HW3SDS

Bailey Brady, Andrew Chen, Cristian Sigala, Cherry Sun

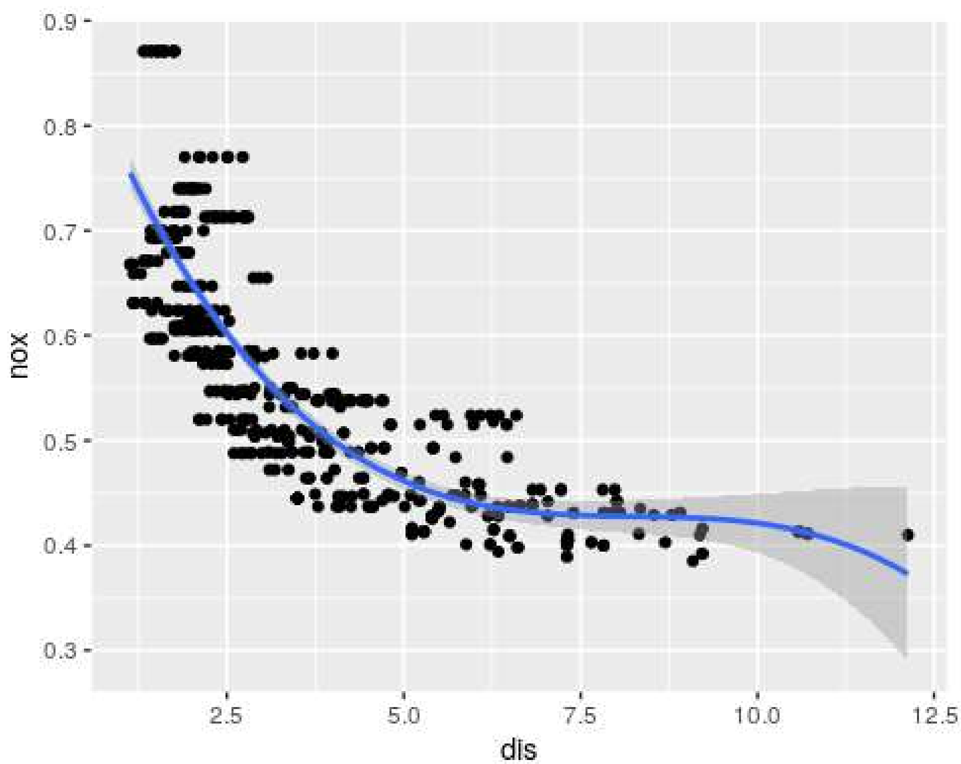
4/11/2021

Question 9 Secttion 7.9

```
rm(list = ls())

library(MASS)

##
## Attaching package: 'MASS'

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

library(boot)
library(gridExtra)

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:randomForest':
##
##     combine

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

data("Boston")
attach(Boston)

cubic <- lm(nox~poly(dis,3), data = Boston)
summary(cubic)

##
## Call:
## lm(formula = nox ~ poly(dis, 3), data = Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.121130 -0.040619 -0.009738  0.023385  0.194904
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)    0.554695   0.002759 201.021  < 2e-16 ***
## poly(dis, 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
## poly(dis, 3)2  0.856330   0.062071  13.796  < 2e-16 ***
## poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06207 on 502 degrees of freedom
## Multiple R-squared:  0.7148, Adjusted R-squared:  0.7131
## F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

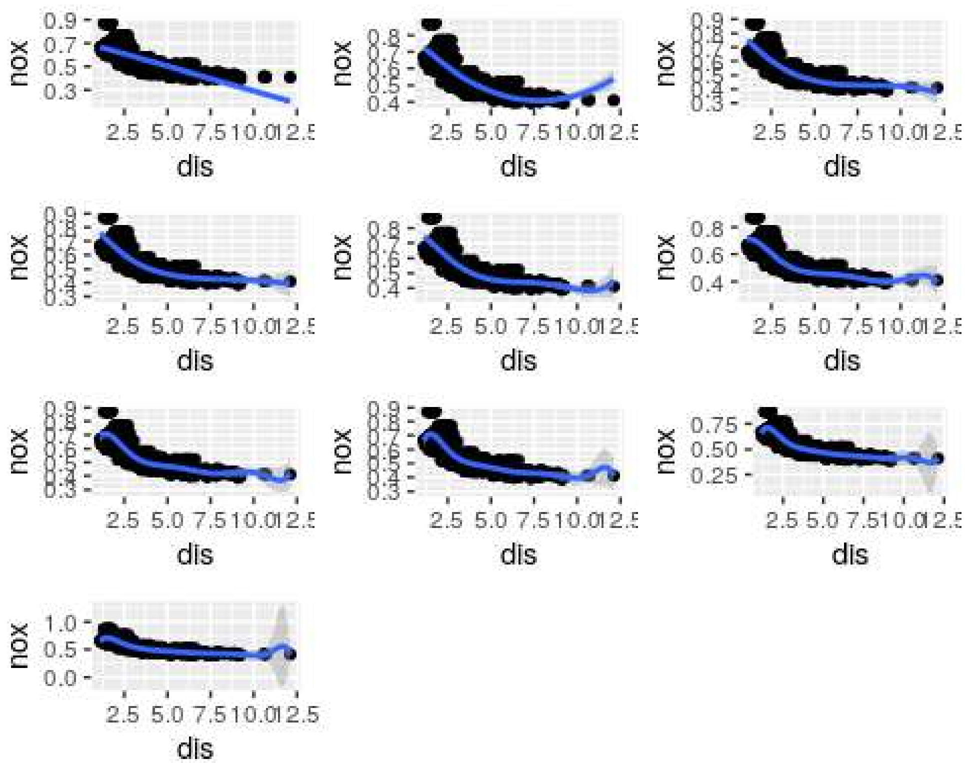
```
ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method = "lm",
formula = y~poly(x,3))
```




```
set.seed(1000)

residuallist = rep(NA,10)
for (i in 1:10) {
  cubic <- lm(nox~poly(dis,i), data = Boston)
  residuallist[i] = sum(cubic$residuals^2)
}

