Jeffrey Kerley
Jakcqc
Homework: Ch 07
STAT 4510/7510
Due Tuesday, April 12, 11:59 pm

Problem 1

Let's consider the following data set.

y	x
3.5	0.2
1.2	0.7
6.9	1.2
2.6	3.6
4.4	3.8
3.0	4.0
8.5	4.2

Suppose we try to fit the step function model with two cutpoints c1 = 1 and c2 = 3.7.

$yi = \beta0 + \beta1I(1 \le xi < 3.7) + \beta2I(xi \ge 3.7) + i$

where $I(\cdot)$ is an indicator function that returns a 1 if the condition is true, and returns a 0 otherwise.

Find the estimates $\hat{}\beta0$, $\hat{}\beta1$, and $\hat{}\beta2$. You can do this manually. Note that the predicted values of the response
should be the mean of yi's that belong to the corresponding range of x.

I used R for Calculation

```
> meansM
       B0   B1  B2
[1,] 2.35 2.35 3.7
```

Problem 2

Let's consider the following function.

$f(x) = 3 + 5x + 3x2 + 1.5x3 + (x - 1)3+$

where $(x - 1)3+ = (x - 1)3$ when $x > 1$ and $(x - 1)3+ = 0$ when $x \le 1$. This function consists of the basis
functions $b1(x) = x, b2(x) = x2, b3(x) = x3, b4(x) = (x - 1)3+$. We will validate this function is the piecewise

cubic polynomial that is continuous at the knot point $x = 1$ up to the second derivative.
(a) Find the cubic polynomial function $f1(x) = a1 + b1x + c1x2 + d1x3$ when $x \leq 1$.
If $f(x) = B0 + B1X + B2X^2 + B3X^3 + B4(X -1)^3$
And we know that $(X-1)^3 = 0$ when $X \leq 1$, then we can reduce our f(x) and put in terms of a1, b1, c1, and d1.

$f1(x) = a1 + b1X + c1X^2 + d1X^3$ which is equal to our f1(x). Our B1 = a1… B3 = d3 are all equal still.

(b) Find the cubic polynomial function $f2(x) = a2 + b2x + c2x2 + d2x3$ when $x > 1$.

If $f(x) = B0 + B1X + B2X^2 + B3X^3 + B4(X -1)^3$
We expand our trinomial when $(X-1)^3 > 0$ as it is equal $= (X-1)^3$ when $x > 1$.
$f(x) = B0 + B1X + B2X^2 + B3X^3 + B4(X^3 - 2X^2 + X - X^2 + 2X - 1)$
$f(x) = B0 + B1X + B2X^2 + B3X^3 + B4(X^3 - 3X^2 + 3X - 1)$
Next we group our X to find our B coefficients.
$f2(x) = (B0 - B4) + (B1 + 3B4)X + (B2 - 3B4)X^2 + (B3 + B4)X^3$
$f2(x) = a1 + b1X + c2X^2 + d3X^3$ where we have
$a1= (B0 - B4)$
$b1 = (B1 + 3B4)$
$c1 = (B2 - 3B4)$
$d1 = (B3 + B4)$

(c) Validate if f(x) is continuous at $x = 1$, that is $f1(1) = f2(1)$.

Here we need to prove the two functions are equal at 1. This is simple to show.
We simply plug in 1 at each function, and compare our coefficients and that they are equal.
$f1(1) = B0 + B1 + B2 + B3$
$F2(1) = (B0 - B4) + (B1 + 3B4)1 + (B2 - 3B4)1^2 + (B3 + B4)1^3$
$= B0 - B4 + B1 + 3B4 + B2 - 3B4 + B3 + B4$
$= B0 + B1 + B2 + B3 + (B4 - 3B4 + 3B4 -B4)$
$= B0 + B1 + B2 + B3$
$f1(1) = f2(1) = B0 + B1 + B2 + B3$

(d) Validate if f′(x) is continuous at $x = 1$, that is $f'1(1) = f'2(1)$.

$f1(x) = B0 + B1X + B2X^2 + B3X^3$
$f1`(1) = B1 + 2B2 + 3B3$
We can use our f2(1) from part c $= B0 + B1 + B2 + B2 + B3$
$f2`(1) = B1 + 2B2 + 3B3$

We again can see our first derivative f1 and f2 are equal again at 1, proving their continuity.

(d) Validate if $f''(x)$ is continuous at $x = 1$, that is $f''_1(1) = f''_2(1)$.

