Statistical Learning - Homework 6 (Applied)

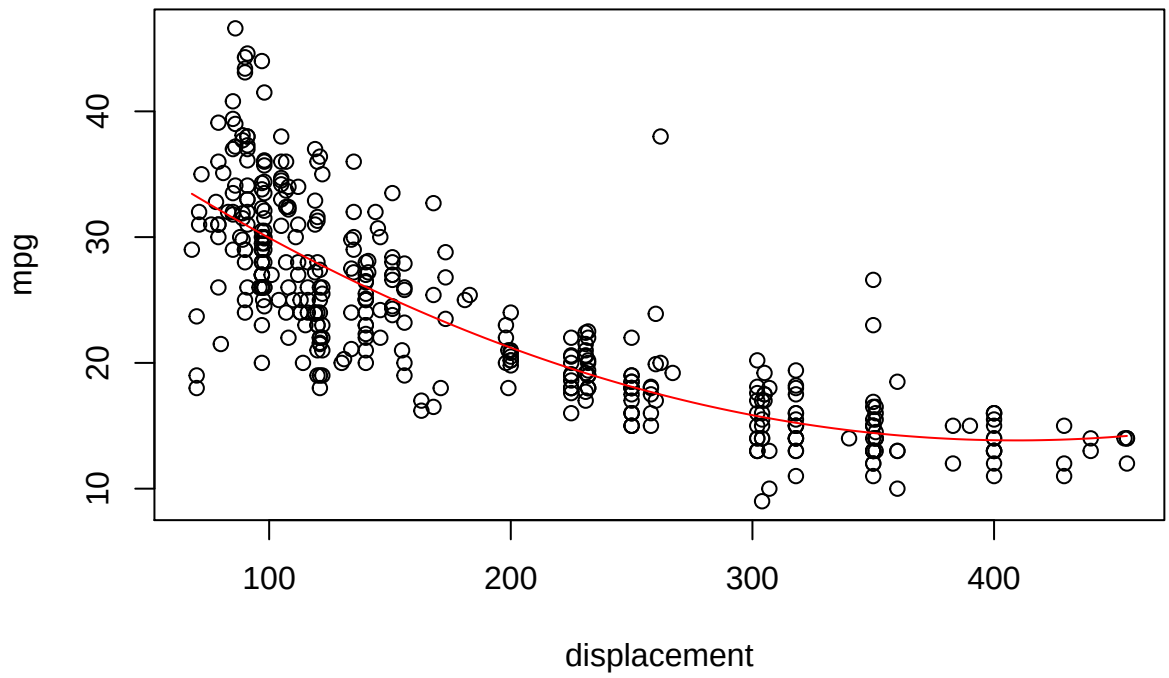
James Hahn

Chapter 7 - Exercise 8

```
library(ISLR)
options(warn=-1)

glm.fit = glm(mpg ~ poly(displacement, 5), data = Auto)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = mpg ~ poly(displacement, 5), data = Auto)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -15.3360   -2.3445   -0.2895    2.1635   20.3439
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)               23.4459     0.2209 106.158  < 2e-16 ***
## poly(displacement, 5)1 -124.2585     4.3728 -28.416  < 2e-16 ***
## poly(displacement, 5)2   31.0895     4.3728   7.110 5.67e-12 ***
## poly(displacement, 5)3   -4.4655     4.3728  -1.021    0.308
## poly(displacement, 5)4    0.7747     4.3728   0.177    0.859
## poly(displacement, 5)5    3.2991     4.3728   0.754    0.451
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 19.12134)
##
##     Null deviance: 23819.0  on 391  degrees of freedom
## Residual deviance:  7380.8  on 386  degrees of freedom
## AIC: 2277.1
##
## Number of Fisher Scoring iterations: 2
```

```
newData = c()
glm.fit = glm(mpg ~ poly(displacement, 2), data = Auto)
newData$pred = predict(glm.fit, newdata = data.frame(displacement = seq(min(Auto$displacement), max(Aut

plot(Auto[, c("displacement", "mpg")])
lines(seq(min(Auto$displacement), max(Auto$displacement), length.out = 100), newData$pred, type = "l",
```



From the above statistics and plot, we can clearly see there is a fair argument that the relationship between displacement and mpg is non-linear. If you look at the above plot, the red polynomial regression line is curved. If the relationship was linear, the red line would be much straighter.

Chapter 7 - Exercise 9

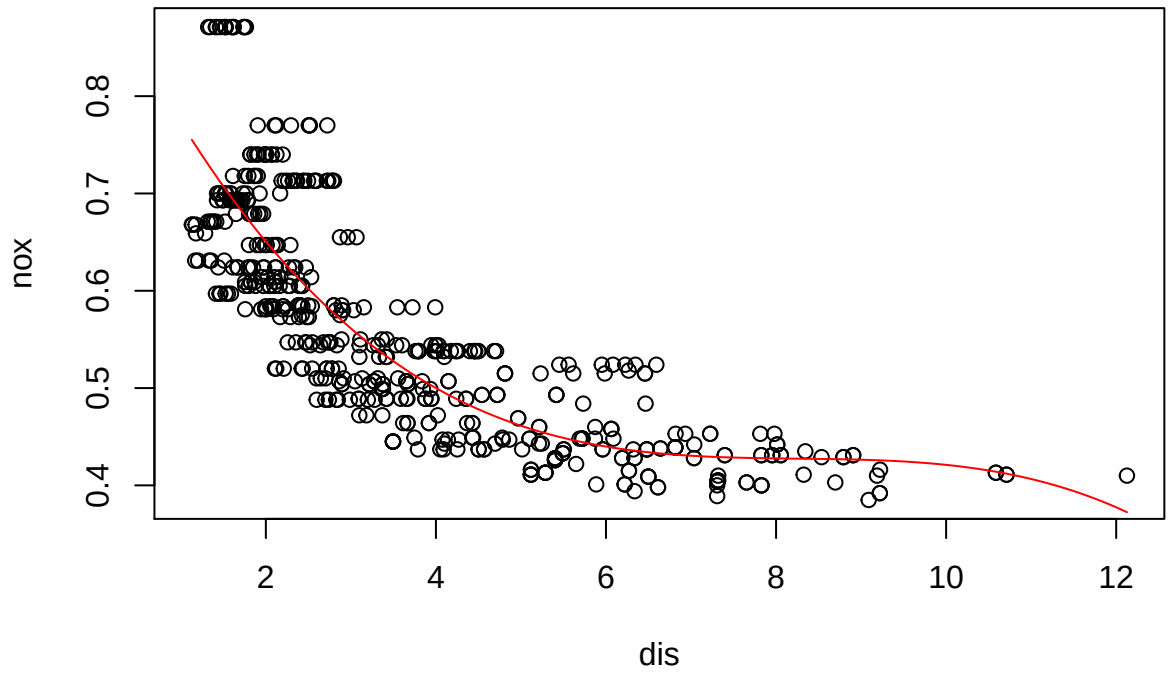
a)

```
library(MASS)

attach(Boston)
poly.fit = glm(nox ~ poly(dis, 3), data = Boston)
summary(poly.fit)
```

```
##
## Call:
## glm(formula = nox ~ poly(dis, 3), data = Boston)
##
## Deviance 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
##
## (Dispersion parameter for gaussian family taken to be 0.003852802)
##
##     Null deviance: 6.7810  on 505  degrees of freedom
## Residual deviance: 1.9341  on 502  degrees of freedom
## AIC: -1370.9
##
## Number of Fisher Scoring iterations: 2
```