residuallist
```

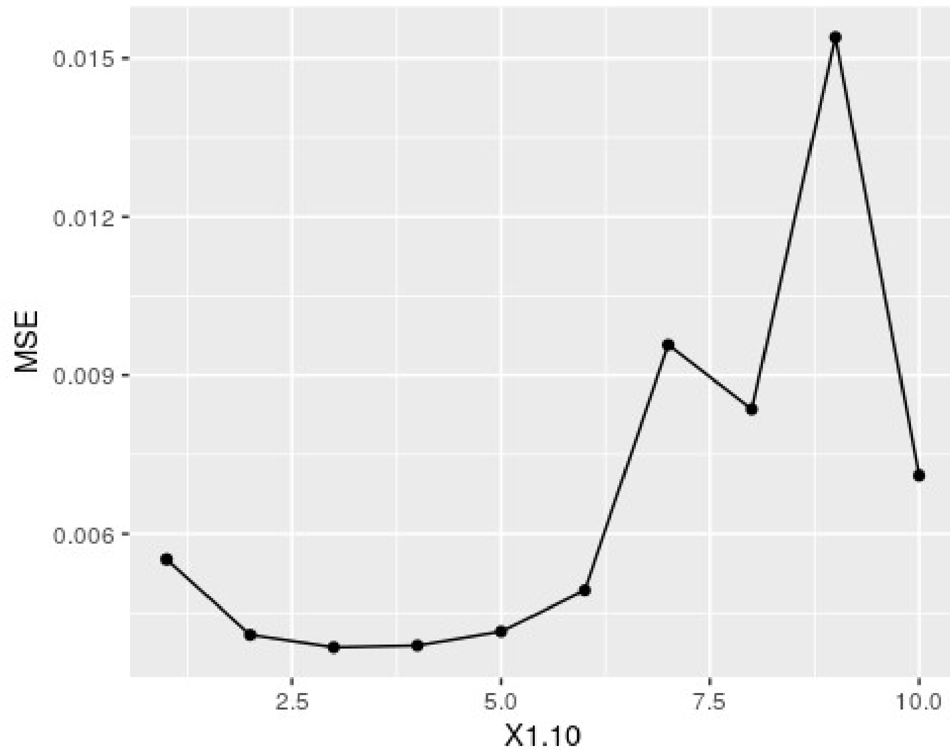
```
##  [1] 2.768563 2.035262 1.934107 1.932981 1.915290 1.878257 1.849484
1.835630
##  [9] 1.833331 1.832171
```

```r
a <- ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method = "lm",
formula = y~poly(x,1))
b <- ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method = "lm",
formula = y~poly(x,2))
c <- ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method = "lm",
formula = y~poly(x,3))
d <- ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method = "lm",
formula = y~poly(x,4))
e <-ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method = "lm",
formula = y~poly(x,5))
f<-ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method = "lm",
formula = y~poly(x,6))
g<-ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method = "lm",
formula = y~poly(x,7))
h<-ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method = "lm",
formula = y~poly(x,8))
i<-ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method = "lm",
formula = y~poly(x,9))
j<-ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method = "lm",
formula = y~poly(x,10))

grid.arrange(a,b,c,d,e,f,g,h,i,j, ncol = 3)
```




```r
set.seed(1000)
MSE <- rep(NA,10)
for (i in 1:10) {
  cubic <- glm(nox~poly(dis,i), data = Boston)
```

```
  MSE[i] <- cv.glm(Boston, cubic, K = 10)$delta[2]
}
cv.plot <- data.frame(1:10,MSE)
ggplot(cv.plot, aes(X1.10,MSE))+geom_point()+geom_line()
```




```
which.min(MSE)

## [1] 3
```

The model with degree 3 on the polynomial has the smallest MSE so it is most optimal one.

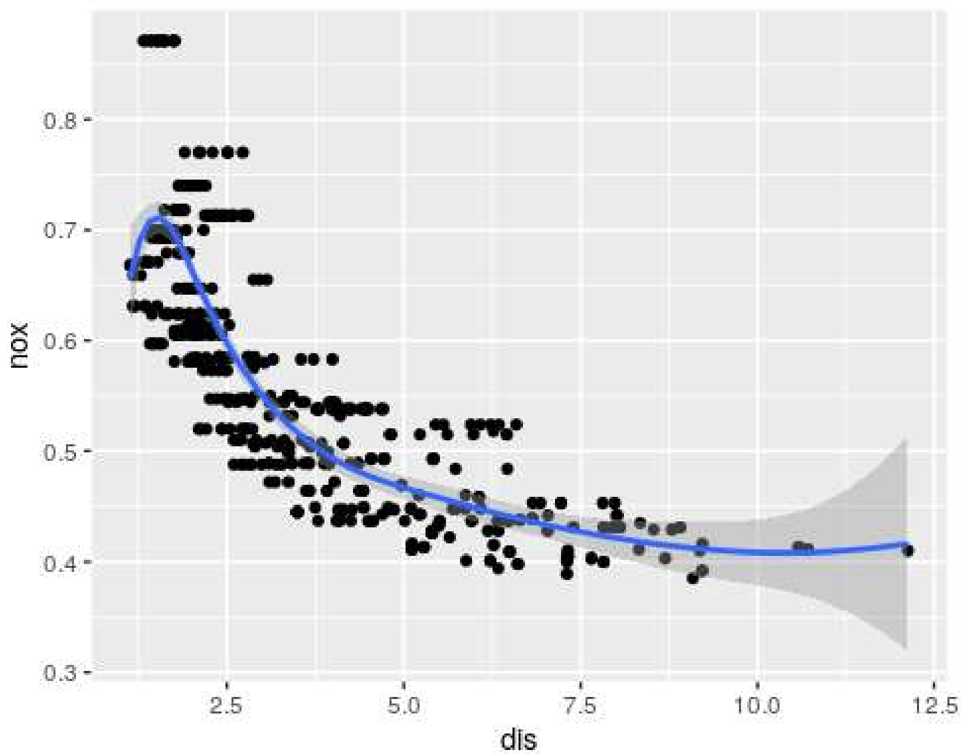
```
library(splines)
quantile(dis)

##        0%       25%       50%       75%      100%
##  1.129600  2.100175  3.207450  5.188425 12.126500
```

The 25th 50th and 75th qunatiles of dis is used for the knots

```
splinemodel <- lm(nox~bs(dis, df = 4, knots=c(2.100175,3.20745,5.188425)))
summary(splinemodel)

