HW5_MSA 8150

Anutida Sangkla 002602236

3/15/2021

Question 1

```
Hitters <- read.csv("MyHitters.csv", header=TRUE,  sep=",")
str(Hitters)
```

```
## 'data.frame':    263 obs. of  20 variables:
##  $ AtBat    : int  475 584 484 642 311 281 193 330 625 190 ...
##  $ Hits     : int  123 158 127 211 81 76 47 77 179 46 ...
##  $ HmRun    : int  27 15 20 14 3 3 10 19 4 2 ...
##  $ Runs     : int  76 70 66 107 42 42 21 47 94 24 ...
##  $ RBI      : int  93 84 65 59 30 25 29 53 60 8 ...
##  $ Walks    : int  72 42 67 52 26 20 24 27 65 15 ...
##  $ Years    : int  4 5 7 5 17 8 6 6 5 5 ...
##  $ CAtBat   : int  1810 2358 3006 2364 8247 2658 1136 1928 1696 479 ...
##  $ CHits    : int  471 636 844 770 2198 657 256 516 476 102 ...
##  $ CHmRun   : int  108 58 116 27 100 48 42 90 12 5 ...
##  $ CRuns    : int  292 265 436 352 950 324 129 247 216 65 ...
##  $ CRBI     : int  343 316 458 230 909 300 139 288 163 23 ...
##  $ CWalks   : int  267 134 377 193 690 179 106 161 166 39 ...
##  $ League   : int  1 1 1 1 1 0 0 1 0 0 ...
##  $ Division : int  0 0 0 1 1 0 1 1 0 1 ...
##  $ PutOuts  : int  226 331 1231 337 153 106 299 149 303 102 ...
##  $ Assists  : int  10 20 80 19 223 144 13 8 450 177 ...
##  $ Errors   : int  6 4 7 4 10 7 5 6 14 16 ...
##  $ Salary   : num  1220 662 1183 740 320 ...
##  $ NewLeague: int  1 1 1 1 1 0 0 1 0 0 ...
```

part (a)

```
set.seed(1)
train<- Hitters[1:131,]
test<- Hitters[132:263,]

## fitting a linear regression model
lm.fit<- lm(Salary~., data = train)
summary(lm.fit)
```

```
## 
## Call:
## lm(formula = Salary ~ ., data = train)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -802.73 -178.60   -7.03  123.23 1754.21
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 299.42849  146.63692   2.042  0.04352 *
## AtBat        -2.54027    1.08881  -2.333  0.02145 *
## Hits          8.36682    4.18091   2.001  0.04781 *
## HmRun        11.64512   10.83139   1.075  0.28465
## Runs         -9.09923    5.00294  -1.819  0.07164 .
## RBI           2.44105    4.53052   0.539  0.59110
## Walks         9.23440    3.14657   2.935  0.00406 **
## Years       -22.93673   20.53294  -1.117  0.26638
## CAtBat       -0.18154    0.23637  -0.768  0.44411
## CHits        -0.11598    1.25713  -0.092  0.92666
## CHmRun       -1.33888    2.56414  -0.522  0.60260
## CRuns         3.32838    1.34538   2.474  0.01488 *
## CRBI          0.07536    1.23878   0.061  0.95160
## CWalks       -1.07841    0.67875  -1.589  0.11494
## League       59.76065  134.76740   0.443  0.65831
## Division    -98.86233   66.90363  -1.478  0.14232
## PutOuts       0.34087    0.13298   2.563  0.01171 *
## Assists       0.34165    0.33215   1.029  0.30591
## Errors       -0.64207    6.58517  -0.098  0.92250
## NewLeague    -0.67442  131.05687  -0.005  0.99590
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 350 on 111 degrees of freedom
## Multiple R-squared:  0.5118, Adjusted R-squared:  0.4283
## F-statistic: 6.125 on 19 and 111 DF,  p-value: 2.285e-10
```

```r
## Reporting the test MSE
y_pred <- predict(lm.fit, test)
y_actual<- test$Salary
lm_mse <- (mean((y_actual - y_pred) ^ 2))
sprintf('%s = %10.3f', 'The test MSE', lm_mse)
```

```
## [1] "The test MSE = 114780.610"
```

