse(predict(tree3), bball$Y)                   # training error
```

```
#> [1] 0.2484
```

```
mse(predict(tree3,X.test),Y.test)              # testing error
```

```
#> [1] 0.3873
```

```
#-- Plot Results
grid = expand.grid(Years = seq(min(bball$Years),max(bball$Years),length=90),
                   Hits = seq(min(bball$Hits),max(bball$Hits),length=90))
grid$yhat3 = predict(tree3,newdata = grid)

p2D + geom_point(data=grid,aes(x=Years, y=Hits, color=yhat3),alpha=.9) +
  geom_point(aes(x=Years, y=Hits, color=Y), alpha=.8, size=3)
```



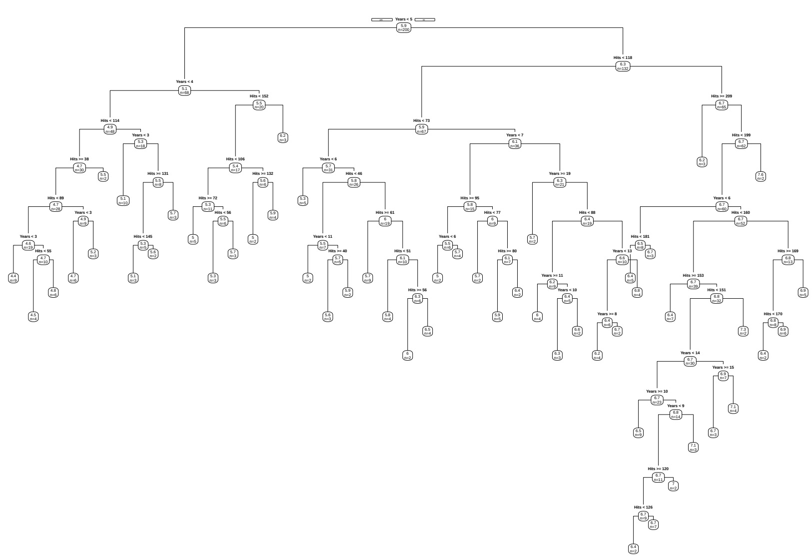
This shows the leaf regions (in 2D).

And we can also use more complex trees:

```
#-- Fit more complex tree to only Years and Hits
tree4 = rpart(Y~Years+Hits,data=bball,xval=0,minsplit=5,cp=0.001)
length(unique(tree4$where))             # number of leaf nodes
```

```
#> [1] 52
```

```
prp(tree4, type=1, extra=1, branch=1)
```



```
mse(predict(tree4), bball$Y)              # training error
```

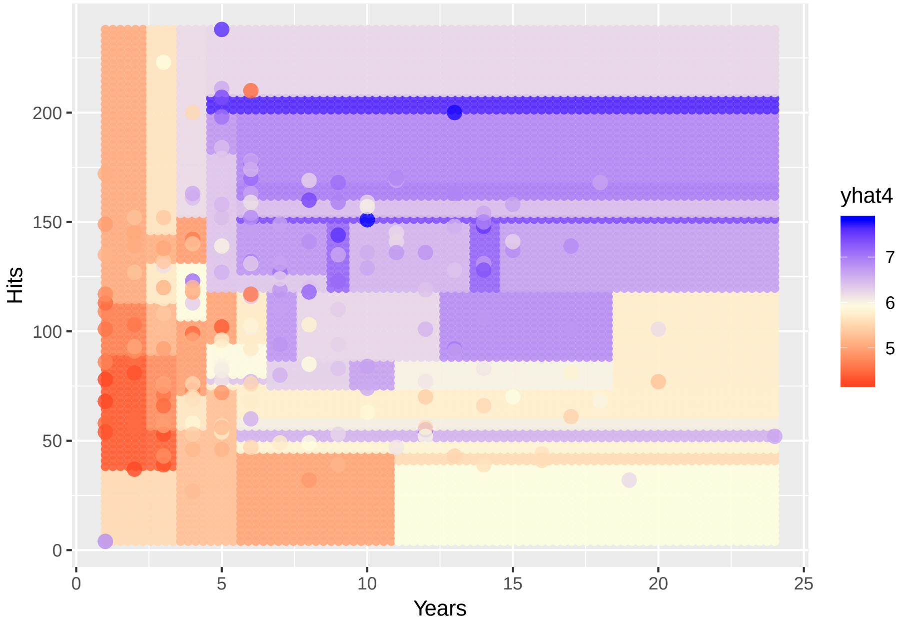
```
#> [1] 0.1274
```

