07 - Trees: Demo

R Code for analyzing CART regression trees

SYS 6018 | Fall 2020

07-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

```r
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

```r
head(bball)
#>   AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI CWalks
#> 1   315   81     7   24  38    39    14   3449   835     69   321  414    375
#> 2   479  130    18   66  72    76     3   1624   457     63   224  266    263
#> 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
#> 7   298   73     0   24  24     7     3    509   108      0    41   37     12
#>   League Division PutOuts Assists Errors     Y NewLeague
#> 1      N        W     632      43     10 6.163         N
#> 2      A        W     880      82     14 6.174         A
#> 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
#> 7      A        W     121     283      9 4.605         A
```

2 Regression Tree

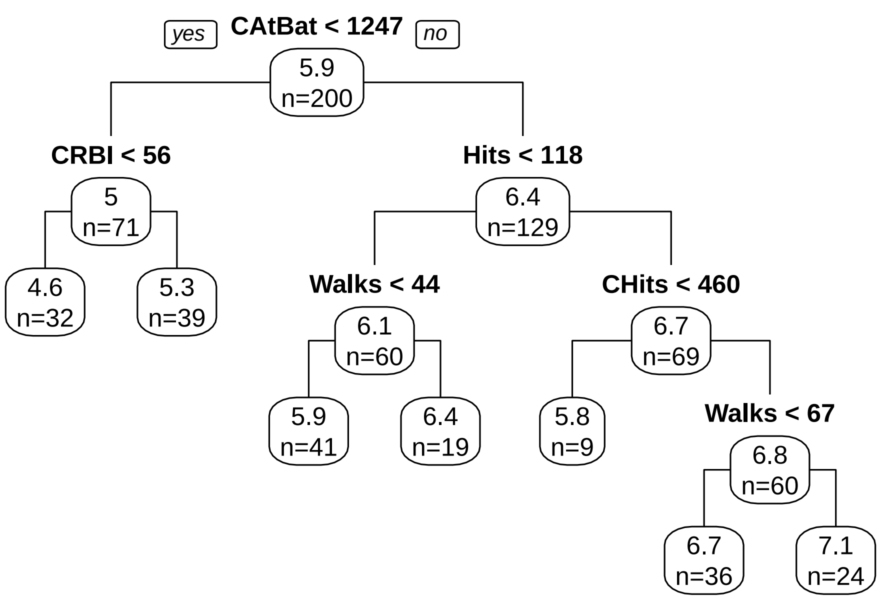
2.1 Build Tree

```r
#################################################################
#-- 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 0.61187      0    1.0000 1.0034 0.07223
#> 2 0.08077      1    0.3881 0.4361 0.04334
#> 3 0.05616      2    0.3074 0.3754 0.04487
#> 4 0.05037      3    0.2512 0.3534 0.04483
#> 5 0.01840      4    0.2008 0.2500 0.02677
#> 6 0.01381      5    0.1824 0.2553 0.02637
#> 7 0.01000      6    0.1686 0.2433 0.02594
#>
#> Variable importance
#> CAtBat  CHits  CRuns   CRBI CWalks CHmRun   Hits  AtBat   Runs  Walks    RBI
#>     17     17     16     15     14     11      2      2      2      2      1
#>  HmRun
#>      1
#>
#> Node number 1: 200 observations
#>   mean=5.898, MSE=0.8053
```