From the result, the test MSE of this model is 114780.610.

part (b)

```r
set.seed(1)
library(pls)
```

```
## Warning: package 'pls' was built under R version 4.0.4
```

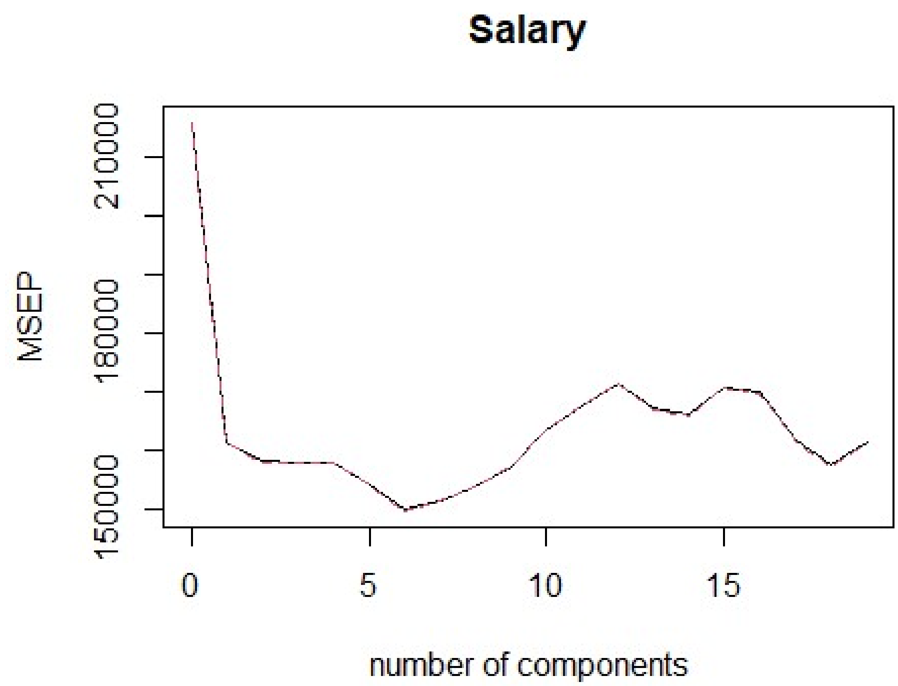
```
##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings

pcr.fit <- pcr(Salary~., data = train, scale = TRUE, validation = "LOO")
summary(pcr.fit)

## Data:    X dimension: 131 19
##  Y dimension: 131 1
## Fit method: svdpc
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 131 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           464.6    401.7    397.5    397.5    397.2    392.4    386.9
## adjCV        464.6    401.6    397.4    397.4    397.1    392.4    386.7
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       388.8    392.0    396.4     404.2     409.3     413.9     408.8
## adjCV    388.7    391.9    396.2     404.0     409.0     413.7     408.6
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        407.4     413.1     412.2     402.0     396.7     401.5
## adjCV     407.1     412.8     411.9     401.7     396.4     401.2
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8
comps
## X         38.89    60.25    70.85    79.06    84.01    88.51    92.61
95.20
## Salary    28.44    31.33    32.53    33.69    36.64    40.28    40.41
41.07
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15
comps
## X         96.78     97.63     98.27     98.89     99.27     99.56
99.78
## Salary    41.25     41.27     41.41     41.44     43.20     44.24
44.30
##         16 comps  17 comps  18 comps  19 comps
## X          99.91     99.97    100.00    100.00
## Salary     45.50     49.66     51.13     51.18

# The graph of cross-validation in terms of number of components based on
MSEP
validationplot(pcr.fit, val.type = 'MSEP')
```



