lab5

shichenh

10/2/2017

Confidence Interval

```
reg <- lm(mpg~disp + hp, data = mtcars)

summary(reg)
```

```
##
## Call:
## lm(formula = mpg ~ disp + hp, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7945 -2.3036 -0.8246  1.8582  6.9363
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 30.735904   1.331566  23.083  < 2e-16 ***
## disp        -0.030346   0.007405  -4.098 0.000306 ***
## hp          -0.024840   0.013385  -1.856 0.073679 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.127 on 29 degrees of freedom
## Multiple R-squared:  0.7482, Adjusted R-squared:  0.7309
## F-statistic: 43.09 on 2 and 29 DF,  p-value: 2.062e-09
```

```
#confidence interval
coefs <- reg$coefficients
a <- 0.95
#question ,why minus three
tscores <- qt(c((1-a)/2, a +(1-a)/2), df = nrow(mtcars)-3)
sum <- summary(reg)
coefs[1] +  sum$coefficients[1,2] * tscores
```

```
## [1] 28.01255 33.45926
```

```
coefs[2] + sum$coefficients[2,2] * tscores
```

```
## [1] -0.04549091 -0.01520165
```

```
coefs[3] + sum$coefficients[3,2] * tscores
```

```
## [1] -0.052216500  0.002536338
```

```
confint(reg)
```

```
##                   2.5 %       97.5 %
## (Intercept) 28.01254573 33.459262767
## disp        -0.04549091 -0.015201645
## hp          -0.05221650  0.002536338
```

Hypothesis Testing

1. The choice of c is 0

```
(30.7359-0)/1.331566
```

```
## [1] 23.08252
```

```
(-0.030345-0)/0.007405
```

```
## [1] -4.097907
```

```
(-0.024840-0)/0.013385
```

```
## [1] -1.855809
```

2. The alternative hypothesis is two-sided because Pr(>|t|), so t could be either postive or negative. *** means p value smaller than 10^3, and . means p value greater or equal to 5%. On the inference side, *** strongly statistical significant to reject the hypothesis, and '.' suggests we are not sure if we have to reject the null hypothesis.

3. Base on the result($p=0.0003 < 0.05$) so we reject the null hypothesis in favor of the alternative hypothesis.

4.

```
t1 <- (coefs[2] - (-0.05))/sum$coefficients[2,2]
p1 <- 1 - pt(t1, df=nrow(mtcars-3))
p1
```

```
##        disp
## 0.006140126
```

Because $p1 < 0.05$, we reject the null hypothesis and accept the alternative hypothesis.