##
## Call:
## lm(formula = nox ~ bs(dis, df = 4, knots = c(2.100175, 3.20745,
##     5.188425)))
##
## Residuals:
```

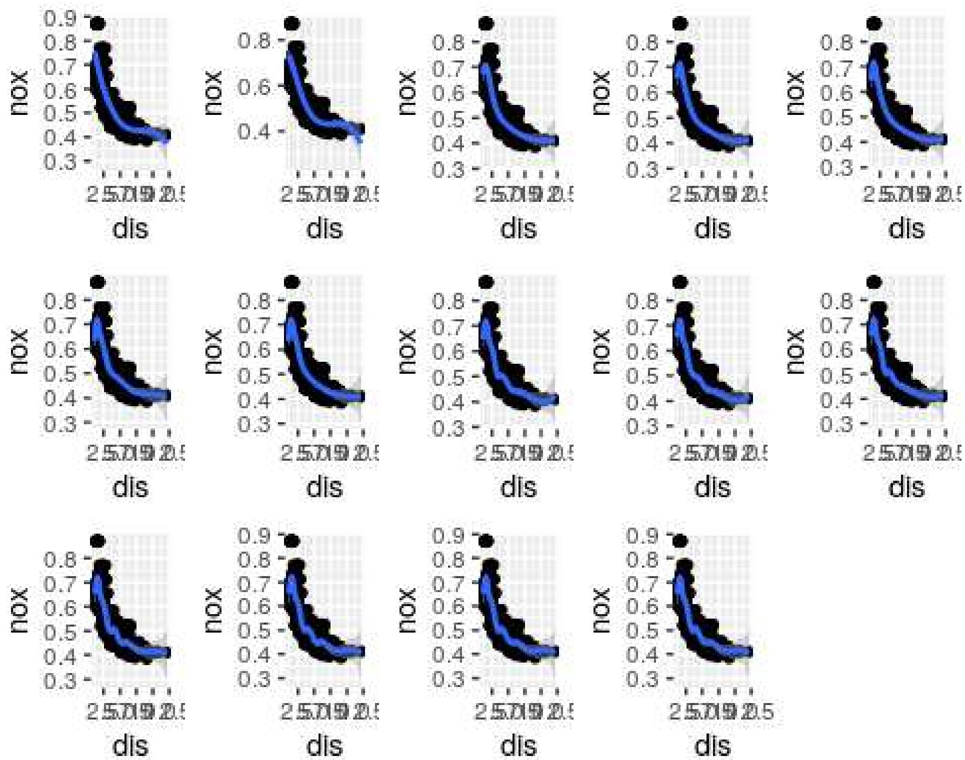
```
##       Min        1Q    Median        3Q       Max 
## -0.128538 -0.037813 -0.009987  0.022644  0.195494 
## 
## Coefficients:
##                                                    Estimate Std.
Error
## (Intercept)                                         0.65622
0.02370
## bs(dis, df = 4, knots = c(2.100175, 3.20745, 5.188425))1  0.10222
0.03516
## bs(dis, df = 4, knots = c(2.100175, 3.20745, 5.188425))2 -0.02963
0.02338
## bs(dis, df = 4, knots = c(2.100175, 3.20745, 5.188425))3 -0.15959
0.02791
## bs(dis, df = 4, knots = c(2.100175, 3.20745, 5.188425))4 -0.22815
0.03324
## bs(dis, df = 4, knots = c(2.100175, 3.20745, 5.188425))5 -0.26272
0.04930
## bs(dis, df = 4, knots = c(2.100175, 3.20745, 5.188425))6 -0.24002
0.05434
##                                                    t value Pr(>|t|)
## (Intercept)                                         27.689  < 2e-16
***
## bs(dis, df = 4, knots = c(2.100175, 3.20745, 5.188425))1   2.907  0.00381
**
## bs(dis, df = 4, knots = c(2.100175, 3.20745, 5.188425))2  -1.267  0.20571
## bs(dis, df = 4, knots = c(2.100175, 3.20745, 5.188425))3  -5.718 1.86e-08
***
## bs(dis, df = 4, knots = c(2.100175, 3.20745, 5.188425))4  -6.864 1.99e-11
***
## bs(dis, df = 4, knots = c(2.100175, 3.20745, 5.188425))5  -5.329 1.50e-07
***
## bs(dis, df = 4, knots = c(2.100175, 3.20745, 5.188425))6  -4.417 1.23e-05
***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.06062 on 499 degrees of freedom
## Multiple R-squared:  0.7295, Adjusted R-squared:  0.7263 
## F-statistic: 224.3 on 6 and 499 DF,  p-value: < 2.2e-16
```

```r
ggplot(Boston, aes(dis,nox))+geom_point()+stat_smooth(method = "lm", formula
= y ~ bs(x,4,knots=c(2.100175,3.20745,5.188425)))
```




```
set.seed(1000)
residuals = rep(NA,16)
for (i in 3:16) {
  splinemodel <- lm(nox~bs(dis, df = i),data = Boston)
  residuals[i] <- sum(splinemodel$residuals^2)
}

plot1 = ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method =
"lm", formula = y~bs(x, df = 3))
plot2 = ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method =
"lm", formula = y~bs(x, df = 4))
plot3 = ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method =
"lm", formula = y~bs(x, df = 5))
plot4 = ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method =
"lm", formula = y~bs(x, df = 6))
plot5 = ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method =
"lm", formula = y~bs(x, df = 7))
plot6 = ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method =
"lm", formula = y~bs(x, df = 8))
plot7 = ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method =
"lm", formula = y~bs(x, df = 9))
plot8 = ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method =
"lm", formula = y~bs(x, df = 10))
plot9 = ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method =
"lm", formula = y~bs(x, df = 11))
plot10 = ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method =
```

```
"lm", formula = y~bs(x, df = 12))
plot11 = ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method =
"lm", formula = y~bs(x, df = 13))
plot12 = ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method =
"lm", formula = y~bs(x, df = 14))
plot13 = ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method =
"lm", formula = y~bs(x, df = 15))
plot14 = ggplot(Boston, aes(dis,nox)) + geom_point()+ stat_smooth(method =
"lm", formula = y~bs(x, df = 16))
grid.arrange(plot1,plot2,plot3,plot4,plot5,plot6,plot7,plot8,plot9,plot10,plo
t11,plot12,plot13,plot14, ncol = 5)
```

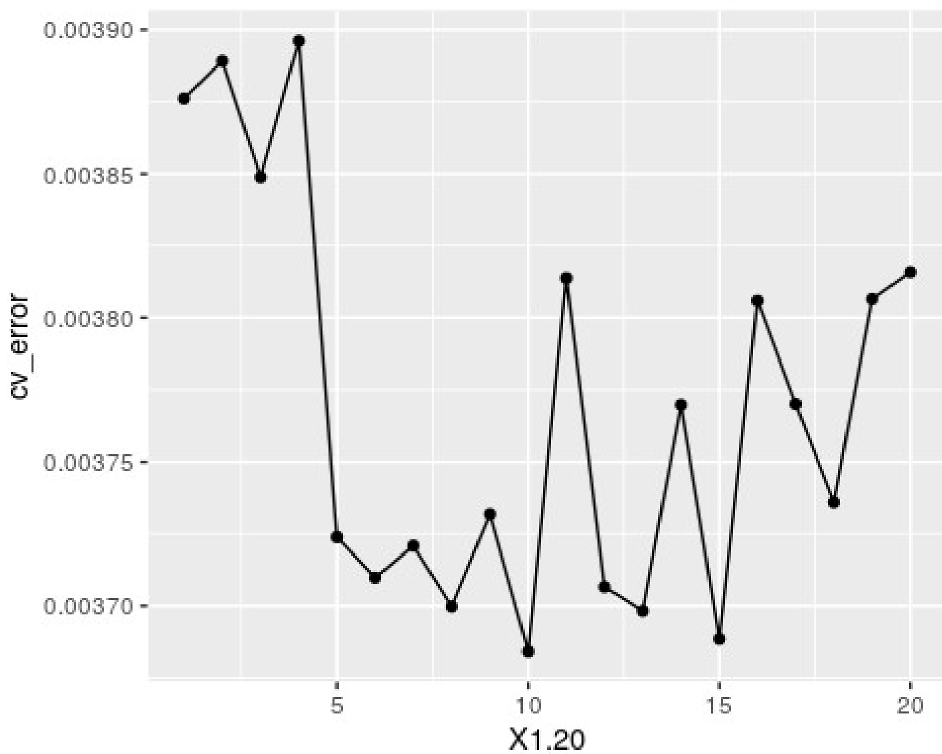