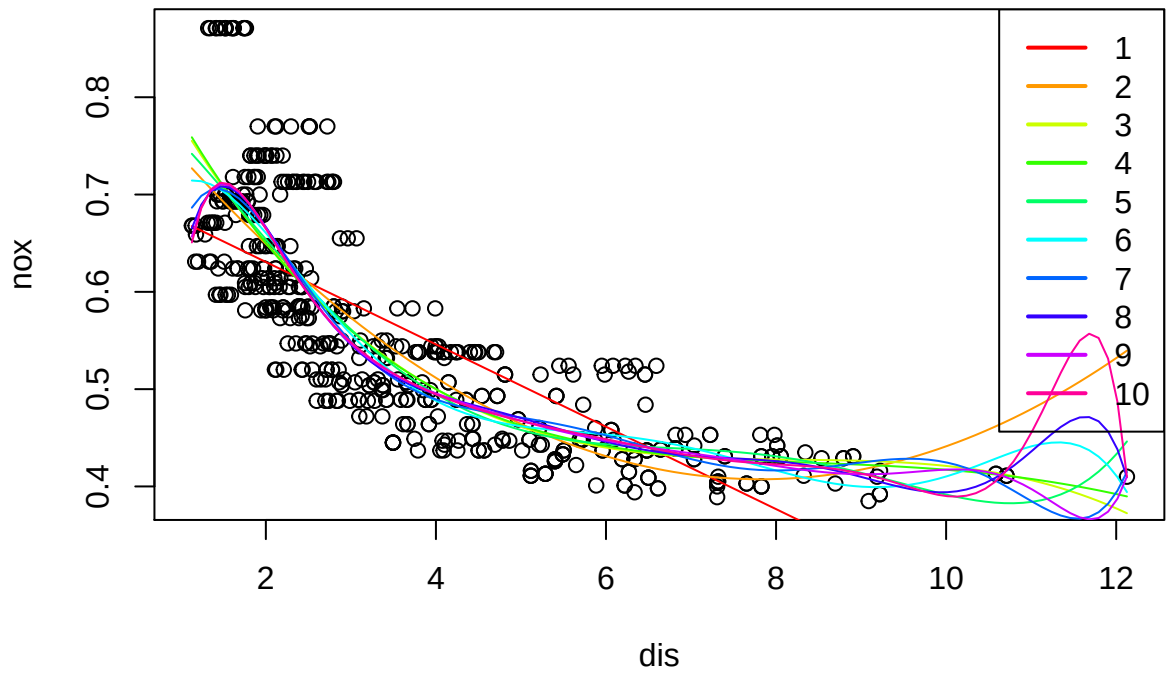
```
plot(Boston[, c('dis', 'nox')])
pred = predict(poly.fit, data.frame(dis=seq(min(dis), max(dis), length.out = 100)))

lines(seq(min(dis), max(dis), length.out = 100), pred, col = "red")
```



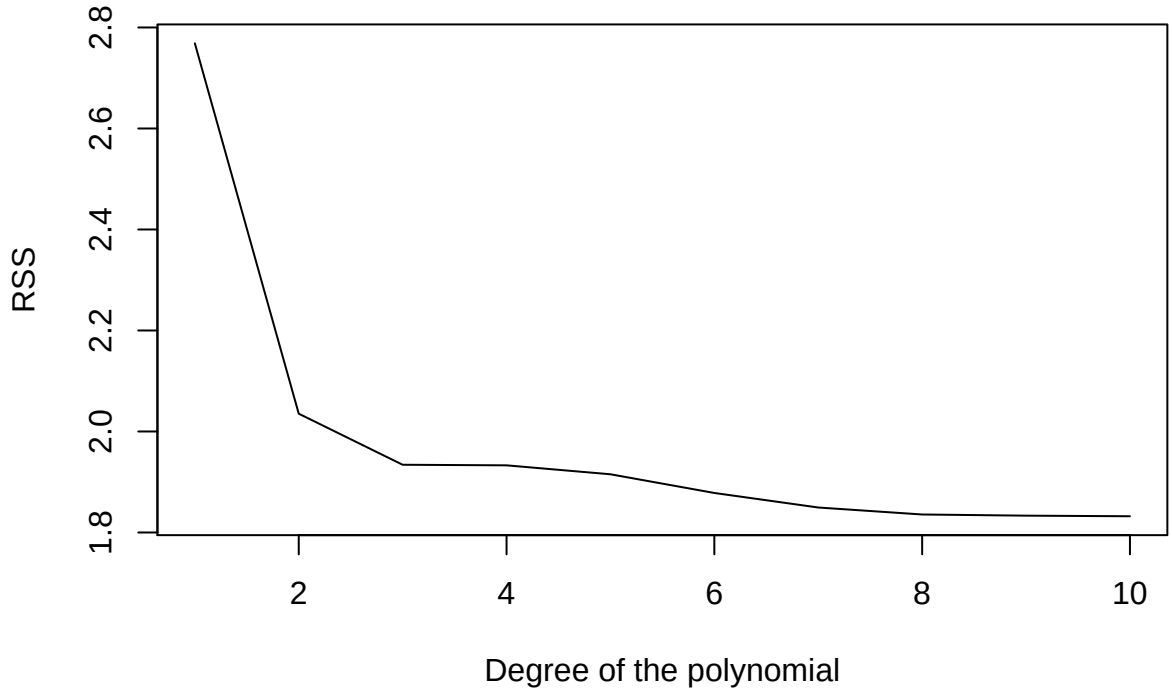
b)

```
x = seq(min(dis), max(dis), length.out = 100)
cols = rainbow(10)
plot(Boston[, c('dis', 'nox')])
rss = c()

for(pwr in 1:10){
  poly.fit = glm(nox ~ poly(dis, pwr), data = Boston)
```

```
  pred = predict(poly.fit, data.frame(dis = x))
  lines(x, pred, col = cols[pwr])

  rss = c(rss, sum(poly.fit$residuals^2))
}

legend(x = 'topright', legend = 1:10, col = cols, lty = c(1, 1), lwd = c(2, 2))
```

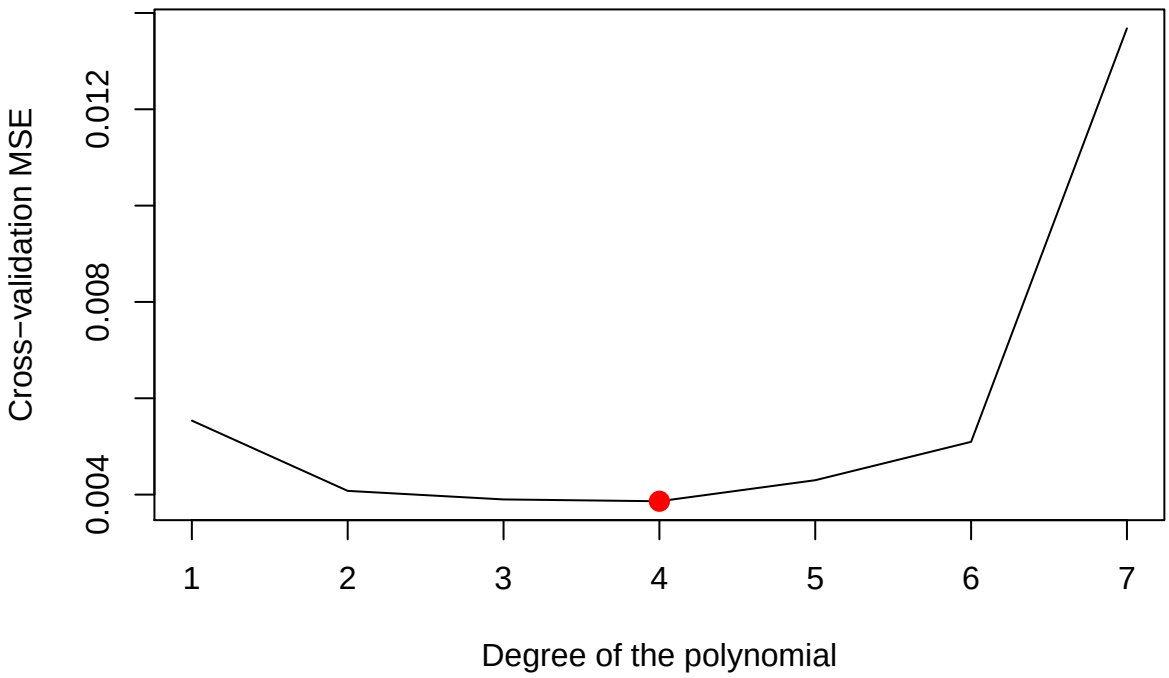



```
plot(rss, xlab = "Degree of the polynomial", ylab = "RSS", type = "l")
```



c)