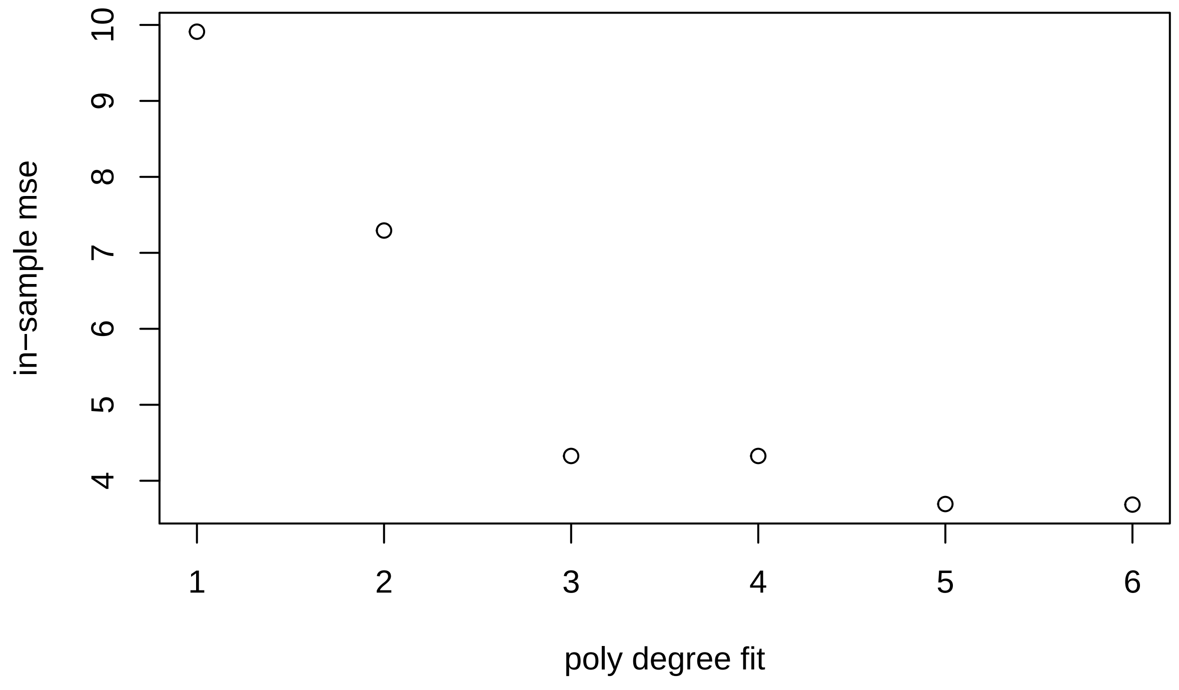
Assessment of model predictive power

Modeling with polynomial regressions

```r
attach(mtcars)
```

```
## The following object is masked from package:ggplot2:
##
##     mpg
```

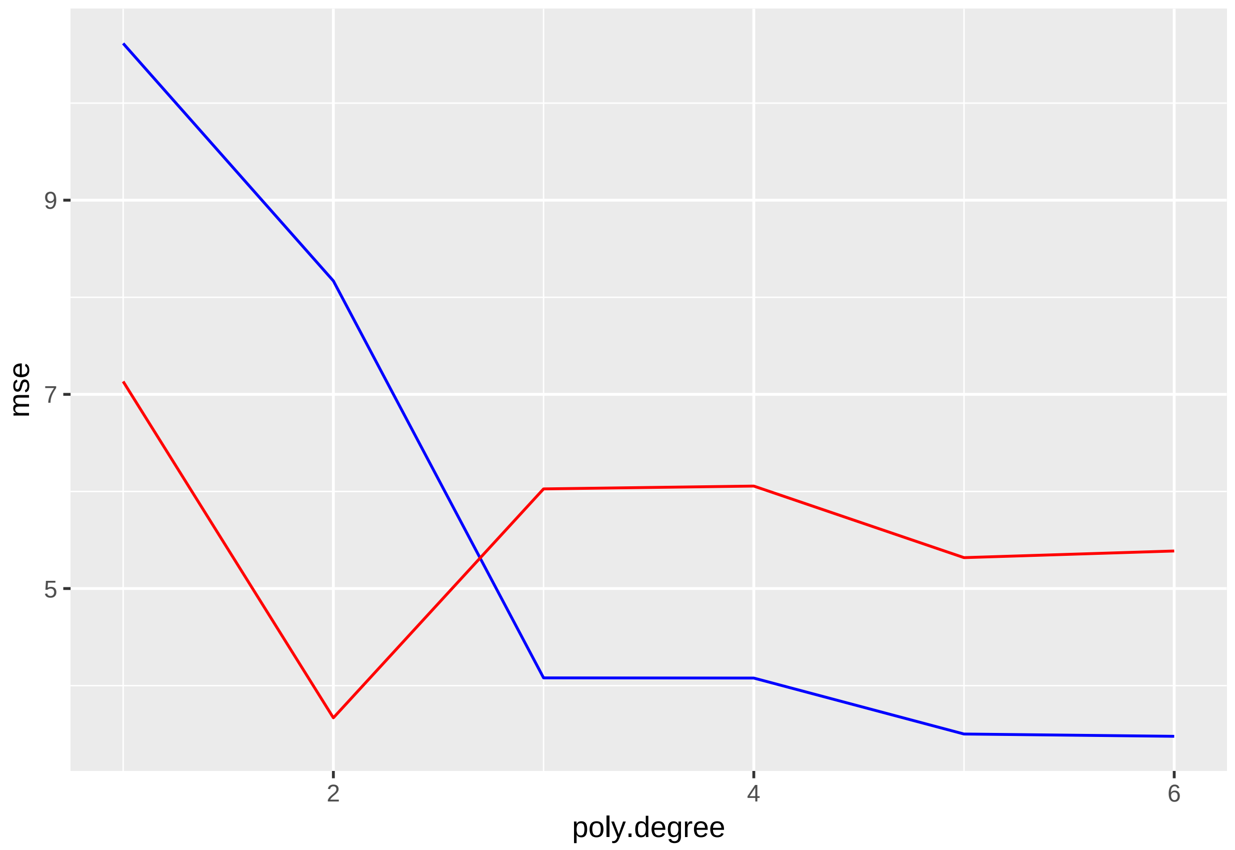
```r
poly1 <- lm(mpg~poly(disp, 1, raw=T))
poly2 <- lm(mpg~poly(disp, 2, raw=T))
poly3 <- lm(mpg~poly(disp, 3, raw=T))
poly4 <- lm(mpg~poly(disp, 4, raw=T))
poly5 <- lm(mpg~poly(disp, 5, raw=T))
poly6 <- lm(mpg~poly(disp, 6, raw=T))

mse <- c(
poly1$residuals %*% poly1$residuals / nrow(mtcars),
poly2$residuals %*% poly2$residuals / nrow(mtcars),
poly3$residuals %*% poly3$residuals / nrow(mtcars),
poly4$residuals %*% poly4$residuals / nrow(mtcars),
poly5$residuals %*% poly5$residuals / nrow(mtcars),
poly6$residuals %*% poly6$residuals / nrow(mtcars)
)

plot(1:6, mse, xlab = "poly degree fit", ylab = "in-sample mse")
```