```
residuals

## [1]       NA       NA 1.934107 1.922775 1.840173 1.833966 1.829884
1.816995
## [9] 1.825653 1.792535 1.796992 1.788999 1.782350 1.781838 1.782798
1.783546
```

Th resulting RSS's are pretty close together with 1.93 at degree one being the biggest and 1.781 being the smallest at degree 14

```
set.seed(1000)
cv_error <- rep(0,20)
for (i in 1:20){
  splinemodel <- glm(nox~bs(dis, df = i), data = Boston)
```

```
  cv_error[i] <- cv.glm(Boston, splinemodel, K = 10)$delta[1]
}

cv.plot2 <- data.frame(1:20,cv_error)
cv.plot2

##    X1.20    cv_error
## 1      1 0.003876104
## 2      2 0.003889102
## 3      3 0.003848859
## 4      4 0.003896136
## 5      5 0.003723802
## 6      6 0.003709755
## 7      7 0.003720826
## 8      8 0.003699725
## 9      9 0.003731784
## 10    10 0.003684162
## 11    11 0.003813839
## 12    12 0.003706551
## 13    13 0.003698232
## 14    14 0.003769753
## 15    15 0.003688437
## 16    16 0.003805985
## 17    17 0.003770049
## 18    18 0.003735931
## 19    19 0.003806571
## 20    20 0.003815923

ggplot(cv.plot2, aes(X1.20,cv_error))+geom_point()+geom_line()
```




```r
which.min(cv_error)

## [1] 10
```

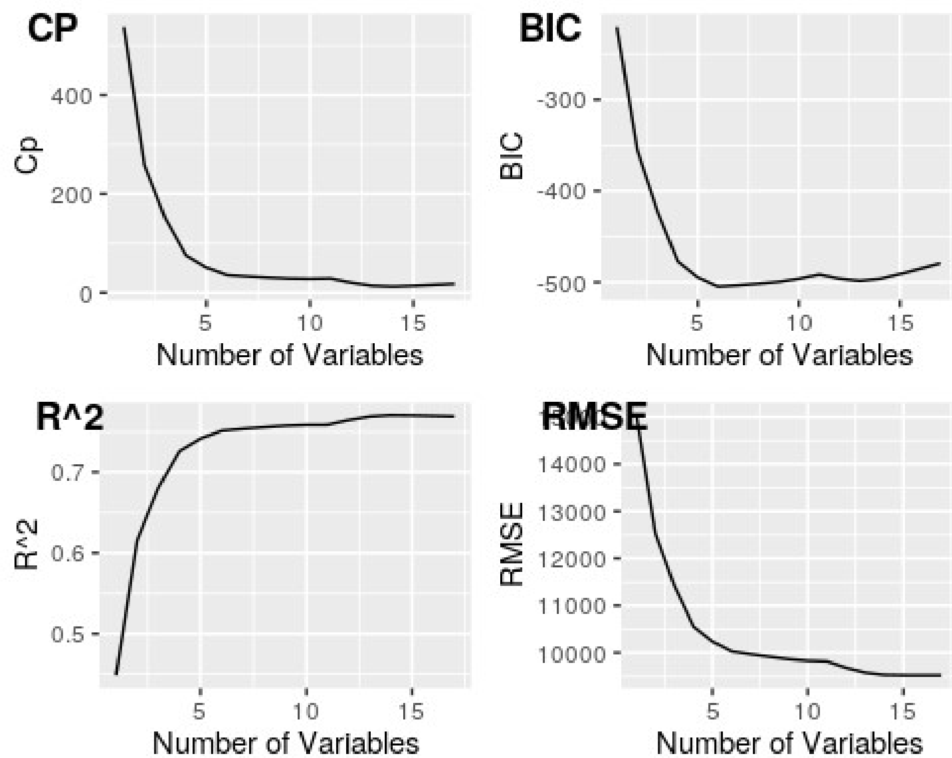
The best degree to use is 10 degrees of freedom. It has the lowest error.

```r
detach(Boston)
```

Question 10 in Section 7.9

```r
set.seed(1)
data <- College
n <- length(data$Outstate)
train <- sample(n, n/2)
trainset <- College[train, ]
testset <- College[-train, ]
lm <- regsubsets(Outstate ~ ., data = trainset, nvmax = 17, method =
"forward")
lm.summary <- summary(lm)

cp <- data.frame(cp =lm.summary$cp, y=1:17)
bic <- data.frame(bic =lm.summary$bic, y=1:17)
adjr2 <- data.frame(adjr2 =lm.summary$adjr2, y=1:17)
RMSE <- data.frame(RMSE = sqrt(lm.summary$rss/17), y=1:17)


var1 <- ggplot(cp,aes(x=y, y=cp))+xlab("Number of Variables") + ylab("Cp") +
```

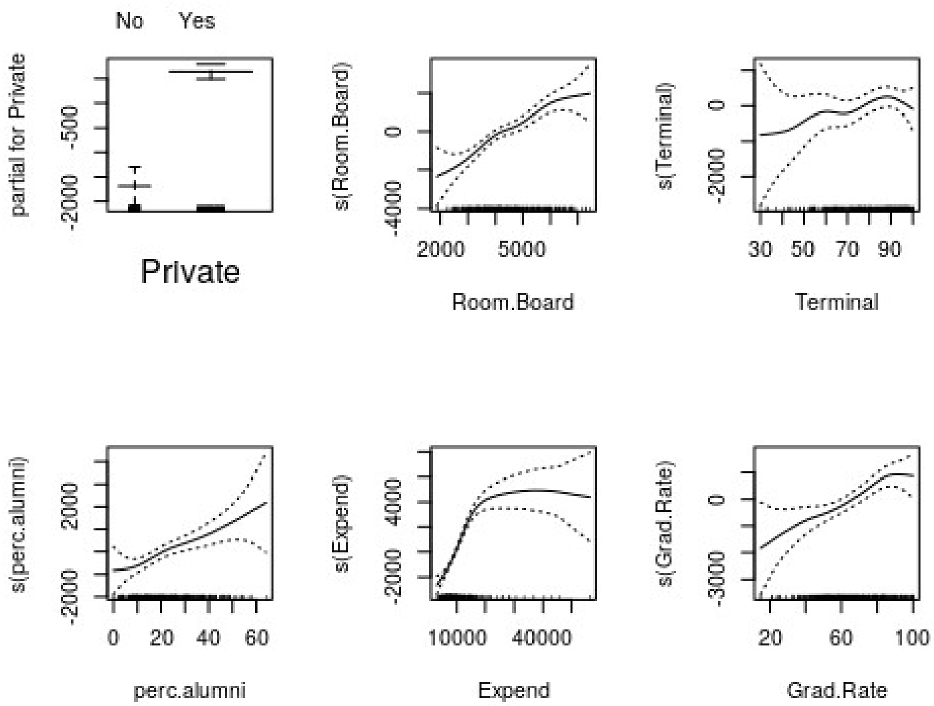
```
geom_line()
var2 <- ggplot(bic, aes(x=y, y=bic))+xlab("Number of Variables") +
ylab("BIC") + geom_line()
var3 <- ggplot(adjr2, aes(x=y, y=adjr2))+xlab("Number of Variables") +
ylab("R^2") + geom_line()
var4 <- ggplot(RMSE, aes(x=y, y=RMSE))+xlab("Number of Variables") +
ylab("RMSE") + geom_line()



figure <- ggarrange(var1, var2, var3, var4,
                    labels = c("CP", "BIC", "R^2", "RMSE"),
                    ncol = 2, nrow = 2)

figure
```




```
which.min(lm.summary$bic)