From the model summary, we will see that the 6 components generate the least cross-validation which equals to 386.9. (We know that this is root mean square error, so if we want MSE, we need to square this quantity.) MSE of 6 components equals to 149691.6.

```
# Test MSE
pcr.pred <- predict(pcr.fit, newdata = test, ncomp = 6)
pcr_mse <- (mean((y_actual - pcr.pred) ^ 2))
sprintf('%s = %10.3f', 'The test MSE', pcr_mse)

## [1] "The test MSE =  96587.921"
```

From the result, the test MSE of this model is 96587.921 which is less than the MSE in part (a). As a result, we can see an improvement in accuracy compared to part (a).

part (c)

```
# Center and Scale X
feature <- Hitters[-19]
X <- as.matrix(feature)
X <- scale(X, scale = TRUE)
head(X)

##           AtBat      Hits     HmRun      Runs       RBI     Walks
## [1,] 0.4844122 0.3361993 1.7563137 0.8322203 1.6039009 1.4221315
## [2,] 1.2243624 1.1118170 0.3859982 0.5972930 1.2561785 0.0407924
```

```
## [3,]  0.5455090  0.4248413  0.9569630  0.4406748  0.5220978  1.1919083
## [4,]  1.6180974  2.2863237  0.2718053  2.0460114  0.2902828  0.5012388
## [5,] -0.6289073 -0.5945419 -0.9843172 -0.4990344 -0.8301561 -0.6959218
## [6,] -0.8325634 -0.7053444 -0.9843172 -0.4990344 -1.0233352 -0.9721896
##            Years       CAtBat       CHits     CHmRun       CRuns
CRBI
## [1,] -0.69087451 -0.3706595179 -0.38751381  0.4715523 -0.20900009
0.03890849
## [2,] -0.48226372 -0.1310005959 -0.13296260 -0.1367381 -0.29052218 -
0.04458779
## [3,] -0.06504215  0.1523917061  0.18792619  0.5688787  0.22578439
0.39454083
## [4,] -0.48226372 -0.1283765931  0.07376383 -0.5138782 -0.02783989 -
0.31053893
## [5,]  2.02106574  2.4444581483  2.27678880  0.3742258  1.77772346
1.78923809
## [6,]  0.14356864  0.0001995439 -0.10056518 -0.2583962 -0.11238132 -
0.09406708
##           CWalks     League   Division    PutOuts    Assists     Errors
## [1,]  0.02550157  1.0567429 -1.0172561 -0.2311648 -0.7496555 -0.3925114
## [2,] -0.47817972  1.0567429 -1.0172561  0.1439228 -0.6807283 -0.6952402
## [3,]  0.44208008  1.0567429 -1.0172561  3.3589598 -0.2671650 -0.2411471
## [4,] -0.25474215  1.0567429  0.9792988  0.1653564 -0.6876210 -0.6952402
## [5,]  1.62743530  1.0567429  0.9792988 -0.4919400  0.7184941  0.2129461
## [6,] -0.30776123 -0.9427059 -1.0172561 -0.6598364  0.1739691 -0.2411471
##       NewLeague
## [1,]  1.0730066
## [2,]  1.0730066
## [3,]  1.0730066
## [4,]  1.0730066
## [5,]  1.0730066
## [6,] -0.9284171
```

```
# Xtr, Xts, ytr, yts
X.train<- X[1:131,]
X.test<-X[132:263,]
y.tr<- train$Salary
y.ts<- test$Salary
```