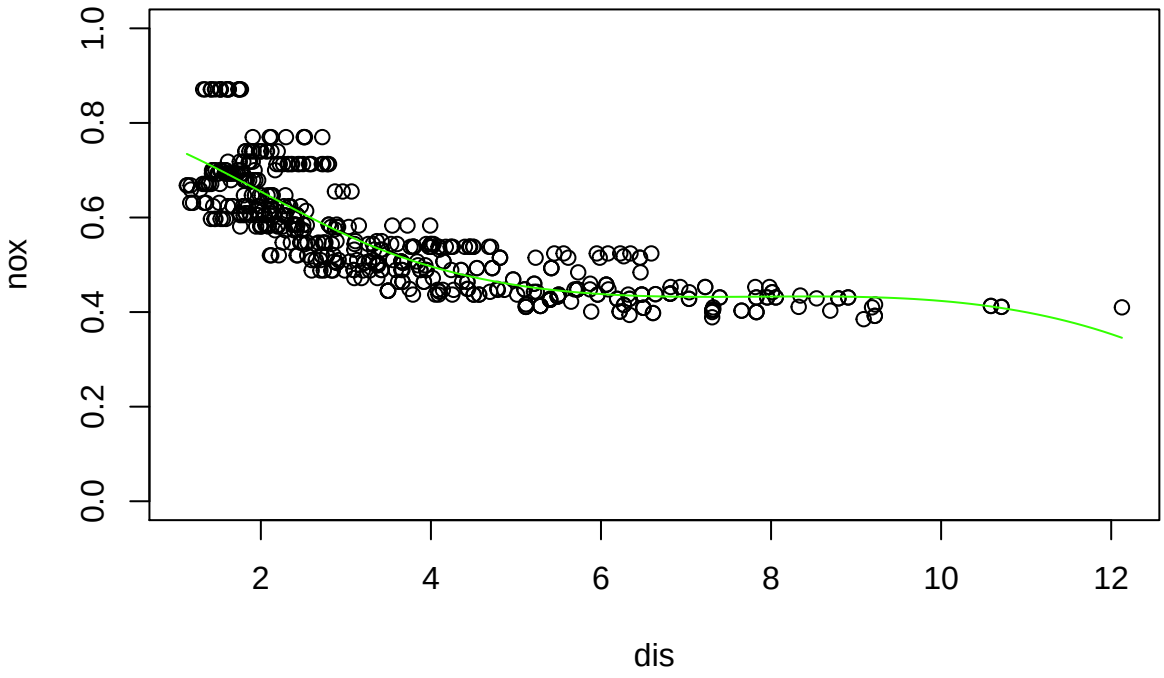
```
library(boot)
set.seed(1)

poly.mse = c()
for(degree in 1:7){
  poly.fit = glm(nox ~ poly(dis, degree, raw = T), data = Boston)
  mse = cv.glm(poly.fit, data = Boston, K = 10)$delta[1]
  poly.mse = c(poly.mse, mse)
}

plot(poly.mse, type = "l", xlab = "Degree of the polynomial", ylab = "Cross-validation MSE")
points(which.min(poly.mse), poly.mse[which.min(poly.mse)], col = "red", pch = 20, cex = 2)
```



Clearly, from the above graphic, the polynomial with the smallest MSE is the one with degree 4.

d)

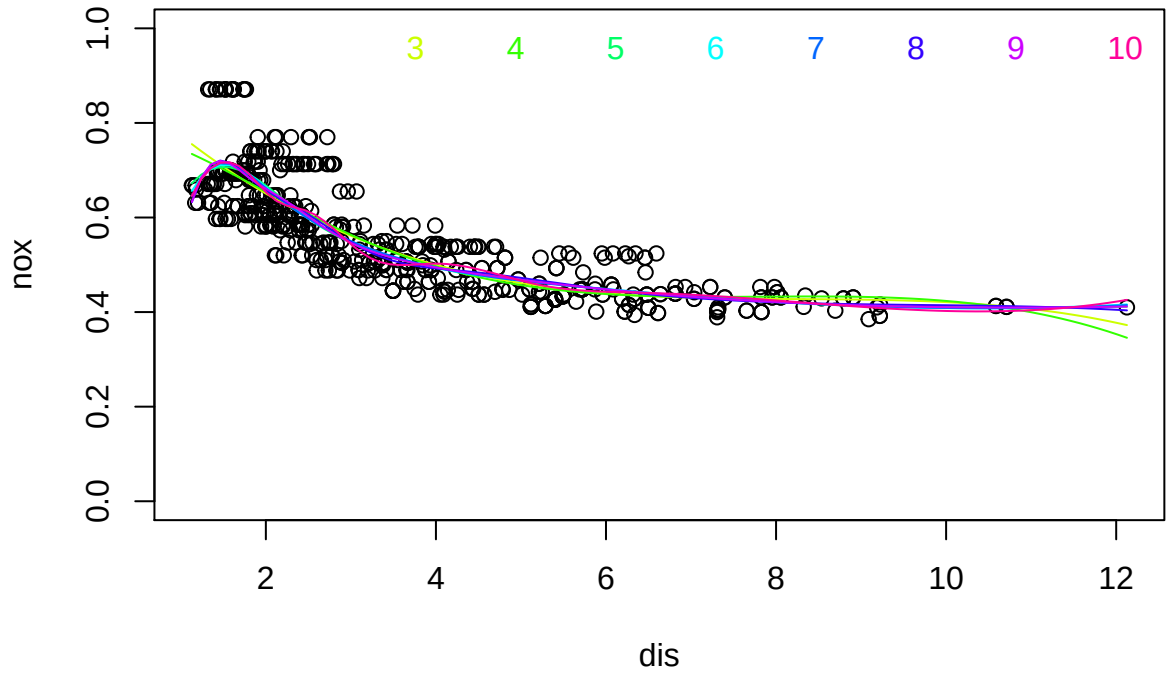
```
library(splines)
library(MASS)

spline.fit = lm(nox ~ bs(dis, df = 4), data = Boston)
x = seq(min(Boston[, "dis"]), max(Boston[, "dis"]), length.out = 100)
y = predict(spline.fit, data.frame(dis = x))

plot(Boston[, c("dis", "nox")], ylim = c(0, 1))
lines(x, y, col = cols[4])
```



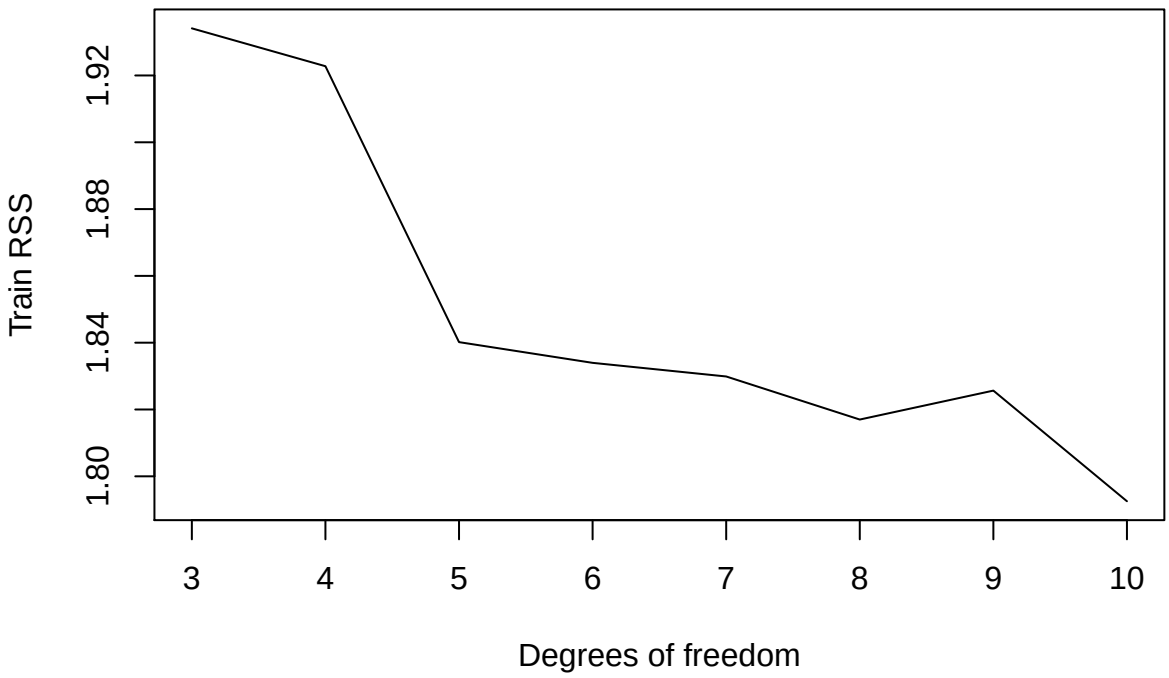
From the above plot, the polynomial with 4 degrees of freedom is shown above. I went ahead and used 100 knots just so the curve was as smooth as possible; using something like 5 knots makes it a lot jumpier/jagged. I figured 100 knots is a safe number to have in order to get the full power of the model while still making it look nice and ensuring it fits the data.