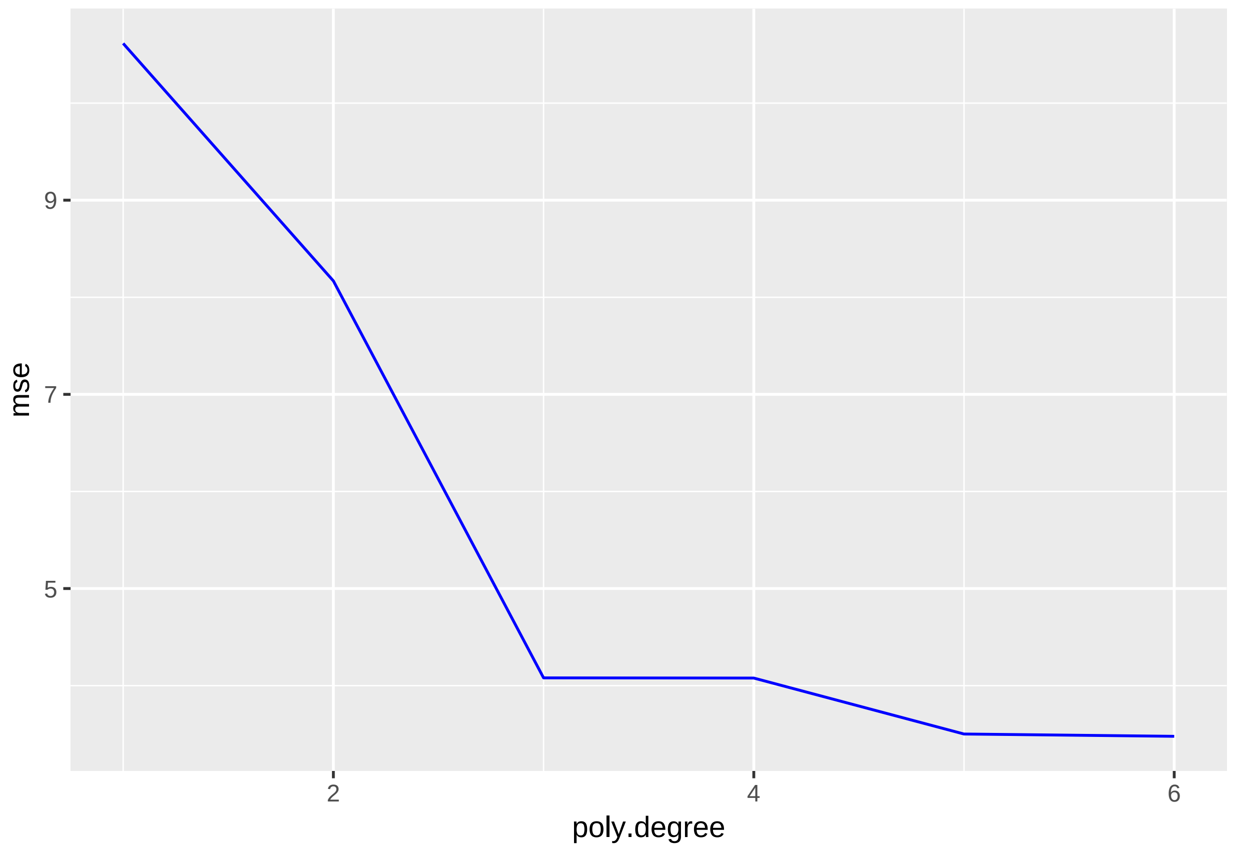
Polynomial with degree 5 has smallest mse. The in-sample mse decreases(seemingly exponentially) as the degree increases.