Here f1` =B1 + 2B2X + 3B2X^2
f1``(1) = 2B2 + 6B3
We use our first derivation for f2 = B1 + 2B2X + 3B3X^2
f2``(1) = 2B2 + 6B3
This again proves our f1``(1) = f2``(1)

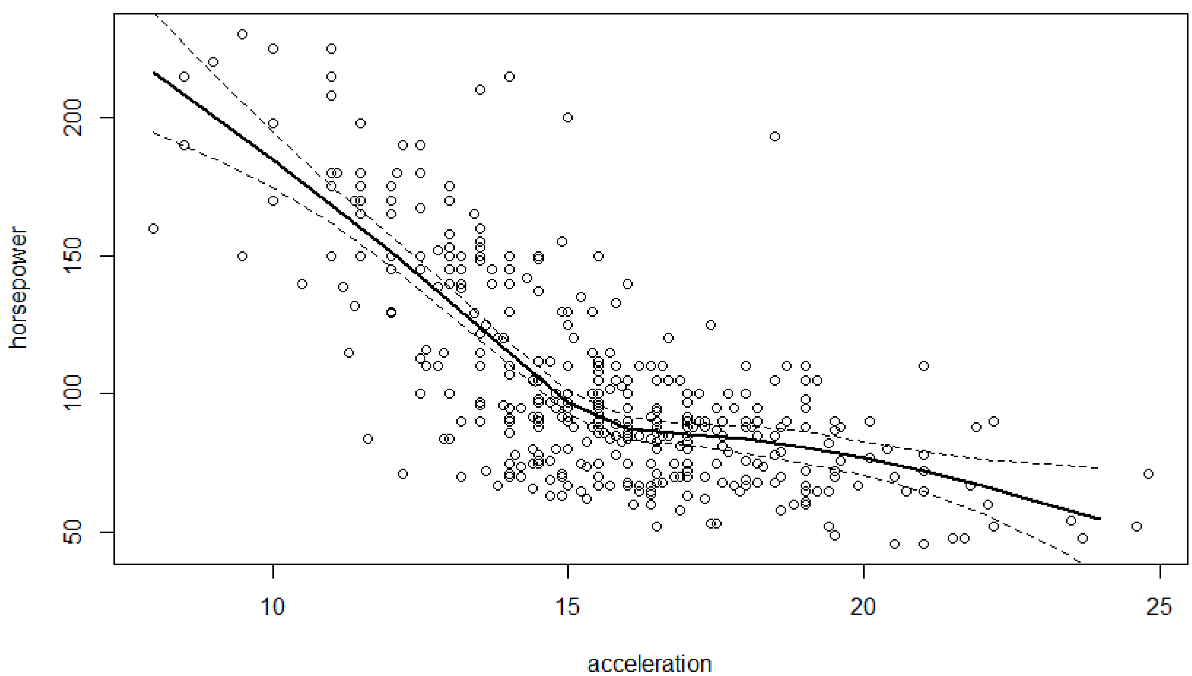
<u>Problem 3</u>
In this problem, we will use the Auto data set. Load the data set onto the global environment by running the
following code.
1
library(ISLR)
Auto <- Auto
attach(Auto)
(a) Suppose we are interested in the relationship between the horsepower and acceleration. Fit the
model to predict horsepower with polynomial functions of acceleration with different degrees from 1
to 3. Compare the models using anova() function. Comment on the outcome.

```
> p1=lm(horsepower~acceleration,data=Auto)
> p2=lm(horsepower~poly(acceleration,2),data=Auto)
> p3=lm(horsepower~poly(acceleration,3),data=Auto)
> anova(p1,p2,p3)
Analysis of Variance Table

Model 1: horsepower ~ acceleration
Model 2: horsepower ~ poly(acceleration, 2)
Model 3: horsepower ~ poly(acceleration, 3)
  Res.Df    RSS Df Sum of Sq       F    Pr(>F)
1    390 304135
2    389 258062  1     46073 69.5644 1.306e-15 ***
3    388 256974  1      1087  1.6418    0.2008
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Using our RSS from p1 to p2 there is a significant decrease, from p2 to p3, we do not see any significant decrease. In tandem with the p3 P value, we can see there is no significance to using P3 over P2.

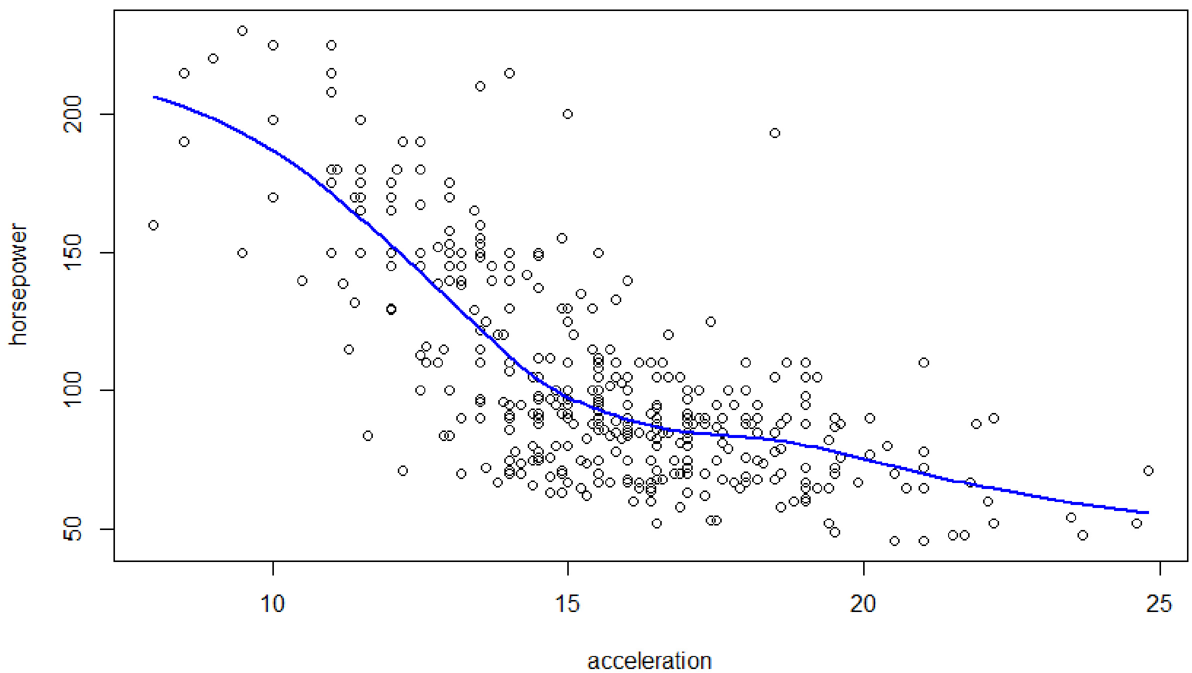
(b) Fit a natural cubic spline model with 4 degrees of freedom. Make predictions on the range of acceleration. Plot the data points and predicted curve with 95% confidence interval.