e)

```
plot(Boston[, c("dis", "nox")], ylim = c(0, 1))

x = seq(min(Boston[, "dis"]), max(Boston[, "dis"]), length.out = 100)

rss = c()
for(df in 3:10){
  spline.fit = lm(nox ~ bs(dis, df = df), data = Boston)
  y = predict(spline.fit, data.frame(dis = x))
  lines(x, y, col = cols[df])

  rss = c(rss, sum(spline.fit$residuals^2))
}

legend(x = "topright", legend = 3:10, text.col = cols[3:10], text.width = 0.5, bty = "n", horiz = T)
```



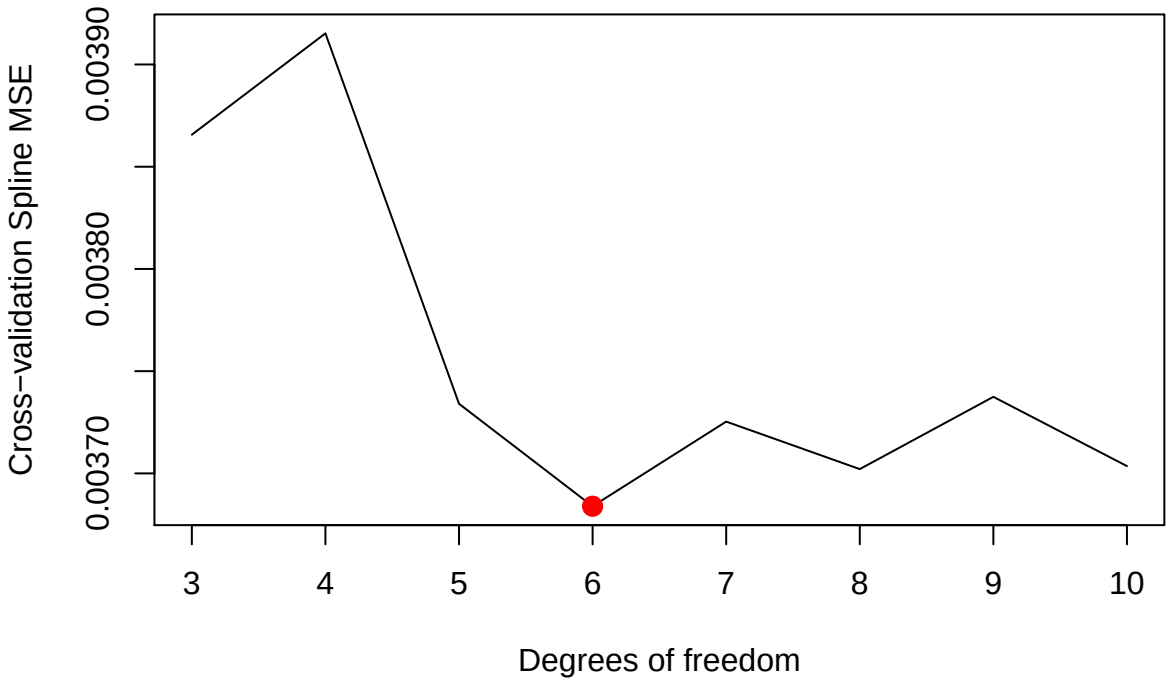

```
plot(3:10, rss, xlab = "Degrees of freedom", ylab = "Train RSS", type = "l")
```



The model with the lowest training RSS is the one with 10 degrees of freedom. However, it's important to note in the above graph the scale on the y-axis does not have a large range. As such, all models are very close to being the same; the one with 10 df is only marginally better.

f)

```
library(boot)

set.seed(1)
spline.mse = c()

for(df in 3:10){
  Boston.model = model.frame(nox ~ bs(dis, df = df), data = Boston)
  names(Boston.model) = c("nox", "bs.dis")

  spline.fit = glm(nox ~ bs.dis, data = Boston.model)
  mse = cv.glm(spline.fit, data = Boston.model, K = 10)$delta[1]
  spline.mse = c(spline.mse, mse)
}

plot(3:10, spline.mse, type = "l", xlab = "Degrees of freedom", ylab = "Cross-validation Spline MSE")

x = which.min(spline.mse)
points(x + 2, spline.mse[x], col = "red", pch = 20, cex = 2)
```

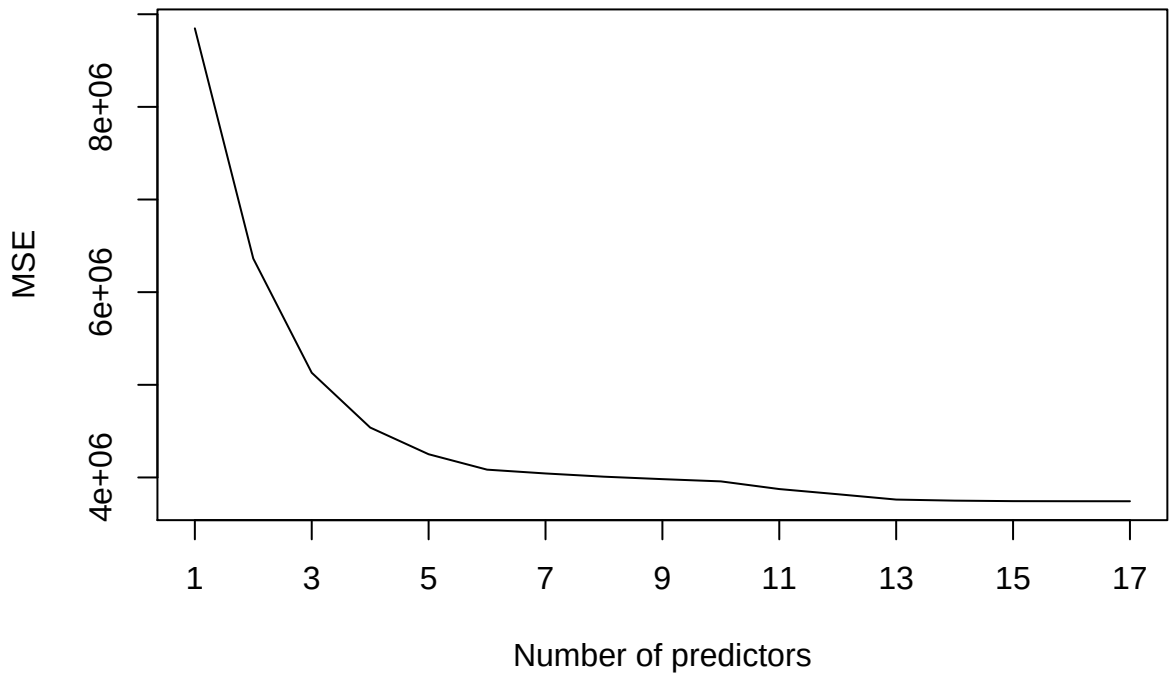


Clearly, from the above plot, the model with 6 degrees of freedom has the smallest MSE by a very small margin compared to the other models.

Chapter 7 - Exercise 10

a)

```
library(ISLR)
library(leaps)
set.seed(1)

train = sample(1:nrow(College), 500)
test = -train

forward = regsubsets(Outstate ~ ., data = College, method = "forward", nvmax = 17)

plot(1 / nrow(College) * summary(forward)$rss, type = "l", xlab = "Number of predictors", ylab = "MSE",
axis(side = 1, at = seq(1, 17, 2), labels = seq(1, 17, 2))
```




```
which(summary(forward)$which[7, -1])
```

```
##  PrivateYes  Room.Board    Personal         PhD perc.alumni      Expend
##           1           9          11          12          15          16
##   Grad.Rate
##          17
```

Clearly, we can see the above 7 predictors (Private, Room/Board, Personal, PhD, Percentage Alumni, Expenditure, and Graduation Rate) are the predictors we found to be the most useful.