Holdout set

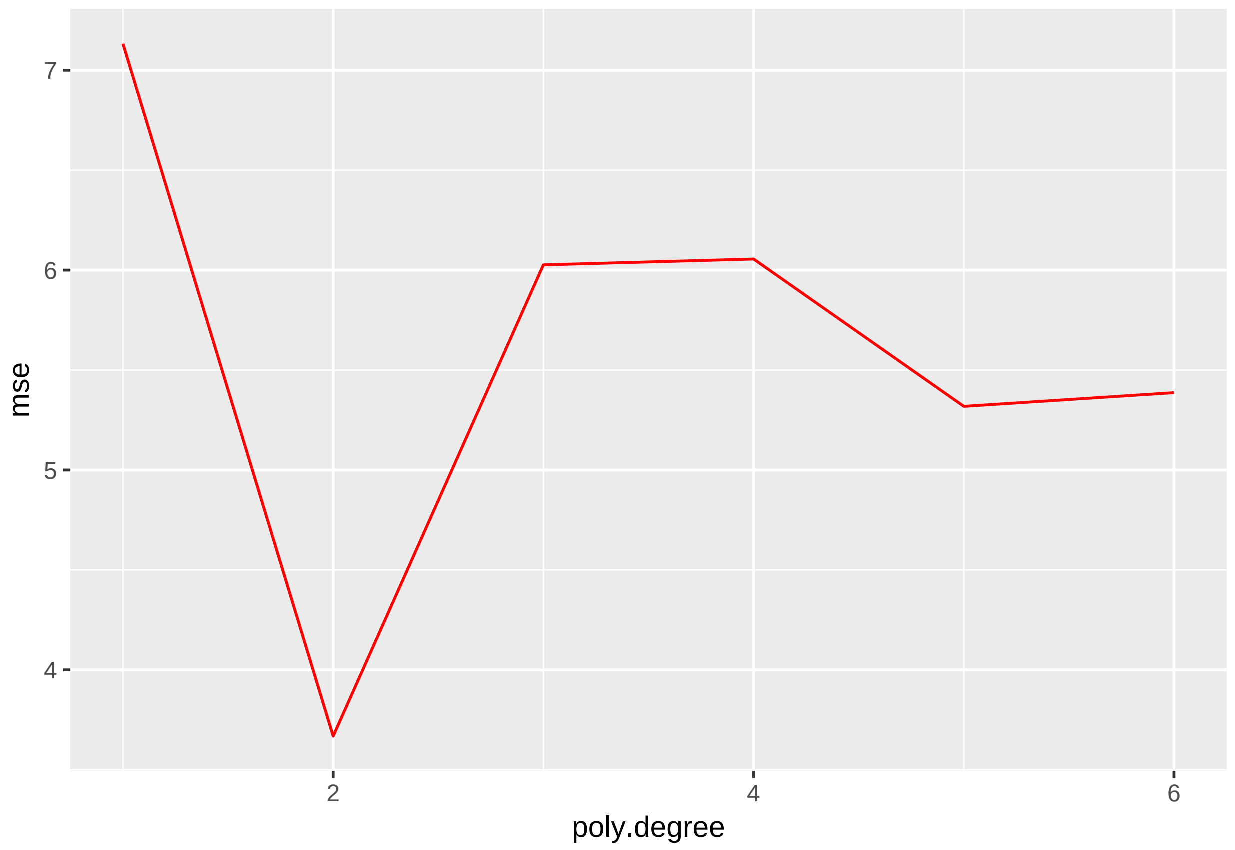
```r
fits.df <-data.frame(poly.degree = 1:6)
for (i in 1:1) {
  holdout.length <- floor(nrow(mtcars) * 0.2)
  holdout.index <- sample(seq_len(nrow(mtcars)), holdout.length)
  holdout <- mtcars[holdout.index,]
  train <- mtcars[-holdout.index,]
  performance <- data.frame()
  for (i in 1:6) {
    reg <- lm(mpg~poly(disp, i, raw=T), data = train)
    train.mse <- reg$residuals %*% reg$residuals / nrow(train)
    Xt = cbind(rep(1, nrow(holdout)), poly(holdout$disp, i, raw=T))
    holdout.resi = (Xt %*% as.matrix(reg$coefficients)) - holdout$mpg
    holdout.mse = (holdout.resi[,1] %*% holdout.resi[,1]) / nrow(holdout)
    performance <- rbind(performance, c(i, train.mse, holdout.mse))
  }
  colnames(performance) <- c("poly.degree", "train.mse", "holdout.mse")
}

ggplot(performance, aes(x=poly.degree)) +
  geom_line(aes(y=train.mse), color = "blue") +
  geom_line(aes(y=holdout.mse), color = "red") +
  ylab("mse")
```



```
ggplot(performance, aes(x=poly.degree)) +
  geom_line(aes(y=train.mse), color = "blue") +
  ylab("mse")
```



```
ggplot(performance, aes(x=poly.degree)) +
  geom_line(aes(y=holdout.mse), color = "red") +
  ylab("mse")
```