Take an SVD of X train to produce the matrices

```
SVD <- svd(X.train)

# 6 rows and column of ds
d<-diag(SVD$d[1:6])
print(d)
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
## [1,] 30.52191  0.00000  0.00000  0.00000  0.00000  0.00000
## [2,]  0.00000 23.17414  0.00000  0.00000  0.00000  0.00000
## [3,]  0.00000  0.00000 16.28562  0.00000  0.00000  0.00000
```

```
## [4,]  0.00000  0.00000  0.00000 13.98578  0.00000  0.00000
## [5,]  0.00000  0.00000  0.00000  0.00000 11.07932  0.00000
## [6,]  0.00000  0.00000  0.00000  0.00000  0.00000 10.13017
```

```
dim(d)
```

```
## [1] 6 6
```

```
# 6 columns of U
u.tr<-SVD$u[,1:6]
dim(u.tr)
```

```
## [1] 131   6
```

```
# 6 columns of V
v<-SVD$v[,1:6]
dim(v)
```

```
## [1] 19  6
```

Reduction the matrices

```
C<-  v  %*% solve(d)
print(C)
```

```
##                [,1]         [,2]          [,3]         [,4]          [,5]
##  [1,] -0.0082885758  0.016221829 -0.0014580873  0.002673786 -0.0090218047
##  [2,] -0.0081051391  0.016379509  0.0001163945  0.001336259 -0.0063275234
##  [3,] -0.0065183483  0.007024864  0.0065529014 -0.022091391  0.0018109076
##  [4,] -0.0077135025  0.015959291  0.0039751501 -0.007091922 -0.0033118365
##  [5,] -0.0083603520  0.011470960  0.0024224652 -0.010394974  0.0019972247
##  [6,] -0.0076375042  0.009278033 -0.0030056134 -0.009460491 -0.0010047132
##  [7,] -0.0083825196 -0.012490249  0.0001269619  0.006174271  0.0021290080
##  [8,] -0.0103661799 -0.009166026 -0.0026082353  0.007753754  0.0023303944
##  [9,] -0.0104200732 -0.008504373 -0.0025134652  0.007030430  0.0029896739
## [10,] -0.0097534225 -0.008481161  0.0031336173 -0.006039137 -0.0003862224
## [11,] -0.0103726261 -0.007761843 -0.0004961119  0.004753712  0.0042329299
## [12,] -0.0107694389 -0.009793481 -0.0010111850  0.001537655  0.0014788721
## [13,] -0.0095559886 -0.008721560  0.0001396867  0.002460232  0.0049494028
## [14,]  0.0008459058 -0.002025148 -0.0399851092 -0.015596816  0.0050121792
## [15,] -0.0001921613 -0.006183547 -0.0006411534 -0.005054658 -0.0885840120
## [16,] -0.0026381464  0.004605511 -0.0078731169 -0.019236195 -0.0029352812
## [17,] -0.0015774937  0.008230119 -0.0143274771  0.044664313 -0.0066815509
## [18,] -0.0015051784  0.007659089 -0.0144548677  0.035825849 -0.0039958207
## [19,]  0.0002052783 -0.001569024 -0.0399372274 -0.015336651  0.0008644338
##                [,6]
##  [1,]  0.0018286985
##  [2,]  0.0015937031
##  [3,]  0.0345520823
##  [4,]  0.0067188374
##  [5,]  0.0224674920
```

```
##  [6,] -0.0241652696
##  [7,]  0.0014504660
##  [8,] -0.0026643845
##  [9,] -0.0044297111
## [10,]  0.0093098095
## [11,] -0.0027227627
## [12,]  0.0001062401
## [13,] -0.0076442291
## [14,]  0.0123780834
## [15,]  0.0040265524
## [16,] -0.0824093964
## [17,] -0.0025945959
## [18,] -0.0019507381
## [19,]  0.0160749330
```