```
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] 0.1358
```

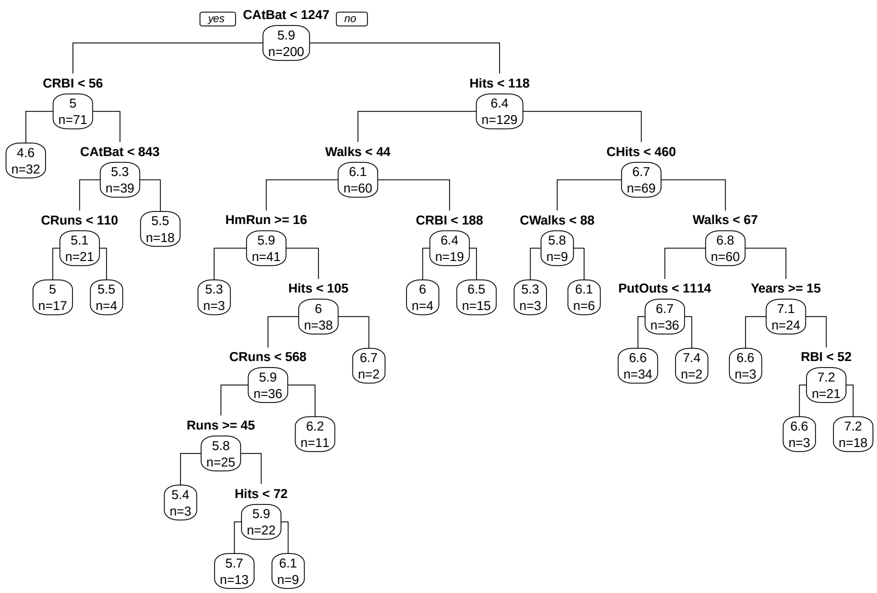
```
mse(predict(tree, X.test), Y.test)      # testing error
#> [1] 0.4931
```

Build a more complex tree

```
#-- 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  0.611866      0   1.00000
#> 2  0.080767      1   0.38813
#> 3  0.056162      2   0.30737
#> 4  0.050368      3   0.25121
#> 5  0.018404      4   0.20084
#> 6  0.013809      5   0.18243
#> 7  0.008264      6   0.16862
#> 8  0.007883      7   0.16036
#> 9  0.007740      8   0.15248
#> 10 0.007267      9   0.14474
#> 11 0.007129     10   0.13747
#> 12 0.006491     11   0.13034
#> 13 0.005916     12   0.12385
#> 14 0.005816     14   0.11202
#> 15 0.005332     15   0.10620
#> 16 0.005078     16   0.10087
#> 17 0.005000     18   0.09071
#>
#> Variable importance
#> CAtBat  CHits  CRuns   CRBI  CWalks CHmRun   Hits  AtBat   Runs    RBI  Walks
#>     17     17     16     15     13     10      2      2      2      2      2
#>  HmRun  Years
#>      1      1
#>
#> Node number 1: 200 observations
#>   mean=5.898, MSE=0.8053
```

```
length(unique(tree2$where))
#> [1] 19
```

```r

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

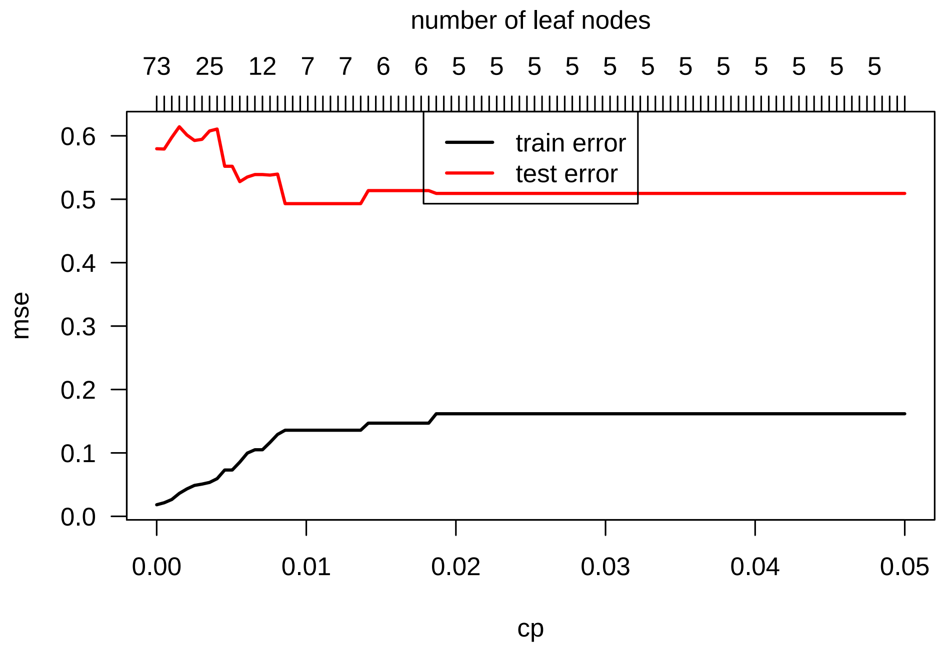



```r

mse(predict(tree2), bball$Y)              # training error
#> [1] 0.07305
```

```r
mse(predict(tree2, X.test), Y.test)      # testing error
#> [1] 0.552
```