The third degree regression gives least holdout mse.(however the answer is not always consistent)

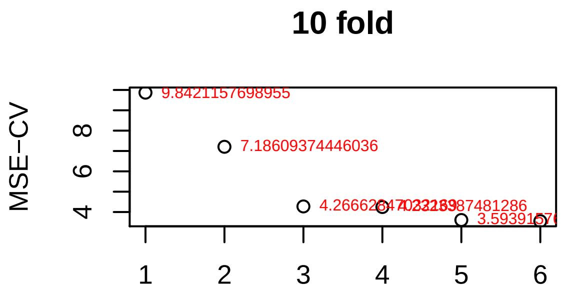
Cross-validation

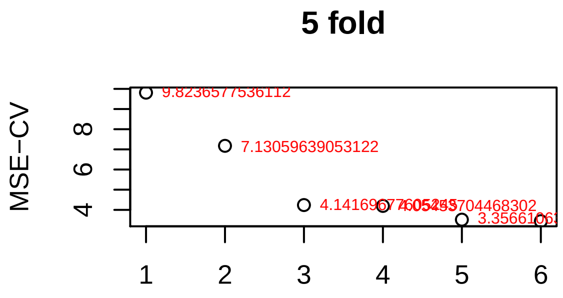
```
library(caret)
```

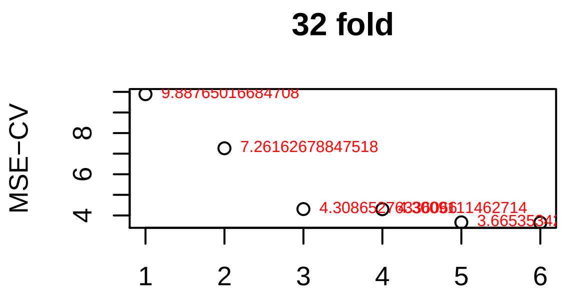
```
## Loading required package: lattice
```

```
crossval <- function(df, n) {
  performance.cross <- matrix(-1, nrow = 6, ncol = n)
  folds <- createFolds(mtcars$mpg, n)
  for (deg in 1:6) {
    for (fold in 1:length(folds)) {
      data = mtcars[-folds[[fold]],]
      temp.reg = lm(mpg~poly(disp, deg, raw=T), data = data)
      performance.cross[deg, fold] = temp.reg$residuals %*% temp.reg$residuals / nrow(da
    }
  }
  return(performance.cross)
}
```

```r
par(mfrow=c(2, 2))
plot(1:6, apply(crossval(mtcars, 10), 1, mean),
     xlab = "", ylab = "MSE-CV", main = "10 fold")
text(1:6, apply(crossval(mtcars, 10), 1, mean),
     apply(crossval(mtcars, 10), 1, mean),
     cex=0.6, pos=4, col="red")

plot(1:6, apply(crossval(mtcars, 5), 1, mean),
     xlab = "", ylab = "MSE-CV", main = "5 fold")
text(1:6, apply(crossval(mtcars, 5), 1, mean),
     apply(crossval(mtcars, 5), 1, mean),
     cex=0.6, pos=4, col="red")

plot(1:6, apply(crossval(mtcars, 32), 1, mean),
     xlab = "", ylab = "MSE-CV", main = "32 fold")
text(1:6, apply(crossval(mtcars, 32), 1, mean),
     apply(crossval(mtcars, 32), 1, mean),
     cex=0.6, pos=4, col="red")
```







2. deg5, or 6 polynomial regression generally has the best mse because it tends to overfit the data(even though we have less data for each fold calculation).

3. The CV-MSE stays pretty close but not the same because each time the data used to train the same degree polynomial is different. But in general the data shares similar linear relationship.