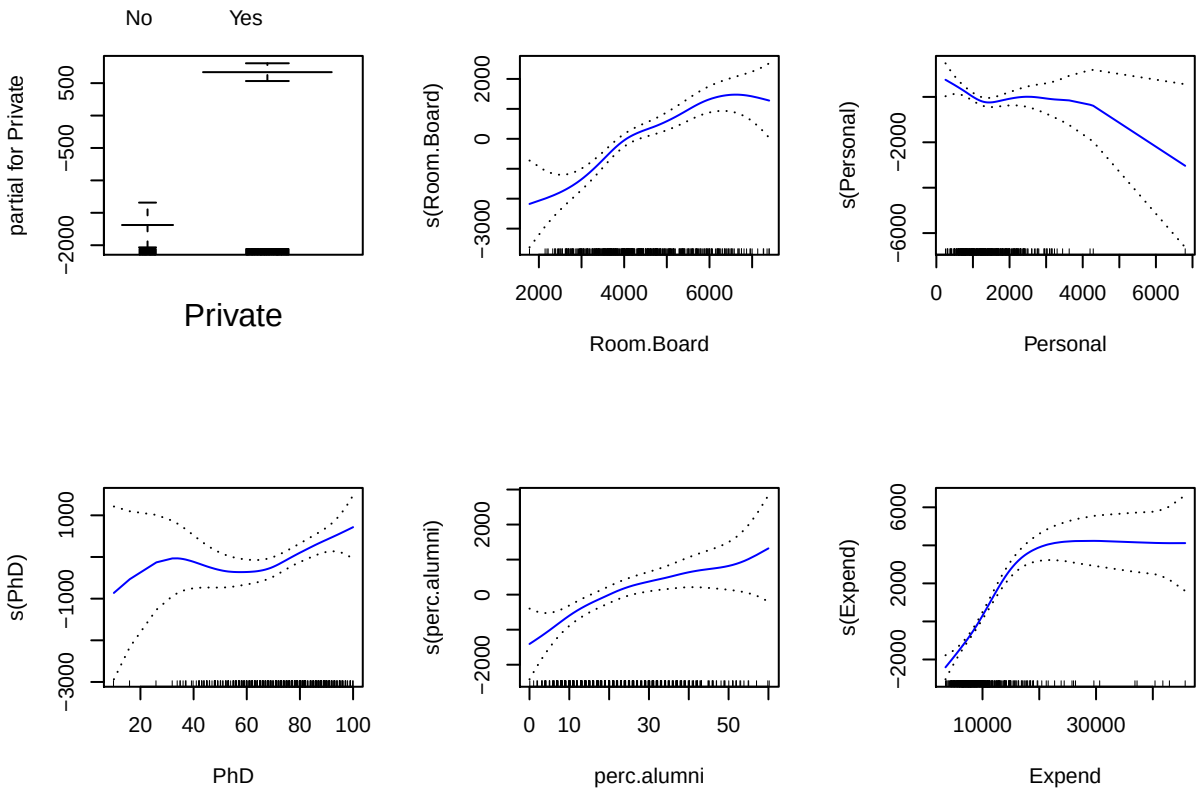
b)

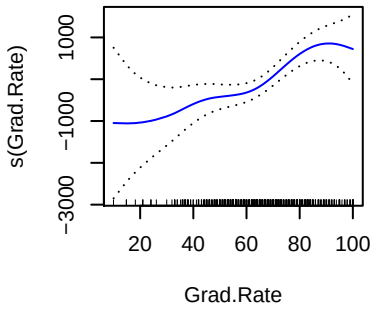
```
library(gam)
```

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

```
## Loaded gam 1.16
```

```
gam.fit = gam(Outstate ~ Private + s(Room.Board) + s(Personal) + s(PhD) + s(perc.alumni) + s(Expend) + s

par(mfrow = c(2, 3))
plot(gam.fit, se = T, col = "blue")
```


No
Yes
partial for Private
−2000
−500
500
Private
s(Room.Board)
−3000
0
2000
2000
4000
6000
Room.Board
s(Personal)
−6000
−2000
0
2000
4000
6000
Personal
s(PhD)
−3000
−1000
1000
20
40
60
80
100
PhD
s(perc.alumni)
−2000
0
2000
0
10
30
50
perc.alumni
s(Expend)
−2000
2000
6000
10000
30000
Expend



Unfortunately, Private is a qualitative predictor, so we couldn't fit a smooth spline onto it, but we did so with the other 6 predictors which were quantitative. It doesn't look like any of the splines are too complex, as in they don't look like they overfit. All of the models look different from each other, which is a good indicator that they all have different relationships with out-of-state tuition, reducing concerns with collinearity.

c)

```
gam.pred = predict(gam.fit, College[test, ])
gam.mse = mean((College[test, "Outstate"] - gam.pred)^2)
gam.mse
```

```
## [1] 3145951
```

```
gam.tss = mean((College[test, "Outstate"] - mean(College[test, "Outstate"]))^2)
test.rss = 1 - gam.mse / gam.tss
test.rss
```

```
## [1] 0.8026092
```

We can see the test MSE is lower than the training MSE, which shows the model performs better on the test set. As such, we don't have any concerns of overfitting. In addition, the correlation is about 0.8, so the model explains a good amount of variance in out-of-state tuition.

d)

```
summary(gam.fit)
```

```
##
## Call: gam(formula = Outstate ~ Private + s(Room.Board) + s(Personal) +
##     s(PhD) + s(perc.alumni) + s(Expend) + s(Grad.Rate), data = College[train,
```

```
##     ])
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -7771.52 -1116.73    86.96  1267.36  6790.30
##
## (Dispersion Parameter for gaussian family taken to be 3573089)
##
##     Null Deviance: 8144554092 on 499 degrees of freedom
## Residual Deviance: 1693643395 on 473.9998 degrees of freedom
## AIC: 8990.709
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                  Df     Sum Sq    Mean Sq F value    Pr(>F)
## Private           1 2238787959 2238787959 626.569 < 2.2e-16 ***
## s(Room.Board)     1 1532738194 1532738194 428.967 < 2.2e-16 ***
## s(Personal)       1   53394155   53394155  14.943 0.0001262 ***
## s(PhD)            1  455911313  455911313 127.596 < 2.2e-16 ***
## s(perc.alumni)    1  320845375  320845375  89.795 < 2.2e-16 ***
## s(Expend)         1  655584971  655584971 183.478 < 2.2e-16 ***
## s(Grad.Rate)      1   71951473   71951473  20.137 9.059e-06 ***
## Residuals       474 1693643395    3573089
## ---
## 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  4.1623   0.00631 **
## s(Personal)           3  3.4479   0.01661 *
## s(PhD)                3  2.0296   0.10889
## s(perc.alumni)        3  1.2215   0.30131
## s(Expend)             3 24.7343 6.883e-15 ***
## s(Grad.Rate)          3  2.4677   0.06144 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the above output, all the predictors are statistically significant, so we will assume all the predictors have a non-linear relationship with out-of-state tuition.

Question 6

```r
set.seed(1)

newsData <- read.table("OnlineNewsPopularity.csv", header=TRUE, sep=",")
train = sample(1:dim(newsData)[1], dim(newsData)[1]/2)
test <-- train
newsData.train = newsData[train, ]
newsData.test = newsData[test, ]

names(newsData)
```

```
##  [1] "url"                           "timedelta"
##  [3] "n_tokens_title"                "n_tokens_content"
##  [5] "n_unique_tokens"               "n_non_stop_words"
##  [7] "n_non_stop_unique_tokens"      "num_hrefs"
##  [9] "num_self_hrefs"                "num_imgs"
## [11] "num_videos"                    "average_token_length"
## [13] "num_keywords"                  "data_channel_is_lifestyle"
## [15] "data_channel_is_entertainment" "data_channel_is_bus"
## [17] "data_channel_is_socmed"        "data_channel_is_tech"
## [19] "data_channel_is_world"         "kw_min_min"
## [21] "kw_max_min"                    "kw_avg_min"
## [23] "kw_min_max"                    "kw_max_max"
## [25] "kw_avg_max"                    "kw_min_avg"
## [27] "kw_max_avg"                    "kw_avg_avg"
## [29] "self_reference_min_shares"     "self_reference_max_shares"
## [31] "self_reference_avg_sharess"    "weekday_is_monday"
## [33] "weekday_is_tuesday"            "weekday_is_wednesday"
## [35] "weekday_is_thursday"           "weekday_is_friday"
## [37] "weekday_is_saturday"           "weekday_is_sunday"
## [39] "is_weekend"                    "LDA_00"
## [41] "LDA_01"                        "LDA_02"
## [43] "LDA_03"                        "LDA_04"
## [45] "global_subjectivity"           "global_sentiment_polarity"
## [47] "global_rate_positive_words"    "global_rate_negative_words"
## [49] "rate_positive_words"           "rate_negative_words"
## [51] "avg_positive_polarity"         "min_positive_polarity"
## [53] "max_positive_polarity"         "avg_negative_polarity"
## [55] "min_negative_polarity"         "max_negative_polarity"
## [57] "title_subjectivity"            "title_sentiment_polarity"
## [59] "abs_title_subjectivity"        "abs_title_sentiment_polarity"
## [61] "shares"
```

After doing applied exercises 9 and 10 on our project, results were shown for polynomial regression, splines, and GAMs. I did some prior investigative work (i.e. pairs with shares and each predictor variable), but decided not to show most of those plots since they all look fairly similar to the first plot with shares vs. num_videos, and there are 58 plots that look very similar to that; it'd be a waste of computational resources to make the rest of the plots if there aren't any significantly strong linear or polynomial relationships. As a note, R would not allow me to use all 59 predictors in our dataset because it "cannot allocate a vector of size 11.7 Gb", so I only picked a couple predictors we found useful in LASSO and previous investigations as a group to create a formula for these problems.