```
mse(predict(tree4,X.test), Y.test)        # testing error
```

```
#> [1] 0.3822
```

```
#-- Plot Results
grid$yhat4 = predict(tree4,newdata = grid)
```

```
p2D + geom_point(data=grid,aes(x=Years,y=Hits,color=yhat4),alpha=.9) +
  geom_point(aes(x=Years,y=Hits,color=Y),alpha=.8, size=3)
```



3 Details of Splitting (for Regression Trees)

Consider only two dimensions, `hits` and `years` on which to make first split.

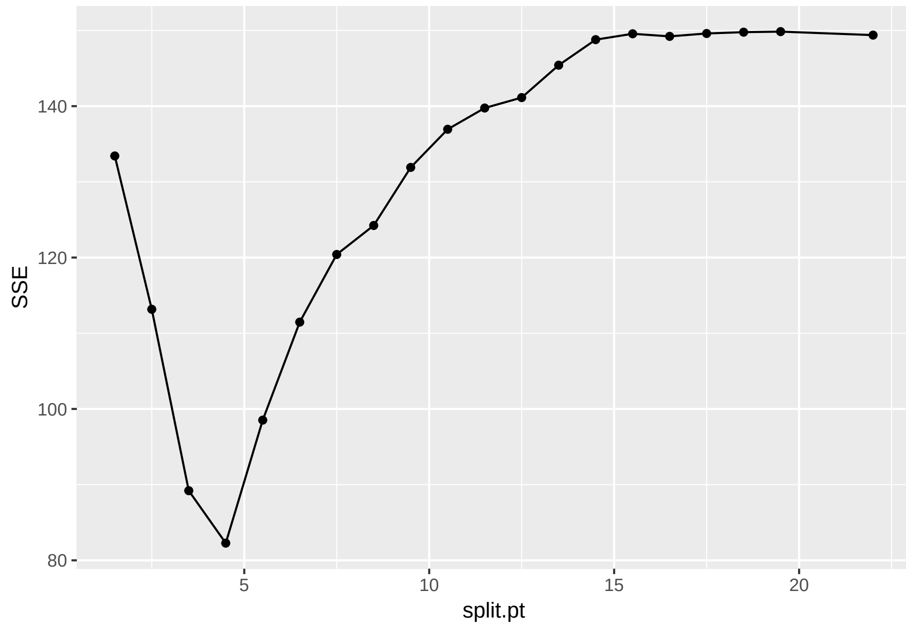
3.1 First Split

3.1.1 Split on Years

```
## Split by Years
years = split_info(x=bball$Years, y=bball$Y)
head(years)
```

```
#> # A tibble: 6 x 9
#>   split.pt   n.L   n.R est.L est.R SSE.L SSE.R   SSE  gain
#>      <dbl> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1      1.5    13   187  4.82  5.98  5.44 128.  133.   16.5
#> 2      2.5    27   173  4.82  6.07  7.18 106.  113.   36.7
#> 3      3.5    48   152  4.92  6.21 12.6   76.6  89.2  60.7
#> 4      4.5    68   132  5.09  6.32 26.0   56.2  82.3  67.6
#> 5      5.5    92   108  5.36  6.37 56.7   41.9  98.5  51.4
#> 6      6.5   115    85  5.53  6.41 81.5   30.0 111.   38.4
```

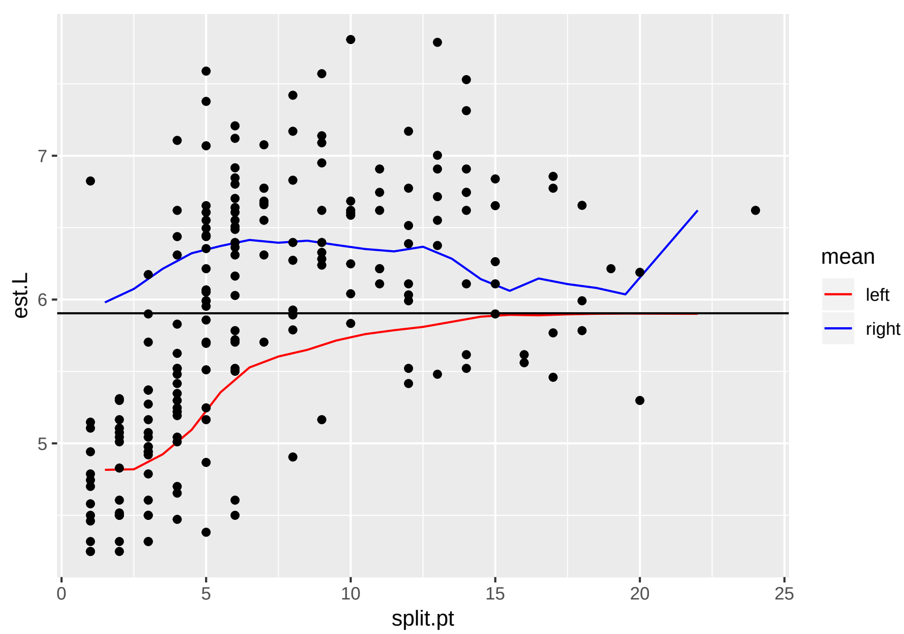
```
ggplot(years,aes(x=split.pt,y=SSE)) + geom_line() + geom_point()
```