## [1] 6

coef <- coef(lm, id = 6)
names(coef)

## [1] "(Intercept)" "PrivateYes"   "Room.Board"   "Terminal"     "perc.alumni"
## [6] "Expend"      "Grad.Rate"
```

While looking at the different plots, we can tell that the BIC model is the best to fit the data because it has the least variance with the lowest amount of variables affecting the fit. After conducting coef tests on the BIC model, we find the variables are “Private”, “Room.Board”, “Terminal”, “perc.alumni”, “Expend”, and “Grad.Rate” that interact with the “outstate” variable.

```
gamlm <- gam(Outstate ~ Private + s(Room.Board) + s(Terminal) +
    s(perc.alumni) + s(Expend) + s(Grad.Rate), data = trainset)

par(mfrow = c(2, 3))
plot(gamlm, se = T)
```



After plotting our gam model, we can see the data is more on a exponential type of curve rather than linear. All the variables are able to explain out of state tuition but Room.Board has the least variance and the straightest line making the relationship almost direct between the two. Terminal and perc.alumni on the other hand has large variance at the ends so although the model fits, it is not the best variable to find a relationship with out of state tuition.

```
Predi <- predict(gamlm, testset)
RSS <- sum((Predi-testset$Outstate)^2)
TSS <- sum((testset$Outstate - mean(testset$Outstate))^2)
RS2_Test <- 1- (RSS/TSS)

RS2_Test

## [1] 0.7656864
```

```r
Predi2 <- predict(gamlm, trainset)
RSS <- sum((Predi2-trainset$Outstate)^2)
TSS <- sum((trainset$Outstate - mean(trainset$Outstate))^2)
RS2_Train <- 1- (RSS/TSS)

RS2_Train

## [1] 0.8049801
```

After calculating the r^2 value of the train and test set, we find out that both are good fits for the data. However, when comparing the two we find the train set has a greater fit than the test which is expected.

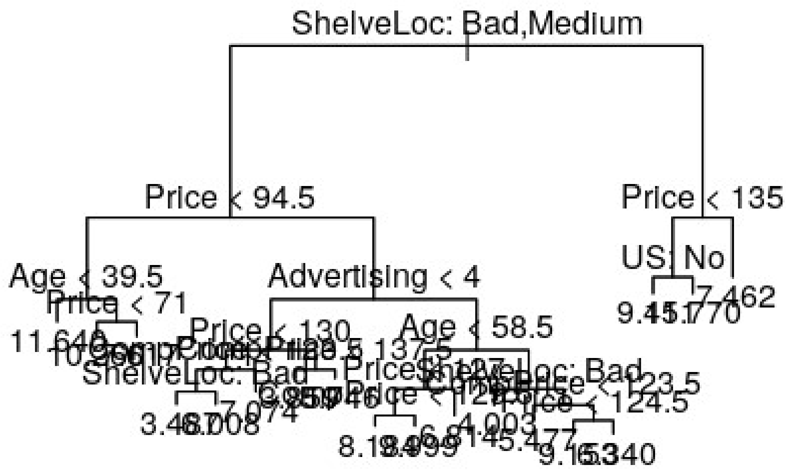
```r
summary(gamlm)

##
## Call: gam(formula = Outstate ~ Private + s(Room.Board) + s(Terminal) +
##     s(perc.alumni) + s(Expend) + s(Grad.Rate), data = trainset)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -7128.62 -1133.86   -74.25  1231.50  7369.50
##
## (Dispersion Parameter for gaussian family taken to be 3724586)
##
##     Null Deviance: 6989966760 on 387 degrees of freedom
## Residual Deviance: 1363197370 on 365.9997 degrees of freedom
## AIC: 6995.069
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                 Df     Sum Sq    Mean Sq F value    Pr(>F)
## Private          1 1764398916 1764398916 473.717 < 2.2e-16 ***
## s(Room.Board)    1 1616561254 1616561254 434.024 < 2.2e-16 ***
## s(Terminal)      1  287918343  287918343  77.302 < 2.2e-16 ***
## s(perc.alumni)   1  354690429  354690429  95.230 < 2.2e-16 ***
## s(Expend)        1  601731164  601731164 161.556 < 2.2e-16 ***
## s(Grad.Rate)     1   90312393   90312393  24.248 1.284e-06 ***
## Residuals      366 1363197370    3724586
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                Npar Df  Npar F     Pr(F)
## (Intercept)
## Private
## s(Room.Board)        3  1.9107    0.1274
## s(Terminal)          3  1.4636    0.2241
## s(perc.alumni)       3  0.3498    0.7893
## s(Expend)            3 26.1184 2.442e-15 ***
```

```
## s(Grad.Rate)         3  0.9075    0.4375
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

When looking at our anova nonperametric effects, we see that only the expend variable is significant in explaining out of state tuition with a nonlinear relationship. Although the rest of the variables have some effect it is not enough to be significant.