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

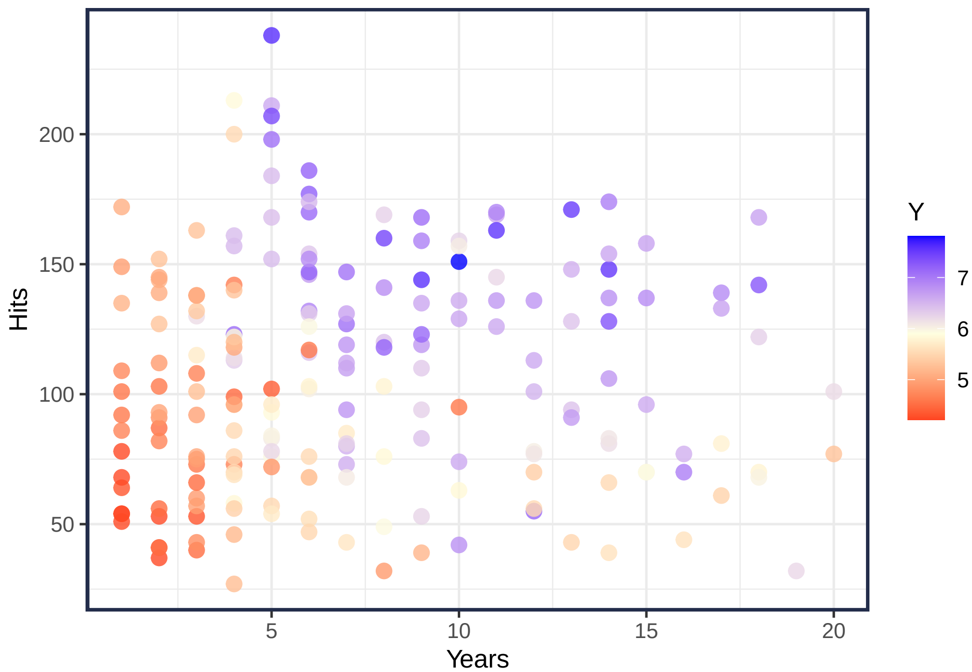
```r
cp = seq(.05,0,length=100)  # cp is like a penalty on the tree size
for(i in 1:length(cp)){
  if(i == 1){train.error = test.error = nleafs = numeric(length(cp))}
  tree.fit = rpart(Y~.,data=bball, xval=0, minsplit=5, cp=cp[i])
  train.error[i] = mse(predict(tree.fit),bball$Y)              # training error
  test.error[i] = mse(predict(tree.fit,X.test),Y.test)    # testing error
  nleafs[i] = length(unique(tree.fit$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="blue")
p2D +
  geom_point(aes(x=Years, y=Hits, color=Y), alpha=.8, size=3)
```