```
# Some basic polynomial regression test with a 5-th degree polynomial on num_videos
attach(newsData)
poly.fit = glm(shares ~ poly(num_videos, 5), data = newsData)
summary(poly.fit)
```

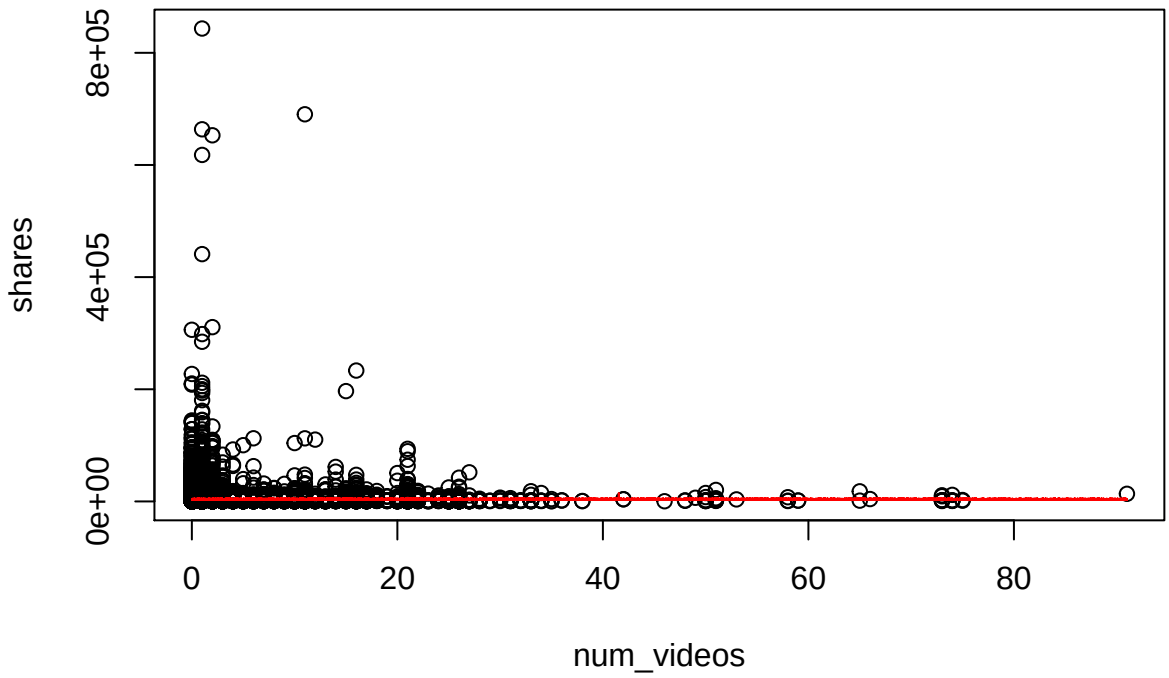
```
##
## Call:
## glm(formula = shares ~ poly(num_videos, 5), data = newsData)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
##  -5364   -2343   -1837    -537  839600
##
## Coefficients:
```

```
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              3395.38      58.33  58.214  < 2e-16 ***
## poly(num_videos, 5)1  55411.76   11613.19   4.771 1.84e-06 ***
## poly(num_videos, 5)2 -48859.54   11613.19  -4.207 2.59e-05 ***
## poly(num_videos, 5)3  72763.33   11613.19   6.266 3.75e-10 ***
## poly(num_videos, 5)4 -40645.26   11613.19  -3.500 0.000466 ***
## poly(num_videos, 5)5  30727.13   11613.19   2.646 0.008151 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 134866278)
##
##     Null deviance: 5.3592e+12  on 39643  degrees of freedom
## Residual deviance: 5.3458e+12  on 39638  degrees of freedom
## AIC: 854640
##
## Number of Fisher Scoring iterations: 2
```

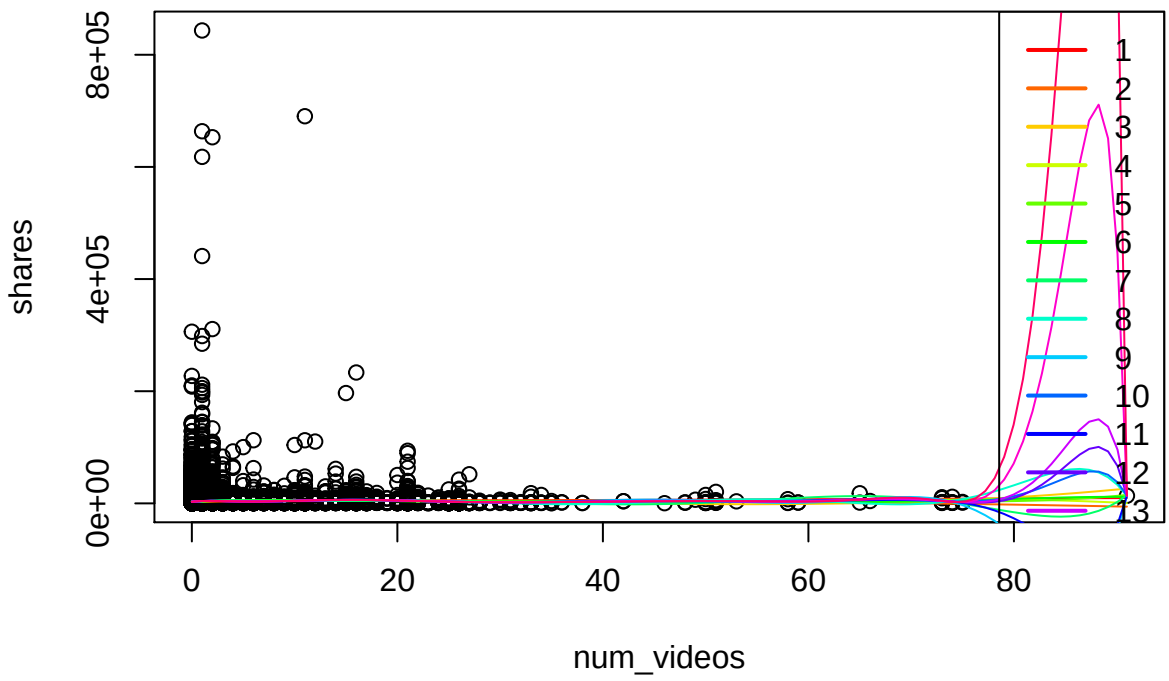
```
plot(newsData[, c('num_videos', 'shares')])
pred = predict(poly.fit, data.frame(dis=seq(min(num_videos), max(num_videos), length.out = length(num_v