Section 8.4 Question 8

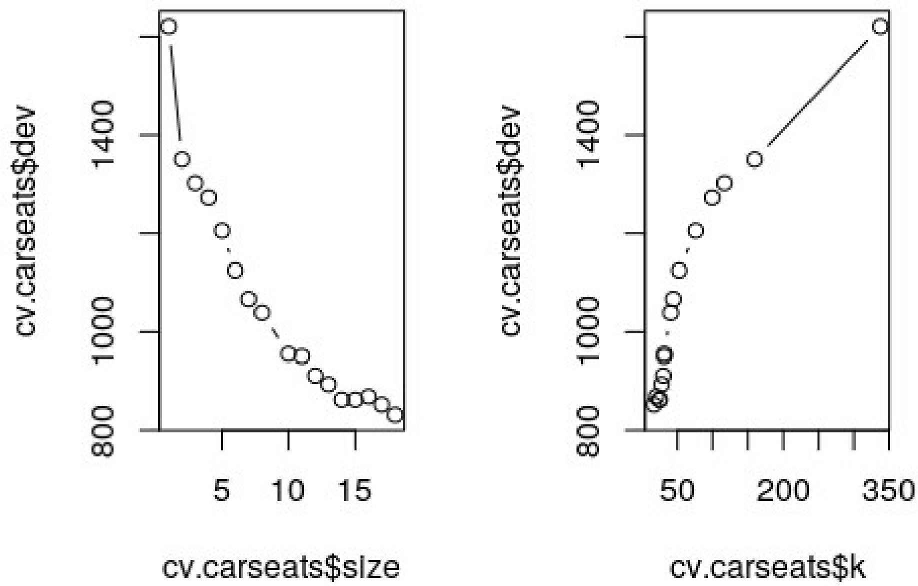
```
set.seed(1)
train = sample(1:nrow(Carseats), nrow(Carseats)/2)
Cartrain <- Carseats[train, ]
Cartest <- Carseats[-train, ]

tree.carseats <- tree(Sales ~ ., data = Cartrain)
summary(tree.carseats)

##
## Regression tree:
## tree(formula = Sales ~ ., data = Cartrain)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Advertising" "CompPrice"
## [6] "US"
## Number of terminal nodes:  18
## Residual mean deviance:  2.167 = 394.3 / 182
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.88200 -0.88200 -0.08712  0.00000  0.89590  4.09900

plot(tree.carseats)
text(tree.carseats, pretty = 0)
```



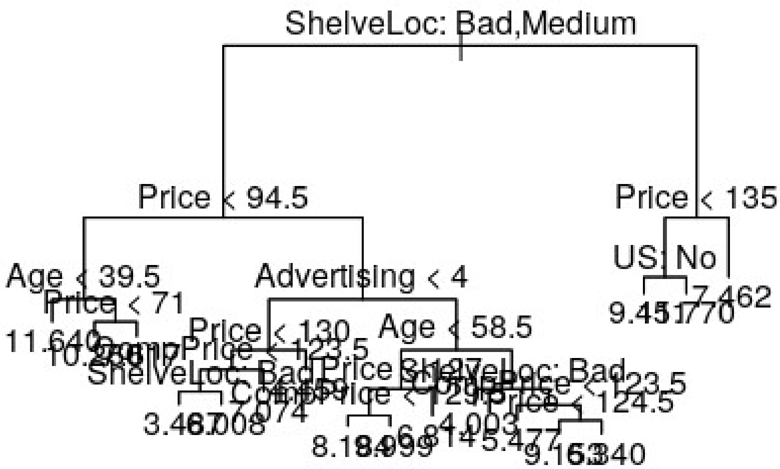

```r
predicty <- predict(tree.carseats, newdata = Cartest)
mean((predicty - Cartest$Sales)^2)

## [1] 4.922039
```

We get a MSE of 4.922.

```r
cv.carseats = cv.tree(tree.carseats)

par(mfrow = c(1,2))
plot(cv.carseats$size, cv.carseats$dev, type="b")
plot(cv.carseats$k, cv.carseats$dev, type="b")
```




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

prune.carseats <- prune.tree(tree.carseats, best = 17)
plot(prune.carseats)
text(prune.carseats, pretty = 0)
```



```r
predicty <- predict(prune.carseats, newdata = Cartest)
mean((predicty - Cartest$Sales)^2)

## [1] 4.827162
```

We get a MSE of 4.827. In this case pruning the tree improved the MSE by about 0.1.

```r
bagcarseats <- randomForest(Sales ~ ., data = Cartrain, mtry = 10, importance
= TRUE)
y.bag <- predict(bagcarseats, newdata = Cartest)
mean((y.bag - Cartest$Sales)^2)

## [1] 2.657296

importance(bagcarseats)

##                %IncMSE IncNodePurity
## CompPrice   23.07909904    171.185734
## Income       2.82081527     94.079825
## Advertising 11.43295625     99.098941
## Population  -3.92119532     59.818905
## Price       54.24314632    505.887016
## ShelveLoc   46.26912996    361.962753
## Age         14.24992212    159.740422
## Education   -0.07662320     46.738585
## Urban        0.08530119      8.453749
## US           4.34349223     15.157608
```

The MSE found is 2.62.

We see that Price and ShelveLoc are the most important and followed by CompPrice.

```
rfcarseats <- randomForest(Sales ~ ., data = Cartrain, mtry = 3, importance =
TRUE)
y.rf <- predict(rfcarseats, newdata = Cartest)
mean((y.rf - Cartest$Sales)^2)

## [1] 3.049406

importance(rfcarseats)

##                %IncMSE IncNodePurity
## CompPrice   12.9489323     158.48521
## Income       2.2754686     129.59400
## Advertising  8.9977589     111.94374
## Population  -2.2513981     102.84599
## Price       33.4226950     391.60804
## ShelveLoc   34.0233545     290.56502
## Age         12.2185108     171.83302
## Education    0.2592124      71.65413
## Urban        1.1382113      14.76798
## US           4.1925335      33.75554
```

The MSE found is 3.00.

We see that Price and ShelveLoc are the most important again and CompPrice and Age are followed after.

Question 10 in Section 8.4

```
Hitters <- Hitters
Hitters = na.omit(Hitters)
Hitters$Salary = log(Hitters$Salary)
Hitters$League <- as.factor(Hitters$League)
Hitters$Division <- as.factor(Hitters$Division)
Hitters$NewLeague <- as.factor(Hitters$NewLeague)
head(Hitters)

##                    AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits
CHmRun
## -Alan Ashby          315   81     7   24  38    39    14   3449   835
69
## -Alvin Davis         479  130    18   66  72    76     3   1624   457
63
## -Andre Dawson        496  141    20   65  78    37    11   5628  1575
225
## -Andres Galarraga    321   87    10   39  42    30     2    396   101
12
## -Alfredo Griffin     594  169     4   74  51    35    11   4408  1133
```

```
19
## -Al Newman          185   37     1   23   8    21     2    214    42
1
##                    CRuns CRBI CWalks League Division PutOuts Assists Errors
## -Alan Ashby          321  414    375      N        W     632      43     10
## -Alvin Davis         224  266    263      A        W     880      82     14
## -Andre Dawson        828  838    354      N        E     200      11      3
## -Andres Galarraga     48   46     33      N        E     805      40      4
## -Alfredo Griffin     501  336    194      A        W     282     421     25
## -Al Newman            30    9     24      N        E      76     127      7
##                      Salary NewLeague
## -Alan Ashby        6.163315         N
## -Alvin Davis       6.173786         A
## -Andre Dawson      6.214608         N
## -Andres Galarraga  4.516339         N
## -Alfredo Griffin   6.620073         A
## -Al Newman         4.248495         A
```