Fitting a linear model

```
lm_mod<- lm(y.tr~u.tr)
summary(lm_mod)
```

```
##
## Call:
## lm(formula = y.tr ~ u.tr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -863.64 -172.53  -30.62  120.24 2015.99
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   554.50      32.48  17.073  < 2e-16 ***
## u.tr1       -2866.66     366.71  -7.817 1.99e-12 ***
## u.tr2         648.39     365.80   1.773  0.07877 .
## u.tr3        -619.61     370.29  -1.673  0.09679 .
## u.tr4        -550.47     366.07  -1.504  0.13519
## u.tr5         945.72     365.99   2.584  0.01093 *
## u.tr6       -1020.48     365.91  -2.789  0.00612 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 365.8 on 124 degrees of freedom
## Multiple R-squared:  0.4042, Adjusted R-squared:  0.3753
## F-statistic: 14.02 on 6 and 124 DF,  p-value: 3.941e-12
```

Testing model

```
Utest<- X.test %*% C
test_mat<- model.matrix(y.ts~Utest)
coefs = coef(lm_mod, id = i)
y_pred = test_mat %*%coefs
```

```r
mse <- (mean((y.ts - y_pred) ^ 2))
sprintf('%s = %10.3f', 'The test MSE', mse)

## [1] "The test MSE =  96860.277"

#### Percentage difference
diff<- ((mse - pcr_mse)/pcr_mse)*100
sprintf('%s = %10.3f', 'The difference of MSE between part b and part c',
diff)

## [1] "The difference of MSE between part b and part c =      0.282"
```

From the results, we can see that the MSE of part (c), which is 96860.277, is different than the MSE of part (b), which is 96587.921, up to 1%.

Question 2

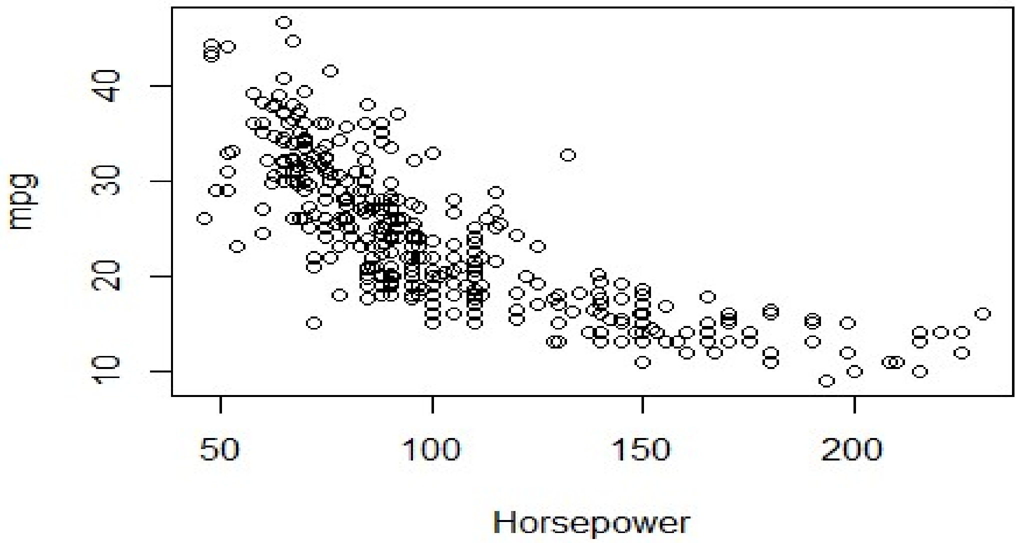
```r
auto <- read.csv("Auto.csv", header=TRUE,  sep=",")
str(auto)

## 'data.frame':    392 obs. of  8 variables:
##  $ mpg         : num  14.5 25.5 22.5 13 27.9 18.6 33.5 12 29.5 14 ...
##  $ cylinders   : int  8 4 6 8 4 6 4 8 4 8 ...
##  $ displacement: num  351 140 232 307 156 225 151 383 98 351 ...
##  $ horsepower  : int  152 89 90 130 105 110 90 180 68 148 ...
##  $ weight      : int  4215 2755 3085 4098 2800 3620 2556 4955 2135 4657
...
##  $ acceleration: num  12.8 15.8 17.6 14 14.4 18.7 13.2 11.5 16.6 13.5 ...
##  $ year        : int  76 77 76 72 80 78 79 71 78 75 ...
##  $ origin      : int  1 1 1 1 1 1 1 1 3 1 ...
```

part (a)