cat(length(pred))
```

```
## 39644
```

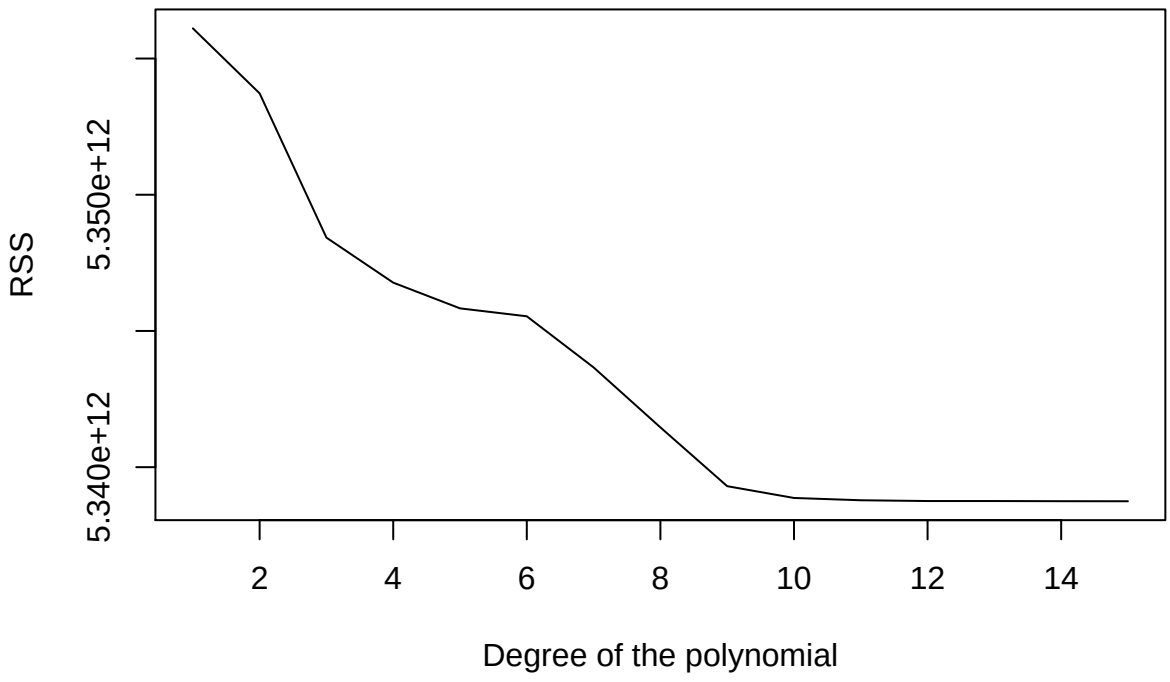
```
lines(seq(min(num_videos), max(num_videos), length.out = length(num_videos)), pred, col = "red")
```



```r
library(boot)
library(ISLR)
library(leaps)
library(gam)
set.seed(1)

# Polynomial regression without cross-validation
x = seq(min(num_videos), max(num_videos), length.out = 100)
cols = rainbow(15)
plot(newsData[, c('num_videos', 'shares')])
rss = c()

for(pwr in 1:15){
  poly.fit = glm(shares ~ poly(num_videos, pwr), data = newsData)
  pred = predict(poly.fit, data.frame(num_videos = x))
  lines(x, pred, col = cols[pwr])

  rss = c(rss, sum(poly.fit$residuals^2))
}

legend(x = 'topright', legend = 1:15, col = cols, lty = c(1, 1), lwd = c(2, 2))
```

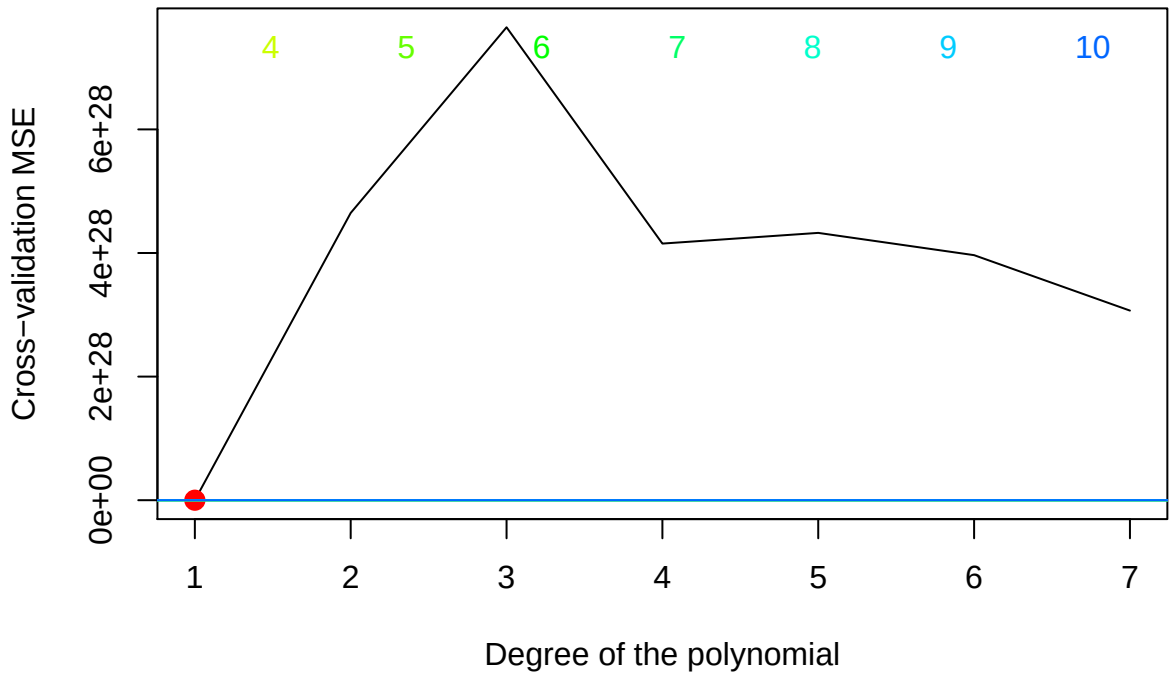



```r
plot(rss, xlab = "Degree of the polynomial", ylab = "RSS", type = "l")
```

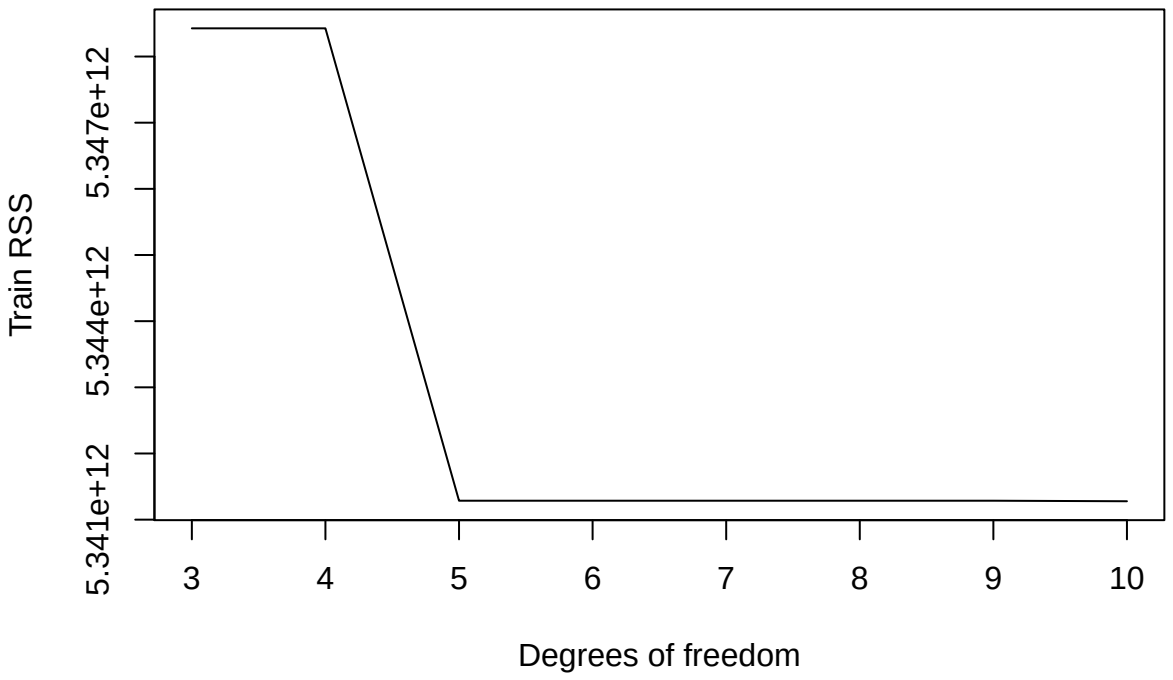



```
# Polynomial regression with cross-validation
poly.mse = c()
for(degree in 1:7){
  poly.fit = glm(shares ~ poly(num_videos, degree, raw = T) + poly(LDA_03, degree, raw = T) + poly(is_w
  mse = cv.glm(poly.fit, data = newsData, K = 10)$delta[1]
  poly.mse = c(poly.mse, mse)
}

plot(poly.mse, type = "l", xlab = "Degree of the polynomial", ylab = "Cross-validation MSE")
points(which.min(poly.mse), poly.mse[which.min(poly.mse)], col = "red", pch = 20, cex = 2)