```
train_set <- Hitters[1:200,]
dim(train_set)
```

```
## [1] 200  20
```

```
head(train_set)
```

```
##                    AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits
CHmRun
## -Alan Ashby          315   81     7   24  38    39    14   3449   835
69
## -Alvin Davis         479  130    18   66  72    76     3   1624   457
63
## -Andre Dawson        496  141    20   65  78    37    11   5628  1575
225
## -Andres Galarraga    321   87    10   39  42    30     2    396   101
12
## -Alfredo Griffin     594  169     4   74  51    35    11   4408  1133
19
## -Al Newman           185   37     1   23   8    21     2    214    42
1
##                    CRuns CRBI CWalks League Division PutOuts Assists Errors
## -Alan Ashby          321  414    375      N        W     632      43     10
## -Alvin Davis         224  266    263      A        W     880      82     14
## -Andre Dawson        828  838    354      N        E     200      11      3
## -Andres Galarraga     48   46     33      N        E     805      40      4
## -Alfredo Griffin     501  336    194      A        W     282     421     25
## -Al Newman            30    9     24      N        E      76     127      7
##                      Salary NewLeague
## -Alan Ashby        6.163315         N
## -Alvin Davis       6.173786         A
## -Andre Dawson      6.214608         N
## -Andres Galarraga  4.516339         N
```

```
## -Alfredo Griffin  6.620073         A
## -Al Newman        4.248495         A
```

```
test_set <- Hitters[-(1:200),]
dim(test_set)
```

```
## [1] 63 20
```

```
head(test_set)
```

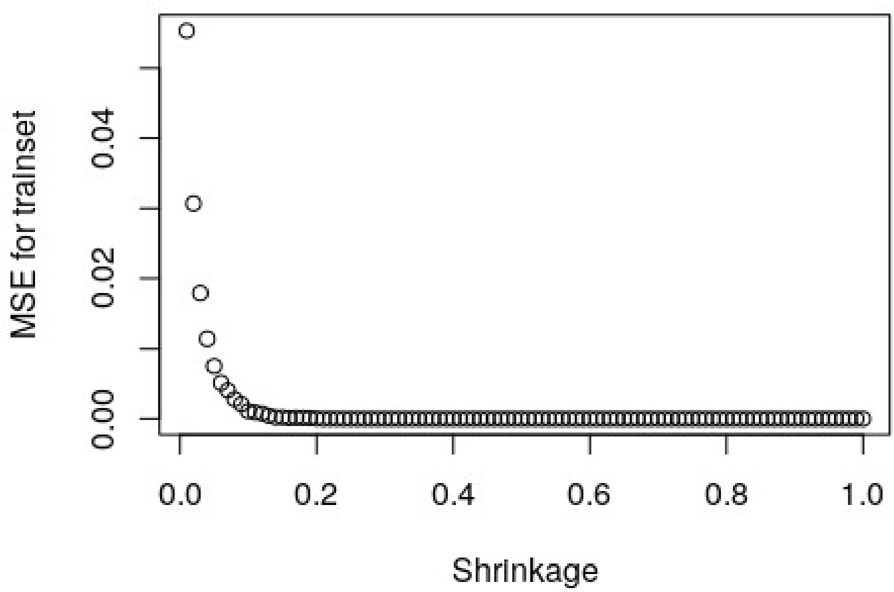
```
##                    AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun
## -Reggie Jackson      419  101    18   65  58    92    20   9528  2510    548
## -Ron Kittle          376   82    21   42  60    35     5   1770   408    115
## -Ray Knight          486  145    11   51  76    40    11   3967  1102     67
## -Rick Leach          246   76     5   35  39    13     6    912   234     12
## -Rick Manning        205   52     8   31  27    17    12   5134  1323     56
## -Rance Mulliniks     348   90    11   50  45    43    10   2288   614     43
##                    CRuns CRBI CWalks League Division PutOuts Assists Errors
## -Reggie Jackson     1509 1659   1342      A        W       0       0      0
## -Ron Kittle          238  299    157      A        W       0       0      0
## -Ray Knight          410  497    284      N        E      88     204     16
## -Rick Leach          102   96     80      A        E      44       0      1
## -Rick Manning        643  445    459      A        E     155       3      2
## -Rance Mulliniks     295  273    269      A        E      60     176      6
##                      Salary NewLeague
## -Reggie Jackson    6.189290         A
## -Ron Kittle        6.052089         A
## -Ray Knight        6.214608         A
## -Rick Leach        5.521461         A
## -Rick Manning      5.991465         A
## -Rance Mulliniks 6.109248         A
```

shows 200, which means that training set consist of first 200 observations shows 63, the rest of data are assigned to test set

```
set.seed(100)
pows <- seq(0.01, 1, by = 0.01)
range <- pows
train_error <- rep(0, length(range))
for (i in 1: length(range)){
  my_boost <- gbm(Salary~., data = train_set, distribution = "gaussian",
n.trees= 1000, interaction.depth = 5, shrinkage = range[i])
  pred <- predict(my_boost, newdata = train_set, n.trees = 1000)
  train_error[i] <- MSE(pred, train_set$Salary)
}
train_error
```

```
##   [1] 5.531229e-02 3.070116e-02 1.793041e-02 1.141311e-02 7.515446e-03
##   [6] 5.135847e-03 4.089155e-03 2.829888e-03 2.107375e-03 1.090970e-03
##  [11] 9.143186e-04 7.527424e-04 4.438250e-04 2.107033e-04 2.084422e-04
##  [16] 1.275705e-04 1.564862e-04 1.481893e-04 1.122995e-04 4.508664e-05
```

```
##  [21] 6.669702e-06 1.632212e-05 1.089263e-05 1.868700e-06 4.964248e-06
##  [26] 1.141003e-06 6.565694e-07 2.394964e-06 4.514636e-08 3.121725e-07
##  [31] 6.639143e-08 2.024619e-08 4.573450e-07 3.784828e-09 2.050641e-08
##  [36] 5.068387e-08 4.006767e-09 2.628265e-09 1.003464e-09 3.465420e-09
##  [41] 5.803472e-09 4.233554e-09 6.757971e-10 1.205403e-08 1.816530e-11
##  [46] 1.157308e-09 5.552622e-11 4.077847e-09 1.506577e-10 2.791922e-10
##  [51] 8.232659e-12 1.485731e-10 6.286405e-12 2.730091e-12 1.224742e-12
##  [56] 1.205243e-12 2.151287e-13 2.859646e-12 1.600019e-13 2.264890e-13
##  [61] 2.532185e-14 2.721992e-12 4.726114e-14 1.251975e-16 1.276339e-14
##  [66] 1.387579e-15 5.746383e-14 5.599344e-17 5.966583e-17 1.391327e-15
##  [71] 6.460855e-16 3.188759e-15 1.395992e-16 7.268948e-16 5.488075e-17
##  [76] 1.568662e-17 6.096367e-18 1.258339e-17 1.650647e-16 1.253426e-17
##  [81] 9.068014e-19 5.372179e-16 2.473670e-18 3.337819e-18 1.435766e-18
##  [86] 3.410540e-18 2.484255e-17 2.448975e-17 1.391628e-17 1.586695e-16
##  [91] 3.882708e-17 2.305812e-17 8.095739e-18 1.055980e-16 3.387161e-16
##  [96] 3.070941e-16 1.862369e-16 1.169657e-15 7.427585e-15 3.361751e-14