```
accRange = range(Auto$acceleration)
accGrid = seq(from = accRange[1], to = accRange[2])

splineC3 = lm(horsepower ~ ns(acceleration, df=4), data=Auto)
pAccRange = predict(splineC3, newdata = list(acceleration = accGrid), se = T)

plot(acceleration, horsepower, col="black")
lines(accGrid, pAccRange$fit, lwd=2)
lines(accGrid, pAccRange$fit+2*pAccRange$se, lty="dashed")
lines(accGrid, pAccRange$fit-2*pAccRange$se, lty="dashed")
```



(c) Fit the smoothing spline model with the tuning parameter λ found by cross-validation. Find the
corresponding effective degrees of freedom. Plot the data points and predicted curve.

> #c
> plot(acceleration, horsepower, col="black")
> smoothS = smooth.spline(acceleration, horsepower, cv = TRUE)
Warning message:
In smooth.spline(acceleration, horsepower, cv = TRUE) :
cross-validation with non-unique 'x' values seems doubtful
> smoothS$df
[1] 7.054516
> lines(smoothS,col="blue",lwd=2)



(d) Fit the local regression model with span 0.2 and 0.5. Plot the data points and both curves with the
appropriate legend. Comment on two models regarding variance-bias trade-offs.

> #d
> plot(acceleration, horsepower, xlim=accRange, col="red")
> LR2=loess(horsepower ~ acceleration, span=.2, data=Auto)
> LR5=loess(horsepower ~ acceleration, span=.5, data=Auto)
> lines(accGrid, predict(LR2, data.frame(acceleration=accGrid)), col="green", lwd=2)
> lines(accGrid, predict(LR5, data.frame(acceleration=accGrid)), col="blue", lwd=2)

If we evaluate the variance-bias tradeoffs, we can first look that our span .2 has equivalent 16.42 parameters, and .5 span only has 7.73 equivalent parameters. This leads us to think that our .2 span has greater variance in respect to the data, while our .5 span has less variance but greater bias.

(e) Let's now consider to use weight in addition to acceleration to predict horsepower. We will use
GAM to fit the model. Fit 3 different models as follows and compare them with anova() function.
Comment on the outcome.
•gam1 - without weight and smoothe spline of acceleration with 5 dof.
•gam2 - with linear term of weight and smoothe spline of acceleration with 5 dof.
•gam3 - with smooth spline of weight with 5 dof and smoothe spline of acceleration with 5 dof.