```
filter(years, min_rank(SSE) == 1)   # optimal split point for Years
```

```
#> # A tibble: 1 x 9
#>   split.pt   n.L   n.R est.L est.R SSE.L SSE.R   SSE  gain
#>      <dbl> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1      4.5    68   132  5.09  6.32  26.0  56.2  82.3  67.6
```

```
ggplot(years,aes(x=split.pt)) +
  geom_line(aes(y=est.L,color="left")) +           # mean left of split pt
  geom_line(aes(y=est.R,color="right")) +          # mean right of split pt
  geom_hline(yintercept=mean(bball$Y))+            # overall mean
  scale_color_manual("mean",values=c('left'='red','right'='blue')) +
  geom_point(data=bball,aes(x=Years,y=Y))          # add points
```

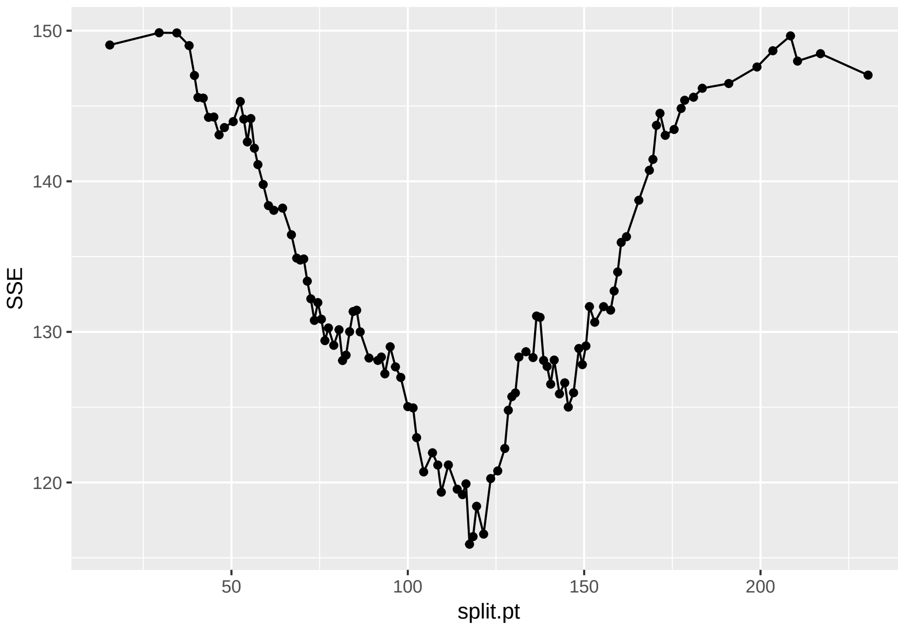


3.1.2 Split on Hits