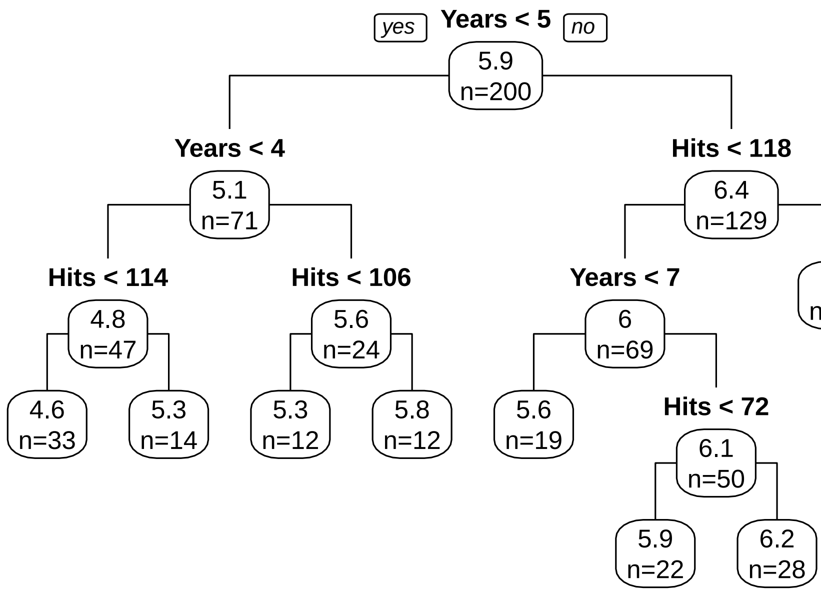


Let's fit a tree with the two predictors

```
#-- 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 0.47952      0    1.0000 1.0110 0.07293
#> 2 0.15162      1    0.5205 0.5722 0.06131
#> 3 0.05761      2    0.3689 0.4242 0.05369
#> 4 0.02587      3    0.3113 0.3831 0.04738
#> 5 0.01892      4    0.2854 0.3593 0.04643
#> 6 0.01019      5    0.2665 0.3427 0.04869
#> 7 0.01011      6    0.2563 0.3539 0.05074
#> 8 0.01000      7    0.2462 0.3539 0.05074
#>
#> Variable importance
#> Years  Hits
#>    73    27
#>
#> Node number 1: 200 observations
#>   mean=5.898, MSE=0.8053
```

```
length(unique(tree3$where))             # number of leaf nodes
#> [1] 8
```

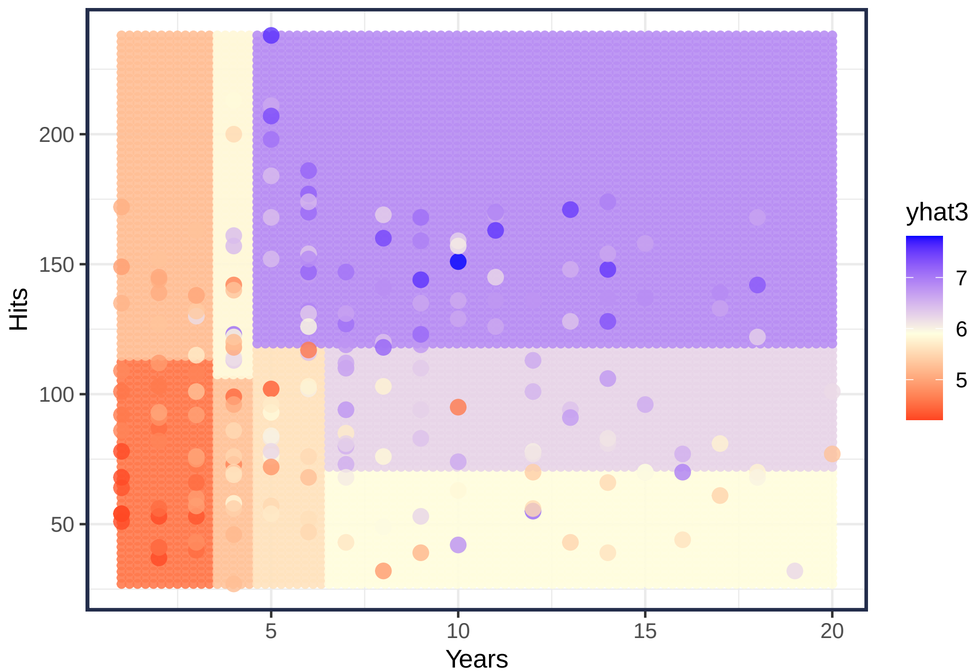
```
prp(tree3, type=1, extra=1, branch=1)
```




```
mse(predict(tree3), bball$Y)              # training error
#> [1] 0.1982
```

```
mse(predict(tree3,X.test),Y.test)         # testing error
#> [1] 0.5179
```

```


#-- 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)
```