```
> gam1 = lm(horsepower ~  ns(acceleration, 5), data = Auto)
> gam2 = lm(horsepower ~ ns(weight) + ns(acceleration, 5), data = Auto)
> gam3 = lm(horsepower ~ ns(weight, 4) + ns(acceleration, 5), data = Auto)

> anova(gam1,gam2,gam3)
Analysis of Variance Table

Model 1: horsepower ~ ns(acceleration, 5)
Model 2: horsepower ~ ns(weight) + ns(acceleration, 5)
Model 3: horsepower ~ ns(weight, 4) + ns(acceleration, 5)
  Res.Df    RSS Df Sum of Sq         F    Pr(>F)
1    386 243270
2    385  58055  1    185215 1269.8113 < 2.2e-16 ***
3    382  55719  3      2336    5.3393  0.001299 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can see that our lowest errors happen with model 3, indication it is probably our best model.

(f) Using the best model you found from (e), plot the fitted curves for each predictor.

RCODE:

```
##Homework,7
library(splines)
library(ISLR)

#Problem,1
```

```
stepFR=t(matrix(c(
 3.5,0.2
 ,1.2,0.7
 ,6.9,1.2
 ,2.6,3.6
 ,4.4,3.8
 ,3.0,4.0
 ,8.5,4.2), nrow = 2))
colnames(stepFR) = c('x','y')
stepFR

means = c(0,0,0)
meanCount = c(0,0,0)
for(x in 1:6){
  if(stepFR[x,2] >= 3.7){
    means[3] = means[3] + stepFR[x,1]
    meanCount[3] = meanCount[3] + 1
  }
  if(stepFR[x,2] < 3.7 && stepFR[x,2] <= 1){
    means[2] = means[2] + stepFR[x,1]
    meanCount[2] = meanCount[2] + 1

  }
  if(stepFR[x,2] < 1){
    means[1] = means[1] + stepFR[x,1]
    meanCount[1] = meanCount[1] + 1

  }
}

means = means/meanCount
meansM = matrix(means, ncol = 3)
meansM
colnames(meansM) = c('B0', 'B1', 'B2')
meansM



#Problem 2
#see work on doc
```

```
#problem 3
Auto <- Auto
attach(Auto)


p1=lm(horsepower~acceleration,data=Auto)
p2=lm(horsepower~poly(acceleration,2),data=Auto)
p3=lm(horsepower~poly(acceleration,3),data=Auto)

anova(p1,p2,p3)

#b

accRange = range(Auto$acceleration)
accGrid = seq(from = accRange[1], to = accRange[2])

splineC3 = lm(horsepower ~ ns(acceleration, df=4), data=Auto)
pAccRange = predict(splineC3, newdata = list(acceleration = accGrid), se = T)

plot(acceleration, horsepower, col="black")
lines(accGrid, pAccRange$fit, lwd=2)
lines(accGrid, pAccRange$fit+2*pAccRange$se, lty="dashed")
lines(accGrid, pAccRange$fit-2*pAccRange$se, lty="dashed")

#c
plot(acceleration, horsepower, col="black")
smoothS = smooth.spline(acceleration, horsepower, cv = TRUE)
smoothS$df
lines(smoothS,col="blue",lwd=2)

#d
plot(acceleration, horsepower, xlim=accRange, col="red")
LR2=loess(horsepower ~ acceleration, span=.2, data=Auto)
LR5=loess(horsepower ~ acceleration, span=.5, data=Auto)

lines(accGrid, predict(LR2, data.frame(acceleration=accGrid)), col="green", lwd=2)
lines(accGrid, predict(LR5, data.frame(acceleration=accGrid)), col="blue", lwd=2)

LR2
```

LR5

```
#e and f
gam1 = lm(horsepower ~  ns(acceleration, 5), data = Auto)
gam2 = lm(horsepower ~ ns(weight) + ns(acceleration, 5), data = Auto)
gam3 = lm(horsepower ~ ns(weight, 4) + ns(acceleration, 5), data = Auto)
gam1
gam2
gam3
anova(gam1,gam2,gam3)
```