Plotting mpg in terms of horsepower

```r
plot(auto$mpg~auto$horsepower, xlab = 'Horsepower', ylab = 'mpg')
```



Fitting a polynomail model

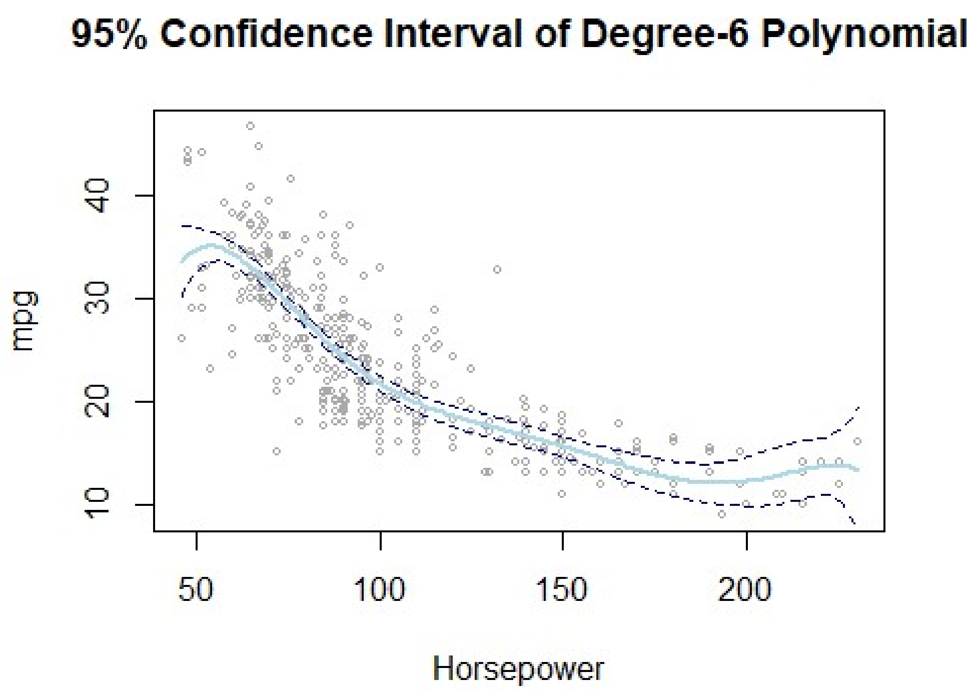
```
poly.fit <- lm(mpg~poly(horsepower,6), data = auto)
summary(poly.fit)

##
## Call:
## lm(formula = mpg ~ poly(horsepower, 6), data = auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.595  -2.571  -0.269   2.209  15.362
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            23.4459     0.2177 107.715  < 2e-16 ***
## poly(horsepower, 6)1 -120.1377     4.3096 -27.877  < 2e-16 ***
## poly(horsepower, 6)2   44.0895     4.3096  10.231  < 2e-16 ***
## poly(horsepower, 6)3   -3.9488     4.3096  -0.916  0.36008
## poly(horsepower, 6)4   -5.1878     4.3096  -1.204  0.22941
## poly(horsepower, 6)5   13.2722     4.3096   3.080  0.00222 **
## poly(horsepower, 6)6   -8.5462     4.3096  -1.983  0.04807 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.31 on 385 degrees of freedom
```

```
## Multiple R-squared:  0.6998, Adjusted R-squared:  0.6951
## F-statistic: 149.6 on 6 and 385 DF,  p-value: < 2.2e-16
```