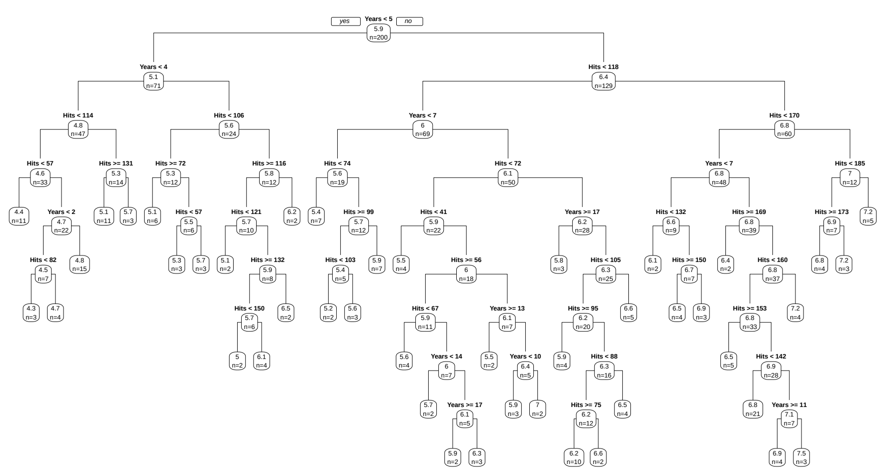


This shows the leaf regions (in 2D).

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] 44
```

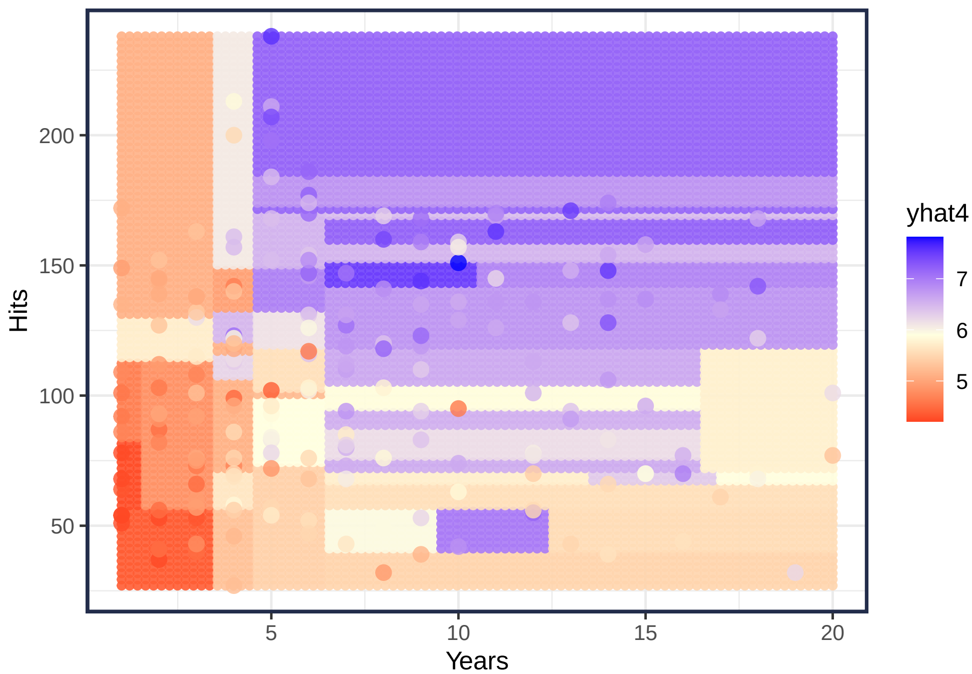
```
prp(tree4, type=1, extra=1, branch=1)
```



```
mse(predict(tree4), bball$Y)              # training error
#> [1] 0.09876
```

```
mse(predict(tree4,X.test), Y.test)        # testing error
#> [1] 0.6959
```

```


#-- 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.