```
## Split by Hits
hits = split_info(x=bball$Hits,y=bball$Y)
head(hits)
```

```
#> # A tibble: 6 x 9
#>   split.pt   n.L   n.R est.L est.R SSE.L SSE.R   SSE   gain
#>      <dbl> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
#> 1     15.5     1   199  6.82  5.90  0     149.  149. 0.850
#> 2     29.5     2   198  6.04  5.90  1.24  149.  150. 0.0348
#> 3     34.5     4   196  5.80  5.91  2.33  148.  150. 0.0465
#> 4     38       5   195  5.49  5.92  4.25  145.  149. 0.890
#> 5     39.5     8   192  5.32  5.93  5.51  142.  147. 2.87
#> 6     40.5     9   191  5.23  5.94  6.10  139.  146. 4.33
```

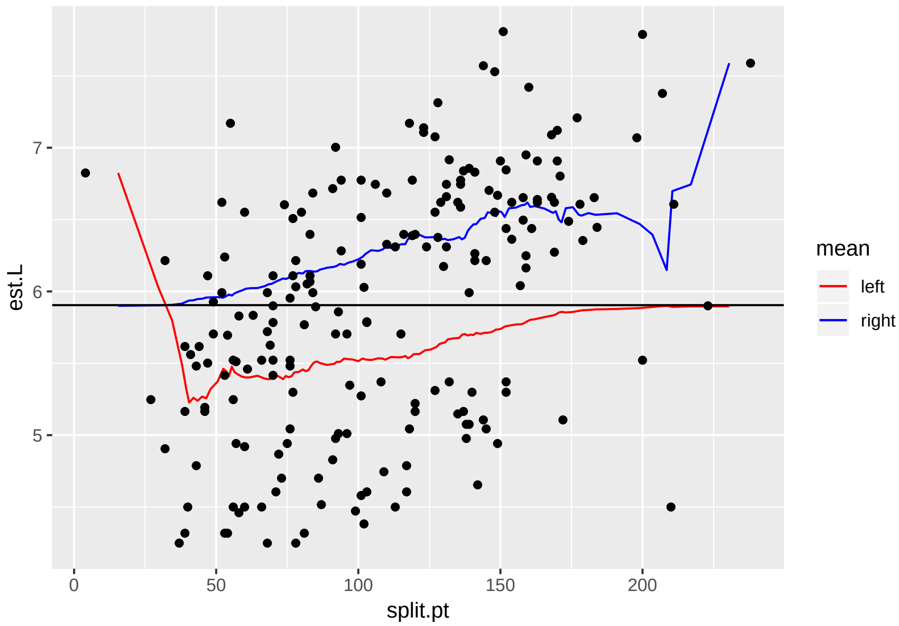
```
ggplot(hits,aes(x=split.pt,y=SSE)) + geom_line() + geom_point()
```




```
filter(hits, min_rank(SSE)==1)     # optimal split point for Hits
```

```
#> # A tibble: 1 x 9
#>   split.pt   n.L   n.R est.L est.R SSE.L SSE.R   SSE  gain
#>      <dbl> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1     118.   111    89  5.54  6.37  62.6  53.3  116.  34.0
```

```
ggplot(hits,aes(x=split.pt)) +
  geom_line(aes(y=est.L,color="left")) +           # mean left of split pt
  geom_line(aes(y=est.R,color="right")) +          # mean right of split pt
  geom_hline(yintercept=mean(bball$Y))+          # overall mean
  scale_color_manual("mean",values=c('left'='red','right'='blue')) +
  geom_point(data=bball,aes(x=Hits,y=Y))            # add points
```



3.1.3 Find best variable to split on

```
## No splits
sum((bball$Y-mean(bball$Y))^2)    # SSE if no splits are made
```

```
#> [1] 149.9
```

```
# (nrow(bball)-1)*var(bball$Y)



## Results (see function split_metrics at top of file)
#  splitting on Years gives the best reduction in SSE, so we would split on
#  Years (at a value of 4.5).
sum((bball$Y-mean(bball$Y))^2)        # no split
```

```
#> [1] 149.9
```

```
filter(years, min_rank(SSE)==1)       # split on years
```

```
#> # A tibble: 1 x 9
#>   split.pt   n.L   n.R est.L est.R SSE.L SSE.R   SSE  gain
#>      <dbl> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1      4.5    68   132  5.09  6.32  26.0  56.2  82.3  67.6
```