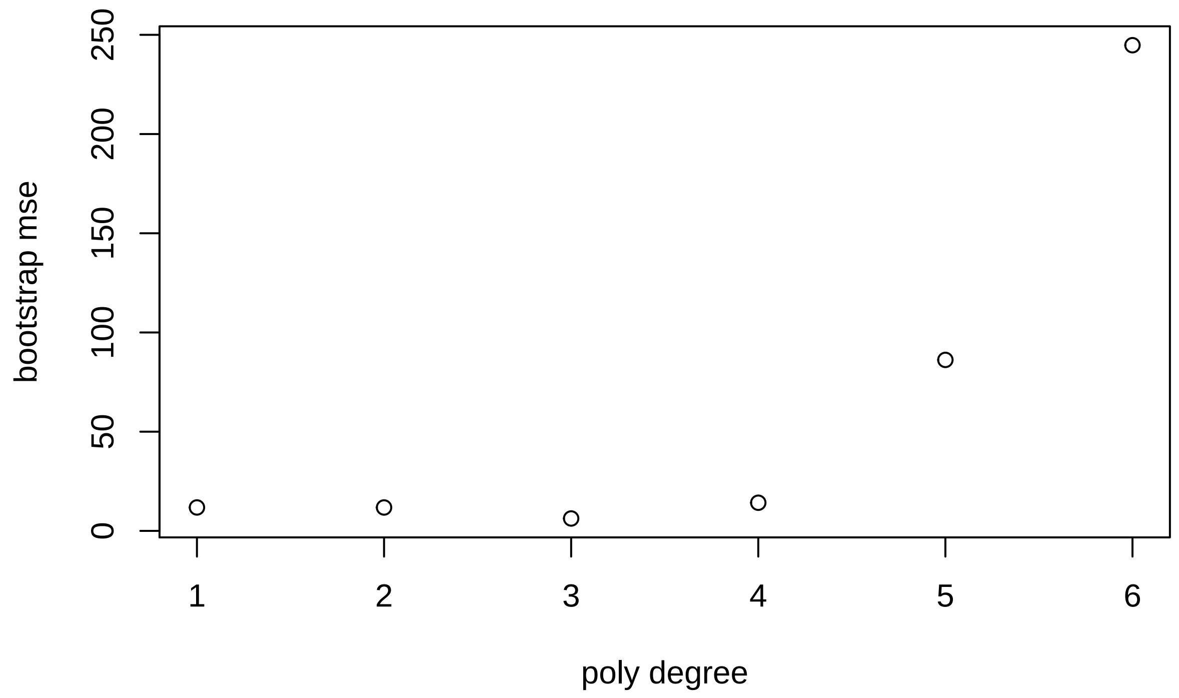
Bootstrap

```
n = 400
performance.boot <- matrix(-1, nrow = 6, ncol = 400)
for (deg in 1:6) {
  for (i in 1:400) {
    sample.index = unique(sample(1:nrow(mtcars), replace = T))
    data = mtcars[sample.index, ]
    temp.reg = lm(mpg~poly(disp, deg, raw=T), data = data)
    testset = mtcars[-sample.index, ]
    Xt = cbind(rep(1, nrow(testset)), poly(testset$disp, deg, raw=T))
    temp.resi = (Xt %*% as.matrix(temp.reg$coefficients)) - testset$mpg
    temp.mse = (temp.resi[,1] %*% temp.resi[,1]) / nrow(testset)
    performance.boot[deg, i] = temp.mse
  }
}
performance.boot[,1:10]
```

```
##           [,1]      [,2]      [,3]      [,4]       [,5]      [,6]
## [1,] 14.453404 12.325395 13.993282 12.784149 10.5224007 17.194515
## [2,]  8.602283  7.819256 11.353205 12.034674  9.2299048 11.008638
## [3,]  7.273908  6.330811  5.175878  3.899586  4.8159731  7.633965
## [4,]  8.158457  5.282809  9.391180  8.376320  2.8239472  8.895572
## [5,]  4.760756  5.268281  7.354303  3.881098  0.6888431  6.520072
## [6,]  9.469232  7.988034  4.203406  4.636219  3.7001456  9.539632
##           [,7]     [,8]      [,9]     [,10]
## [1,]  9.793218 7.094885  5.433007 12.639600
## [2,]  9.908031 7.212774  9.453250 10.482115
## [3,]  5.924934 7.744674  4.640567  7.555410
## [4,]  4.658886 5.131570  5.639248  6.014793
## [5,]  6.587143 6.894735  3.687407  8.645701
## [6,] 10.214360 5.656446 27.719838  6.715342
```

2.

```
plot(1:6, apply(performance.boot, 1, mean),
      xlab = "poly degree", ylab = "bootstrap mse")
```