See `trees.R` for the `split_info()` and `split_metrics()` functions

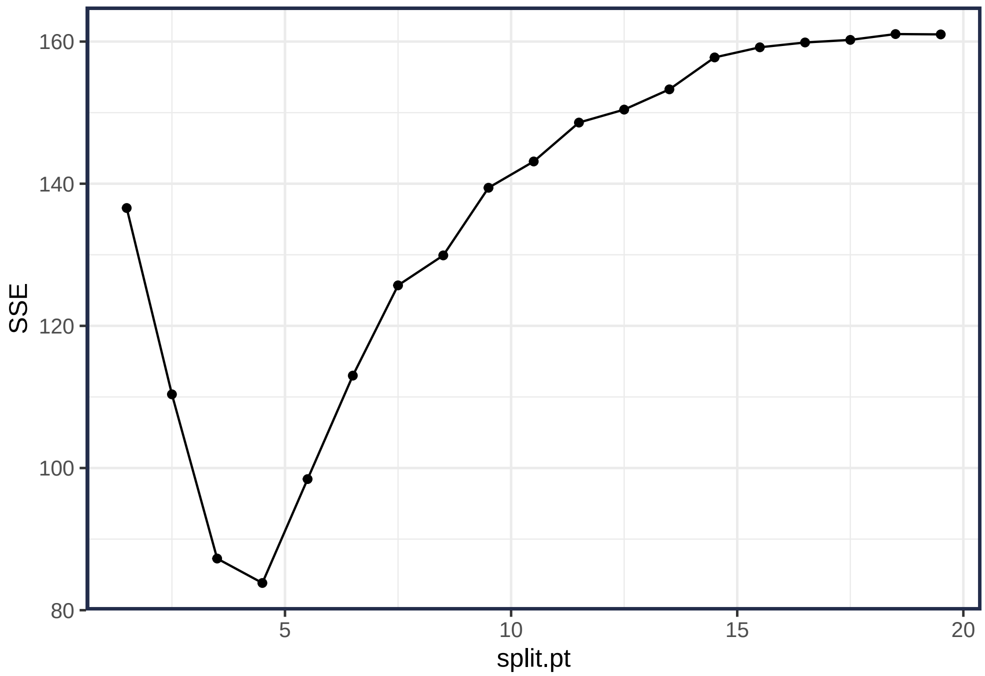
3.1 First Split

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  4.57  5.99  1.37 135.   137.   24.5
#> 2      2.5    29   171  4.68  6.11  3.63 107.   110.   50.7
```

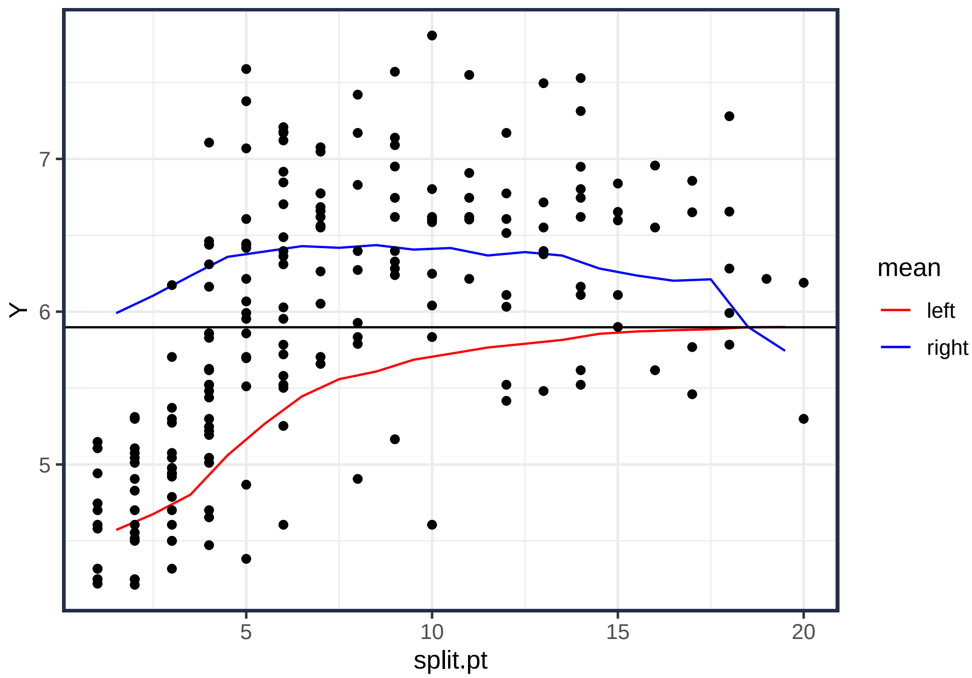
```
#> 3        3.5     47    153  4.80   6.23   8.29  79.0   87.3   73.8
#> 4        4.5     71    129  5.06   6.36 26.5    57.3   83.8   77.2
#> 5        5.5     88    112  5.27   6.39 52.7    45.7   98.4   62.6
#> 6        6.5    108     92  5.45   6.43 78.3    34.7 113.    48.1
```