```
filter(hits, min_rank(SSE)==1)        # split on hits
```

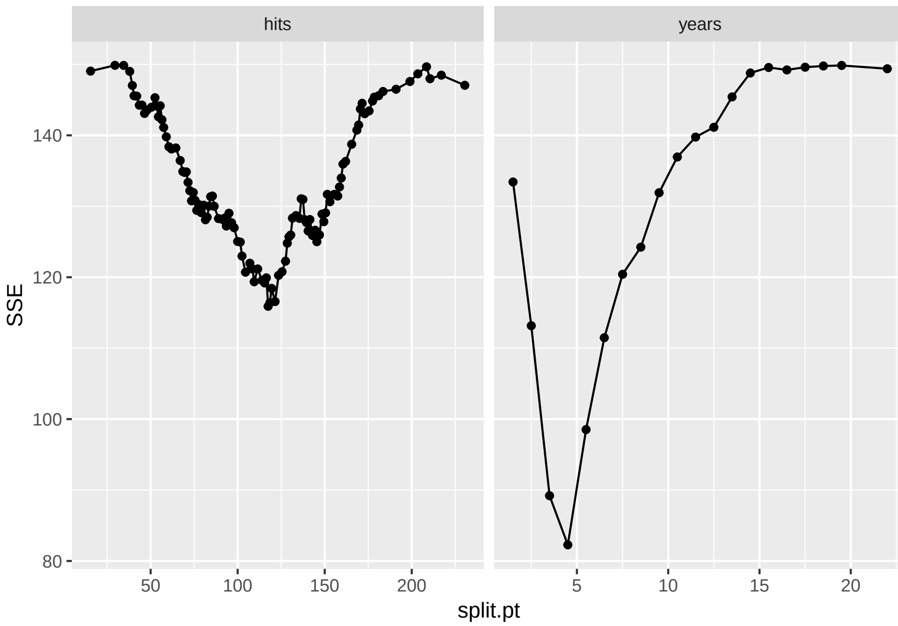
```
#> # A tibble: 1 x 9
#>   split.pt   n.L   n.R est.L est.R SSE.L SSE.R   SSE  gain
#>      <dbl> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1     118.   111    89  5.54  6.37  62.6  53.3  116.  34.0
```

```
split_metrics(bball$Years,bball$Y, 4.5)
```

```
#> # A tibble: 2 x 3
#>   region   SSE     n
#>   <chr>  <dbl> <int>
#> 1 LEFT    26.0    68
#> 2 RIGHT   56.2   132
```

```
## Comparison of splitting on both variables
bind_rows(hits=hits, years=years, .id="split.var") %>%
```

```
ggplot(aes(x=split.pt, y=SSE)) + geom_line() + geom_point() +
facet_wrap(~split.var, scales="free_x")
```



3.2 Second Split

```
#-- 2nd Split
# now we have to compare 4 possibilities. We can split on Years or Hits, but
#  use data that has Years < 4.5 or Years > 4.5

left = (bball$Years<=4.5)                          # split point from previous step
years2.L = split_info(x=bball$Years[left],y=bball$Y[left])
years2.R = split_info(x=bball$Years[!left],y=bball$Y[!left])
hits2.L = split_info(x=bball$Hits[left],y=bball$Y[left])
hits2.R = split_info(x=bball$Hits[!left],y=bball$Y[!left])

#-- Find best region to split on
max(years2.L$gain,na.rm=TRUE)
```

```
#> [1] 4.725
```

```
max(years2.R$gain,na.rm=TRUE)
```

```
#> [1] 2.043
```

```
max(hits2.L$gain,na.rm=TRUE)
```

```
#> [1] 4.687
```

```
max(hits2.R$gain,na.rm=TRUE)
```

```
#> [1] 19.61
```

```
hits2.R[which.max(hits2.R$gain),]
```

```
#> # A tibble: 1 x 9
#>   split.pt   n.L   n.R est.L est.R SSE.L SSE.R   SSE  gain
#>      <dbl> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1     118.    67    65  5.94  6.71  20.7  15.9  36.6  19.6
```

```
# 2nd split on Hits <= 117.5 in region 2.
```

```r
#-- Summary of Splits
# Rule 1: Years < 4.5
# Rule 2: Years >= 4.5 & Hits < 117.5
# ...
prp(tree3, type=1, extra=1, branch=1)
```