plot(range, train_error, xlab = "Shrinkage", ylab = "MSE for trainset")
```



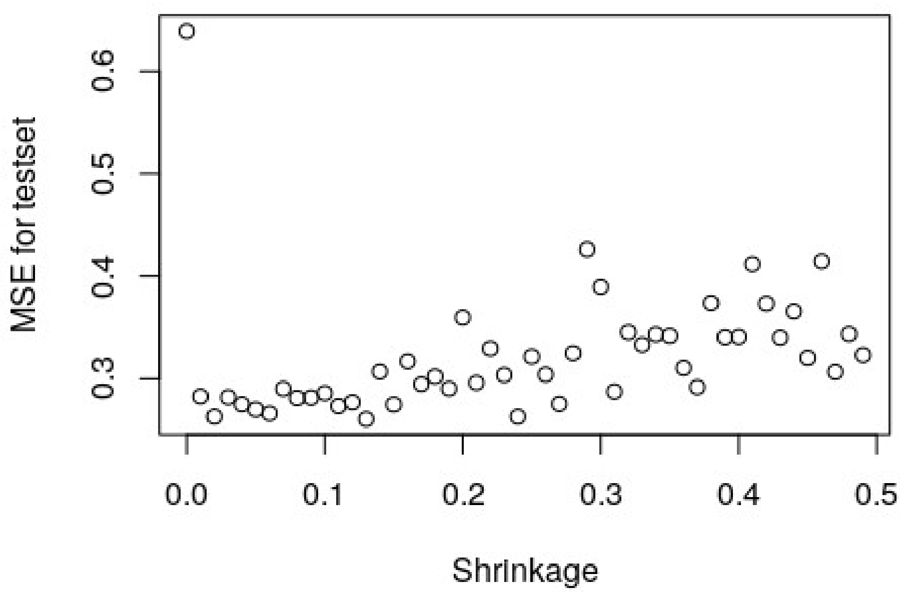
we are seeing this behavior on the graph because we don’t have enough number of trees, we should be using a larger number for trees

```
set.seed(100)
pows <- seq(0.00001, 0.5, by = 0.01)
range <- pows
train_error <- rep(0, length(range))
for (i in 1: length(range)){
  my_boost <- gbm(Salary~., data = train_set, distribution = "gaussian",
```

```r
n.trees= 1000, interaction.depth = 4, shrinkage = range[i])
  pred <- predict(my_boost, newdata = test_set, n.trees = 1000)
  train_error[i] <- MSE(pred, test_set$Salary)
}
train_error
```

```
##  [1] 0.6390523 0.2825796 0.2629287 0.2818142 0.2749321 0.2699292 0.2660277
##  [8] 0.2900506 0.2810656 0.2814218 0.2857974 0.2732245 0.2768972 0.2605141
## [15] 0.3068412 0.2747142 0.3168417 0.2945322 0.3020256 0.2902090 0.3595950
## [22] 0.2958914 0.3293535 0.3035305 0.2628832 0.3214829 0.3039472 0.2753221
## [29] 0.3247932 0.4260514 0.3894280 0.2873221 0.3452304 0.3330359 0.3431202
## [36] 0.3417009 0.3103887 0.2916457 0.3737698 0.3403540 0.3406864 0.4117370
## [43] 0.3731936 0.3399068 0.3656035 0.3202130 0.4144441 0.3065400 0.3435098
## [50] 0.3229746
```

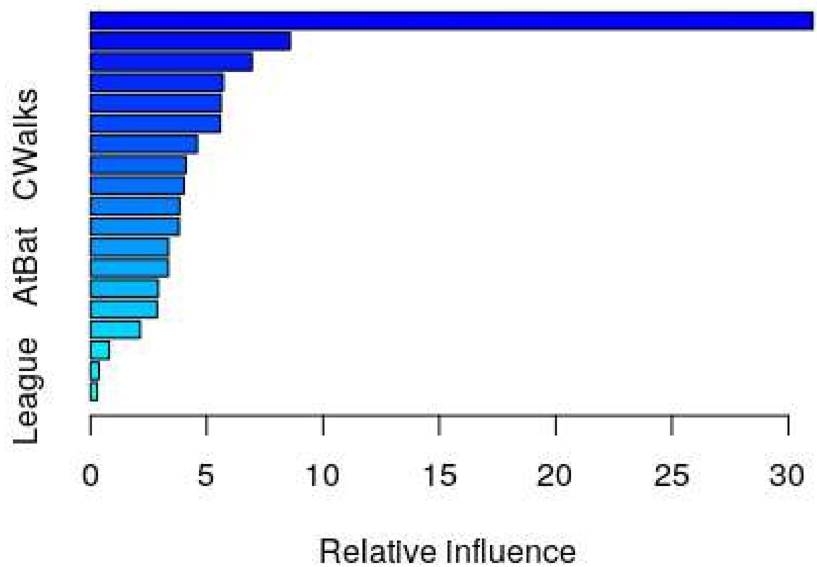
```r
plot(range, train_error, xlab = "Shrinkage", ylab = "MSE for testset")
```




```r
fit = lm(Salary ~ ., data = train_set)
pred = predict(fit, test_set)
error <- MSE(pred, test_set$Salary)
error
```

```
## [1] 0.4917959
```

```r
min_test_error <- min(error)
my_boost <- gbm(Salary~., data = train_set, distribution = "gaussian",
n.trees= 1000, interaction.depth = 4, shrinkage = min_test_error)
summary(my_boost)
```




```
##                 var    rel.inf
## CAtBat       CAtBat 31.0630258
## CRBI           CRBI  8.5887495
## PutOuts     PutOuts  6.9515029
## Assists     Assists  5.7110593
## Runs           Runs  5.6274573
## Walks         Walks  5.5930638
## CWalks       CWalks  4.5922161
## Years         Years  4.1059768
## CRuns         CRuns  4.0195358
## Hits           Hits  3.8284619
## CHmRun       CHmRun  3.8056729
## HmRun         HmRun  3.3645642
## AtBat         AtBat  3.3568675
## CHits         CHits  2.9192027
## RBI             RBI  2.8748625
## Errors       Errors  2.1324180
## Division   Division  0.8089888
## NewLeague NewLeague  0.3674487
## League       League  0.2889254
```

```
train <- model.matrix(Salary~ ., data=train_set)
test <- model.matrix(Salary~., data= test_set)
fit <- glmnet(train, train_set$Salary, alpha=1)
pred <- predict(fit,s=0.1,test)
error <- MSE(pred, test_set$Salary)
error
```

```
## [1] 0.4389054
```

CatBat seems to be the most important predictors in the boosted model the MSE is lower compare to these regression methods.

```
set.seed(100)
my_bagging <- randomForest(Salary ~ ., data = train_set, mtry = 10)
pred <- predict(my_bagging, newdata = test_set)
test_error <- MSE(pred, test_set$Salary)
test_error

## [1] 0.2232362
```

This is the test MSE bagging compare to regressions gives an lower MSE