```

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    71   129  5.06  6.36  26.5  57.3  83.8  77.2
```

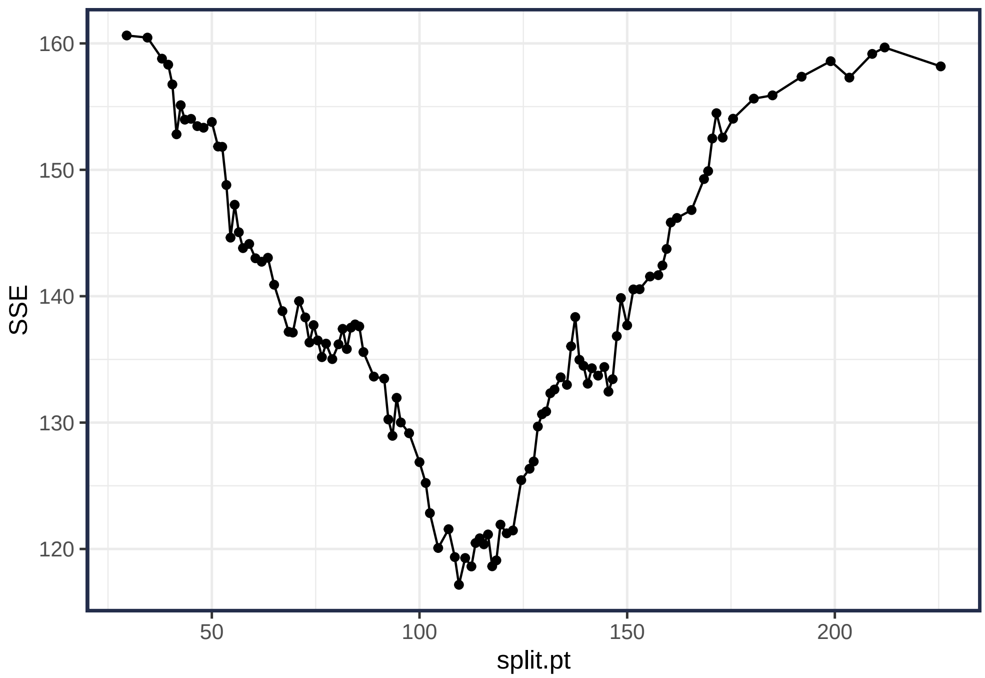
```

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
  labs(y="Y")
```



3.1.2 Split on Hits

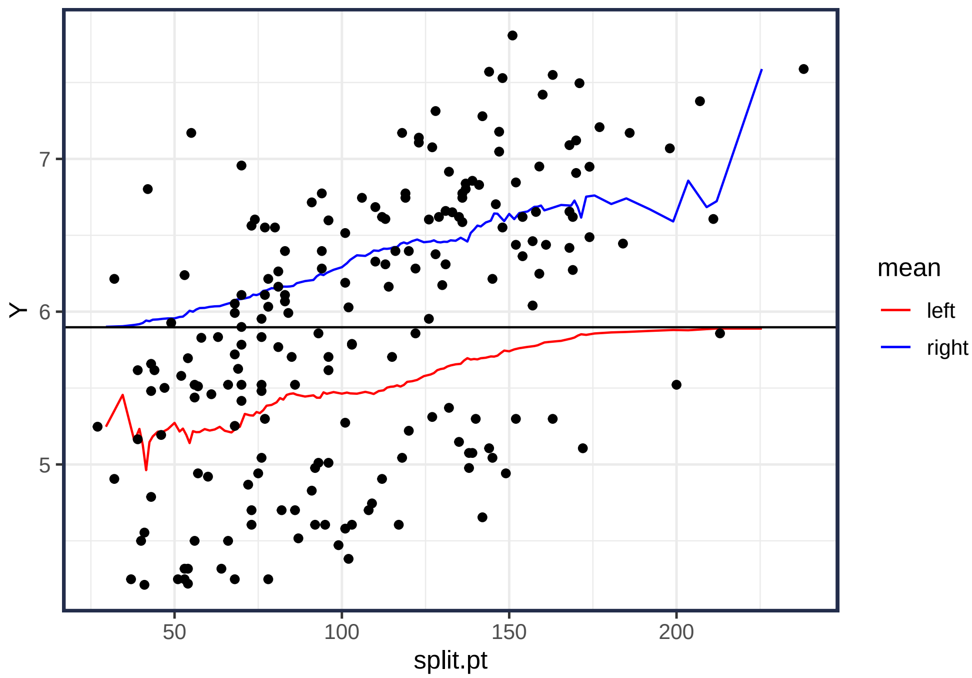
```
## Split by Hits
hits = split_info(x=bball$Hits,y=bball$Y)
head(hits)
#> # A tibble: 6 x 9
#>   split.pt   n.L   n.R est.L est.R SSE.L SSE.R   SSE  gain
#>      <dbl> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1     29.5     1   199  5.25  5.90 0       161.  161. 0.426
#> 2     34.5     3   197  5.46  5.90 0.922   160.  160. 0.596
#> 3     38       4   196  5.15  5.91 2.02    157.  159. 2.26
#> 4     39.5     6   194  5.23  5.92 2.19    156.  158. 2.74
#> 5     40.5     7   193  5.13  5.93 2.65    154.  157. 4.30
#> 6     41.5     9   191  4.96  5.94 3.57    149.  153. 8.25
```