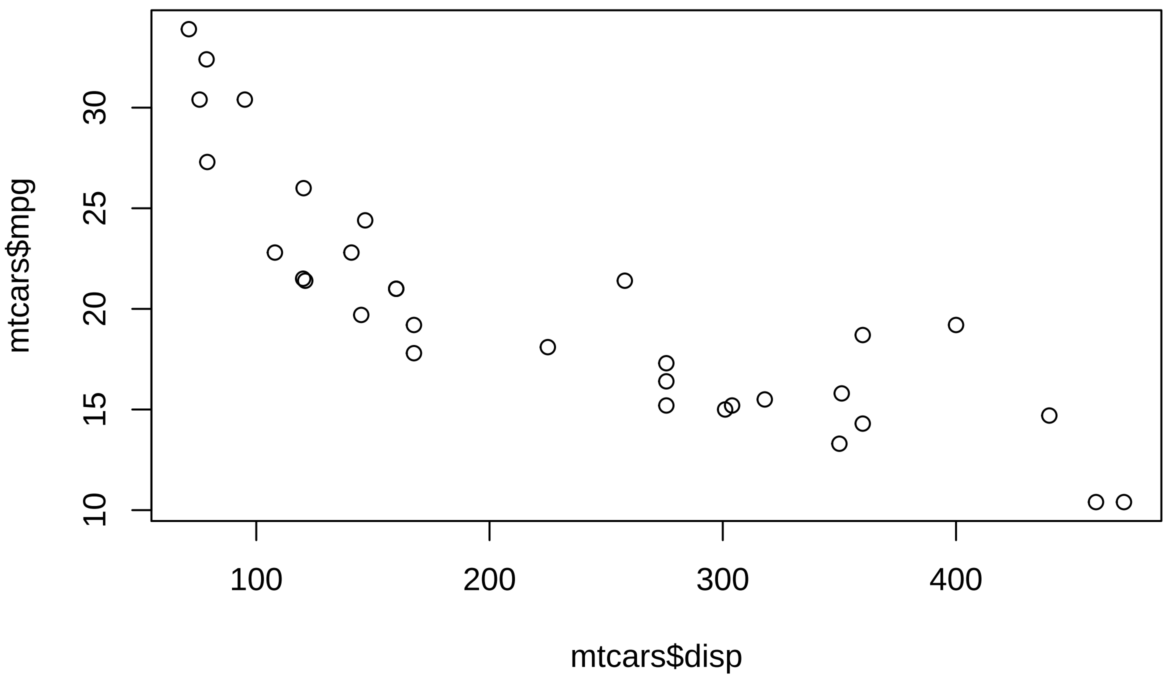


```
which.min(apply(performance.boot, 1, mean))
```

```
## [1] 3
```

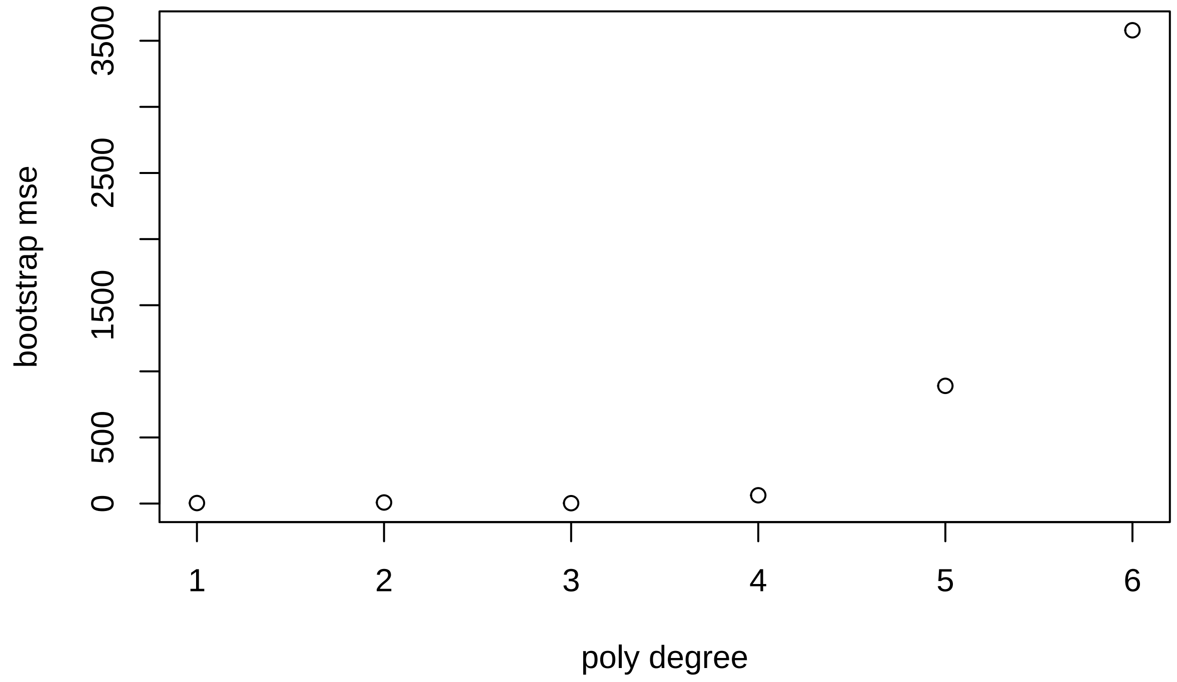
The third degree polynomail regression gives the lowest boostrap mse. It is reasonable since, look at the graph below, the plot resembles are higher degree polynomial function. However, the if the order is too higher it will overfit the training data and increases variance of test data.

```
plot(mtcars$disp, mtcars$mpg)
```