Plot 95% confidence interval

```r
## Plotting 95% confidence interval
hpwlims<- range(auto$horsepower)
hpw.grid<- seq(from = hpwlims[1], to = hpwlims[2])
pred<- predict(poly.fit, newdata = list(horsepower = hpw.grid), interval =
'confidence', level = 0.95)
plot(auto$horsepower, auto$mpg, xlim = hpwlims, cex = 0.5, col = 'darkgrey',
xlab = 'Horsepower', ylab = 'mpg', main = "95% Confidence Interval of Degree-
6 Polynomial")
lines(hpw.grid, pred[,1], lwd = 2, col = 'lightblue')
matlines(hpw.grid, pred[,-1], lwd = 1, col= 'darkblue', lty = 2)
```



part (b)

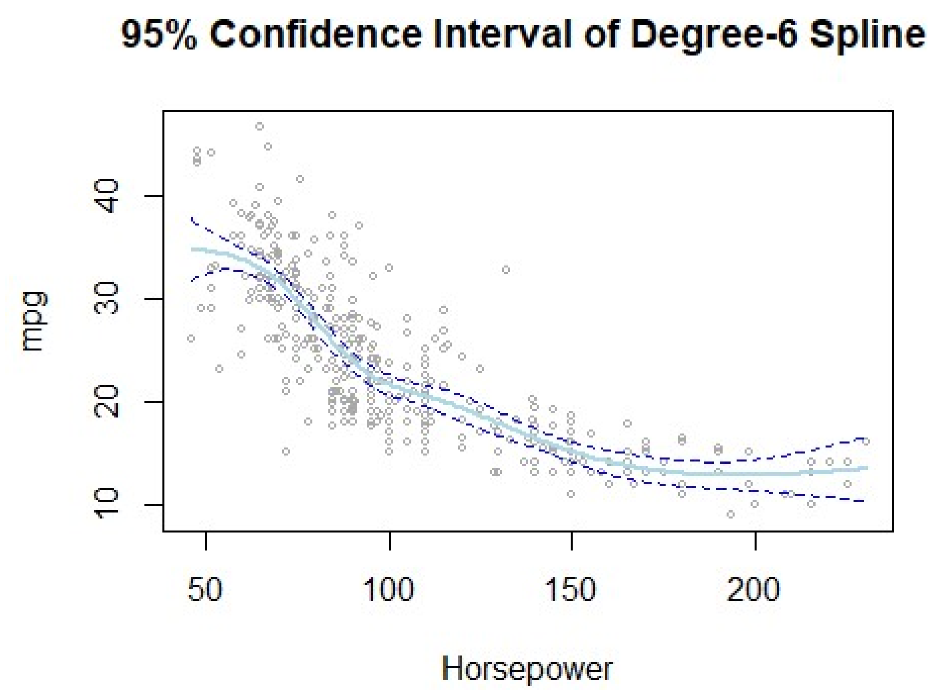
Fitting model to a natural spline

```
library(splines)
fit<- lm(mpg~ns(horsepower, 6), data = auto)
summary(fit)

##
## Call:
## lm(formula = mpg ~ ns(horsepower, 6), data = auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.9491  -2.6183  -0.1595   2.3508  15.1349
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)           34.738      1.509  23.021  < 2e-16 ***
## ns(horsepower, 6)1    -8.210      1.594  -5.149 4.18e-07 ***
## ns(horsepower, 6)2   -13.046      1.835  -7.108 5.76e-12 ***
## ns(horsepower, 6)3   -14.577      1.886  -7.730 9.50e-14 ***
## ns(horsepower, 6)4   -22.802      1.624 -14.039  < 2e-16 ***
## ns(horsepower, 6)5   -22.758      3.512  -6.480 2.81e-10 ***
## ns(horsepower, 6)6   -20.849      1.742 -11.967  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.302 on 385 degrees of freedom
## Multiple R-squared:  0.7009, Adjusted R-squared:  0.6962
## F-statistic: 150.4 on 6 and 385 DF,  p-value: < 2.2e-16
```