```

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     110.   107    93  5.46  6.40  61.3  55.9  117.  43.9
```

```
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
  labs(y = "Y")
```



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] 161.1
```

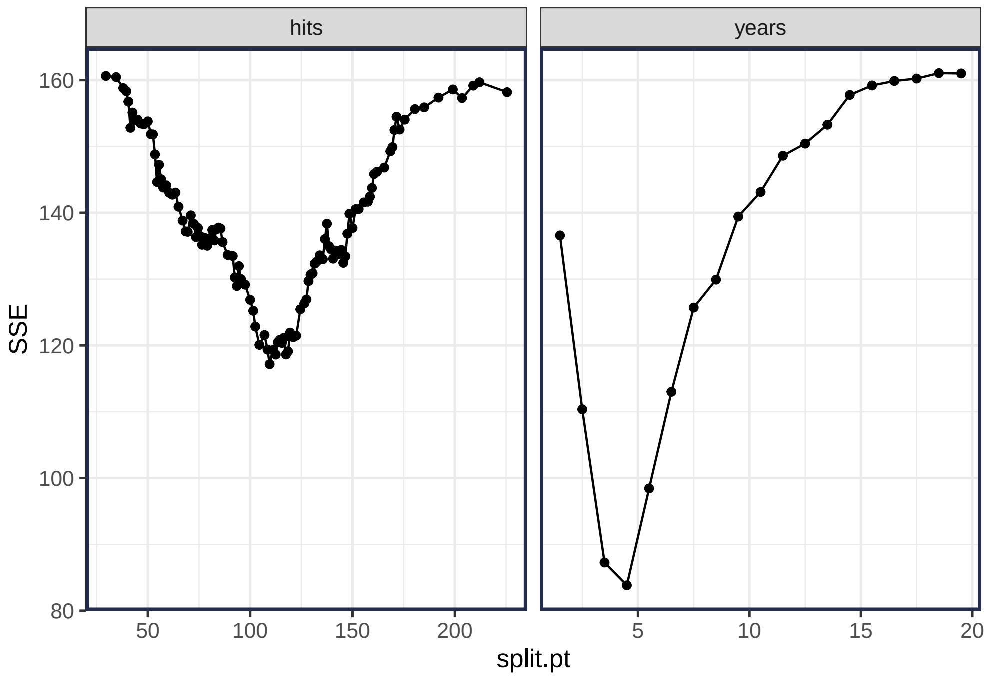
```
# (nrow(bball)-1)*var(bball$Y)



## 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] 161.1
```

```
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    71   129  5.06  6.36  26.5  57.3  83.8  77.2
```

```
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     110.   107    93  5.46  6.40  61.3  55.9  117.  43.9
```

```
split_metrics(bball$Years,bball$Y, 4.5)
#> # A tibble: 2 x 3
#>   region   SSE     n
#>   <chr>  <dbl> <int>
#> 1 LEFT    26.5    71
#> 2 RIGHT   57.3   129
```

```
## 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] 9.278
```

```
max(years2.R$gain,na.rm=TRUE)
#> [1] 1.576
```

```
max(hits2.L$gain,na.rm=TRUE)
#> [1] 8.726
```

```
max(hits2.R$gain,na.rm=TRUE)
#> [1] 24.42
```

```

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.     69    60  5.95  6.83  22.6  10.3  32.9  24.4
```

```

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



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