3.

```r
plot(1:6, apply(performance.boot, 1, sd),
     xlab = "poly degree", ylab = "bootstrap mse")
```



```r
apply(performance.boot, 1, sd)
```

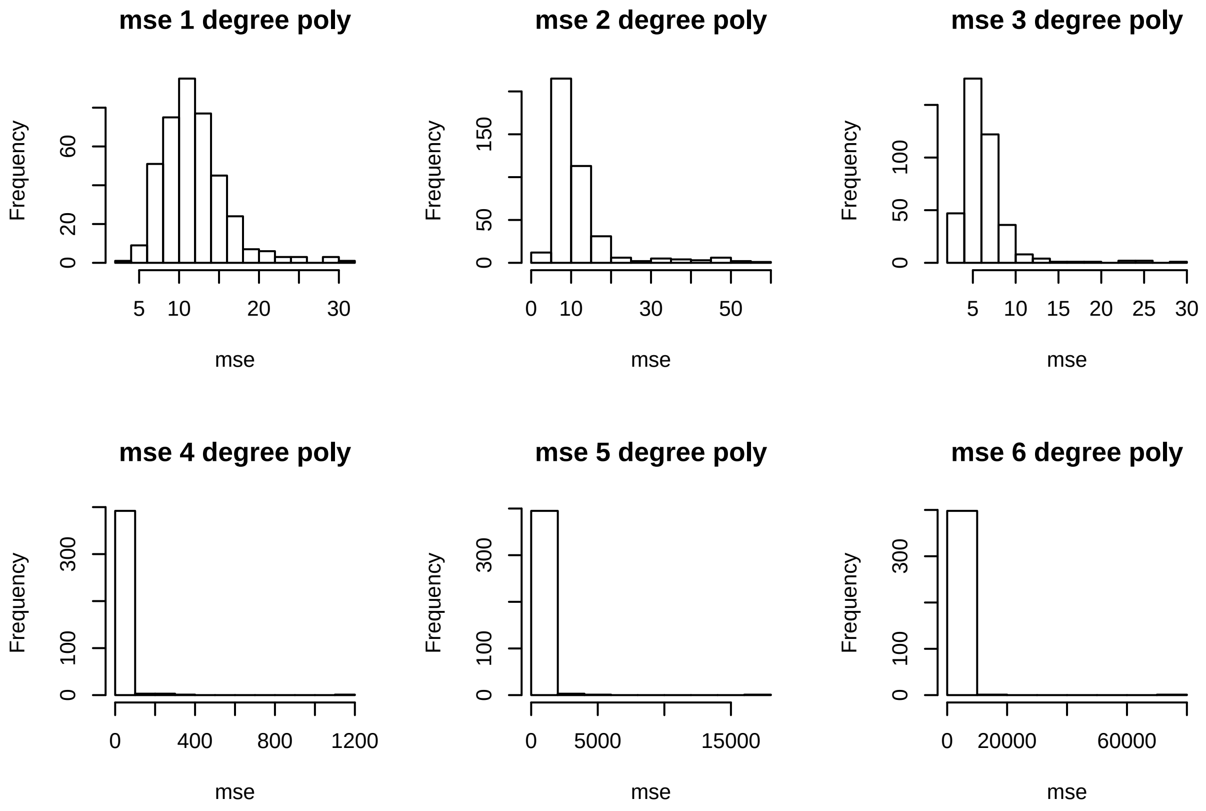
```
## [1]    4.002199    8.215514    2.967048   62.512233  890.140479 3579.145901
```

After third degree, the higher the degree the higher the standard deviation(variance) of the

MSE

4.

```r
par(mfrow=c(2, 3))
for (i in 1:6) {
  hist(performance.boot[i,],
       xlab = "mse", main = paste("mse", i, "degree poly"))
}
```



The distributions of mse tend to be more right skewed as the order of degree increases.

5. bootstrap seems decently reliable to avoid model overfitting.