Plotting 95% confidence interval

```
hpwlims<- range(auto$horsepower)
hpw.grid<- seq(from = hpwlims[1], to = hpwlims[2])
pred<- predict(fit, newdata = list(horsepower = hpw.grid), interval =
'confidence', level =0.95)
plot(auto$horsepower, auto$mpg, xlim = hpwlims, cex = 0.5, col = 'darkgrey',
xlab = 'Horsepower', ylab = 'mpg', main = "95% Confidence Interval of Degree-
6 Spline ")
lines(hpw.grid, pred[,1], lwd = 2, col = 'lightblue')
matlines(hpw.grid, pred[,-1], lwd = 1, col= 'blue', lty = 2)
```



From the result, natural spline fit in part (b) seems to have a narrower confidence interval around the boundaries compared to part (a) when the horsepower is large. However, part (a) seems to have a narrower confidence interval around the boundaries compared to part (b) when the horsepower is small.

part (c)

```
train<- auto[1:350,]
test<- auto[351:392,]

## Fitting linear model
lm.fit<- lm(mpg~horsepower+acceleration+year, data = train)
summary(lm.fit)

##
## Call:
## lm(formula = mpg ~ horsepower + acceleration + year, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.7704  -3.0253  -0.6865   2.1362  15.4392
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.641629   6.015622  -0.107    0.915
## horsepower  -0.160214   0.008482 -18.888  < 2e-16 ***
```

```
## acceleration -0.581611   0.112584  -5.166 4.05e-07 ***
## year          0.657839   0.068474   9.607  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.301 on 346 degrees of freedom
## Multiple R-squared:  0.7079, Adjusted R-squared:  0.7054
## F-statistic: 279.5 on 3 and 346 DF,  p-value: < 2.2e-16
```

```
## Testing the model
y_pred <- predict(lm.fit, test)
y_actual<- test$mpg
lm_mse <- (mean((y_actual - y_pred) ^ 2))
sprintf('%s = %10.3f', 'The test MSE', lm_mse)
```

```
## [1] "The test MSE =     12.059"
```

```
## Fitting GAM model
library(splines)
library(gam)
```

```
## Warning: package 'gam' was built under R version 4.0.4
```

```
## Loading required package: foreach
```

```
## Warning: package 'foreach' was built under R version 4.0.3
```

```
## Loaded gam 1.20
```

```
gam.fit<-gam(mpg~ns(horsepower, 4)+ ns(acceleration,4)+ year, data = train)
summary(gam.fit)
```

```
##
## Call: gam(formula = mpg ~ ns(horsepower, 4) + ns(acceleration, 4) +
##     year, data = train)
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -11.718  -1.935  -0.272   1.480  13.055
##
## (Dispersion Parameter for gaussian family taken to be 11.6243)
##
##     Null Deviance: 21912.78 on 349 degrees of freedom
## Residual Deviance: 3952.262 on 340 degrees of freedom
## AIC: 1863.696
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                       Df  Sum Sq Mean Sq F value    Pr(>F)
## ns(horsepower, 4)      4 15282.5  3820.6 328.676 < 2.2e-16 ***
```

```
## ns(acceleration, 4)   4   740.1   185.0  15.917 5.913e-12 ***
## year                  1  1937.9  1937.9 166.714 < 2.2e-16 ***
## Residuals           340  3952.3    11.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## Testing the model
y_pred <- predict(gam.fit, test)
y_actual<- test$mpg
gam_mse <- (mean((y_actual - y_pred) ^ 2))
sprintf('%s = %10.3f', 'The test MSE', gam_mse)
```

```
## [1] "The test MSE =      8.100"
```

Comparing two models:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
data.frame(Models = c("Linear Regression", "GAM"),
           Test_Error = c(lm_mse, gam_mse))
```

```
##              Models Test_Error
## 1 Linear Regression  12.059045
## 2               GAM   8.100448
```

From the result above, the test error of GAM model ,which equals 8.10, is smaller than the test error of linear regression model.