# Splines without cross-validation and with RSS
df = 4
spline.fit = lm(shares ~ bs(num_videos, df = df), data = newsData)
x = seq(min(newsData[, "num_videos"]), max(newsData[, "num_videos"]), length.out = 100)
y = predict(spline.fit, data.frame(num_videos = x))
x = seq(min(newsData[, "num_videos"]), max(newsData[, "num_videos"]), length.out = 100)

rss = c()
for(df in 3:10){
  spline.fit = lm(shares ~ bs(num_videos, df = df), data = newsData)
  y = predict(spline.fit, data.frame(num_videos = x))
  lines(x, y, col = cols[df])

  rss = c(rss, sum(spline.fit$residuals^2))
}
```

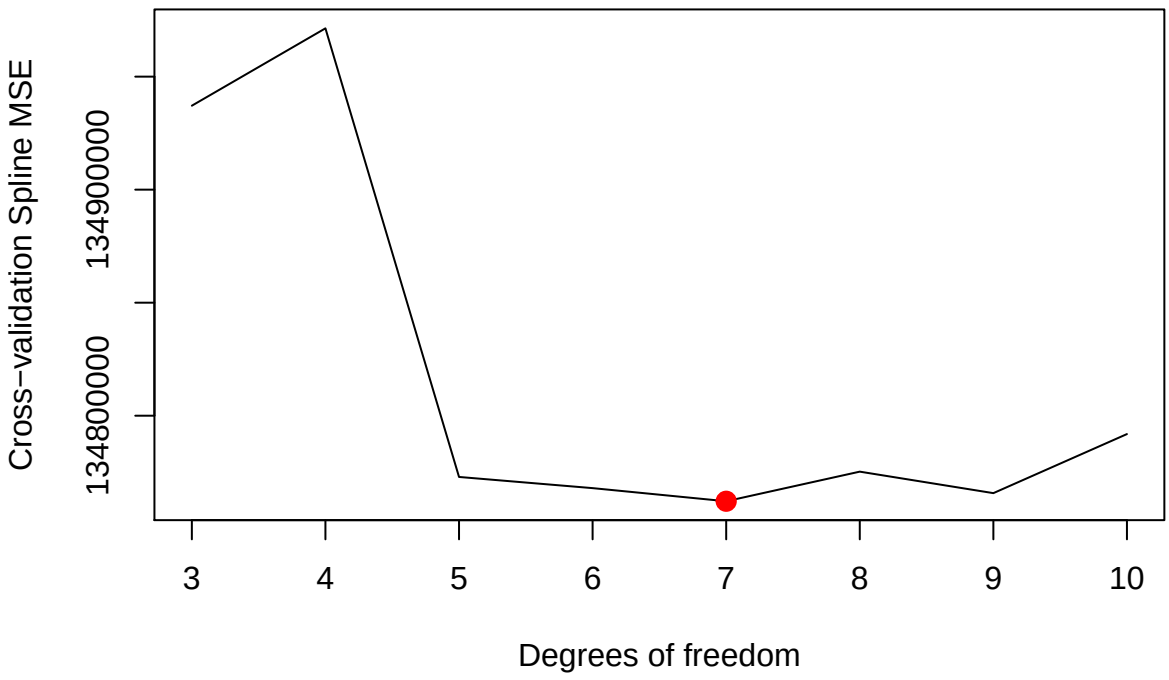
```
legend(x = "topright", legend = 3:10, text.col = cols[3:10], text.width = 0.5, bty = "n", horiz = T)
```




```
plot(3:10, rss, xlab = "Degrees of freedom", ylab = "Train RSS", type = "l")
```



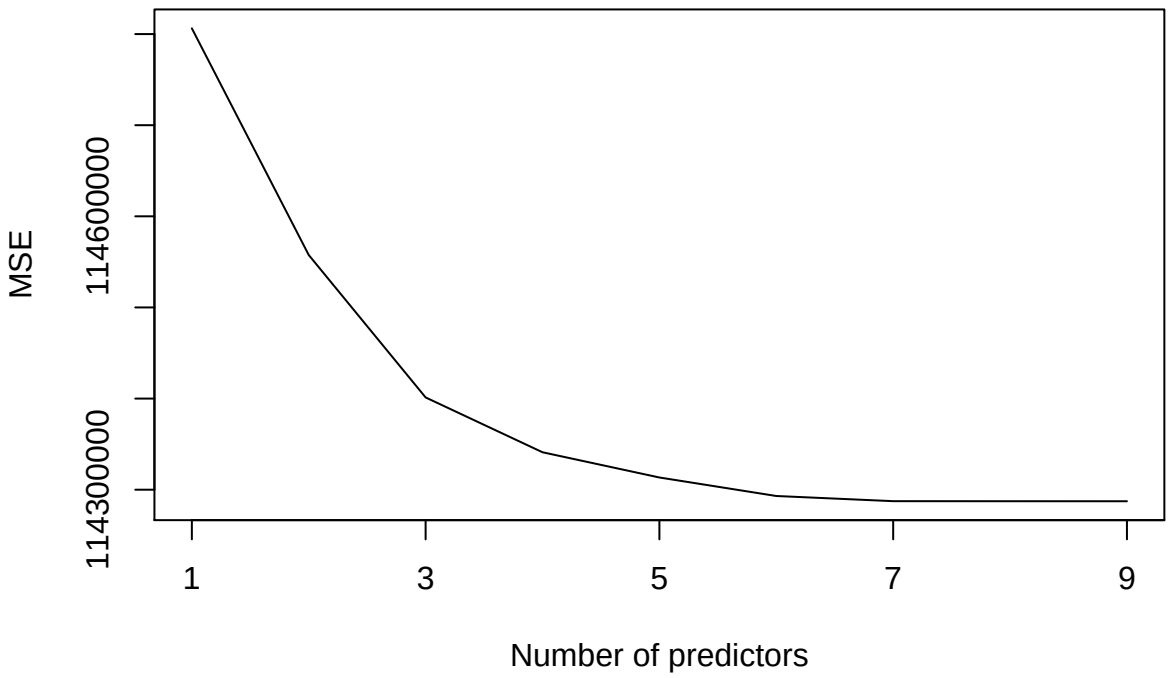

```
# Splines with cross-validation and with MSE
set.seed(1)
spline.mse = c()

# + LDA_03 + is_weekend + num_keywords + self_reference_max_shares + kw_max_avg + num_keywords + title_
for(df in 3:10){
  newsData.model = model.frame(shares ~ bs(num_videos, df = df), data = newsData)
  names(newsData.model) = c("shares", "bs.num_videos")

  spline.fit = glm(shares ~ bs.num_videos, data = newsData.model)
  mse = cv.glm(spline.fit, data = newsData.model, K = 10)$delta[1]
  spline.mse = c(spline.mse, mse)
}

plot(3:10, spline.mse, type = "l", xlab = "Degrees of freedom", ylab = "Cross-validation Spline MSE")

x = which.min(spline.mse)
points(x + 2, spline.mse[x], col = "red", pch = 20, cex = 2)
```




```
train = sample(1:nrow(newsData), 35000)
test = -train

forward = regsubsets(shares ~ LDA_03 + is_weekend + num_keywords + self_reference_max_shares + kw_max_a

which(summary(forward)$which[9, -1])
```

```
##                   LDA_03                is_weekend
##                        1                         2
##             num_keywords self_reference_max_shares
##                        3                         4
##               kw_max_avg  title_sentiment_polarity
##                        5                         6
##         n_non_stop_words                  num_imgs
##                        7                         8
##      rate_positive_words
##                        9
```

```
plot(1 / nrow(newsData) * summary(forward)$rss, type = "l", xlab = "Number of predictors", ylab = "MSE"
axis(side = 1, at = seq(1, 60, 2), labels = seq(1, 60, 2))
```



As we can see in the second graph, polynomial regression was tested with varying degrees. In the end, the polynomial with degree 14 produced the best RSS, but the degree 10 polynomial is better overall because it has nearly identical RSS while having a decent amount less model complexity. As such, polynomial regression results in a model with RSS of 5.34e+12.

Next, polynomial regression was tested with eleven variables, such as num_videos, LDA_03, is_weekend, num_keywords, and more. Some of these variables were chosen from our previous team LASSO discussions, while others were chosen for common sense of what would make most sense, and some were chosen for their correlation with the number of article shares. In the end, with eleven variables, the polynomial with degree 1 had the lowest cross-validation MSE by far compared to the higher degrees. Therefore, we can conclude this polynomial is probably best compared to other methods I have explored.

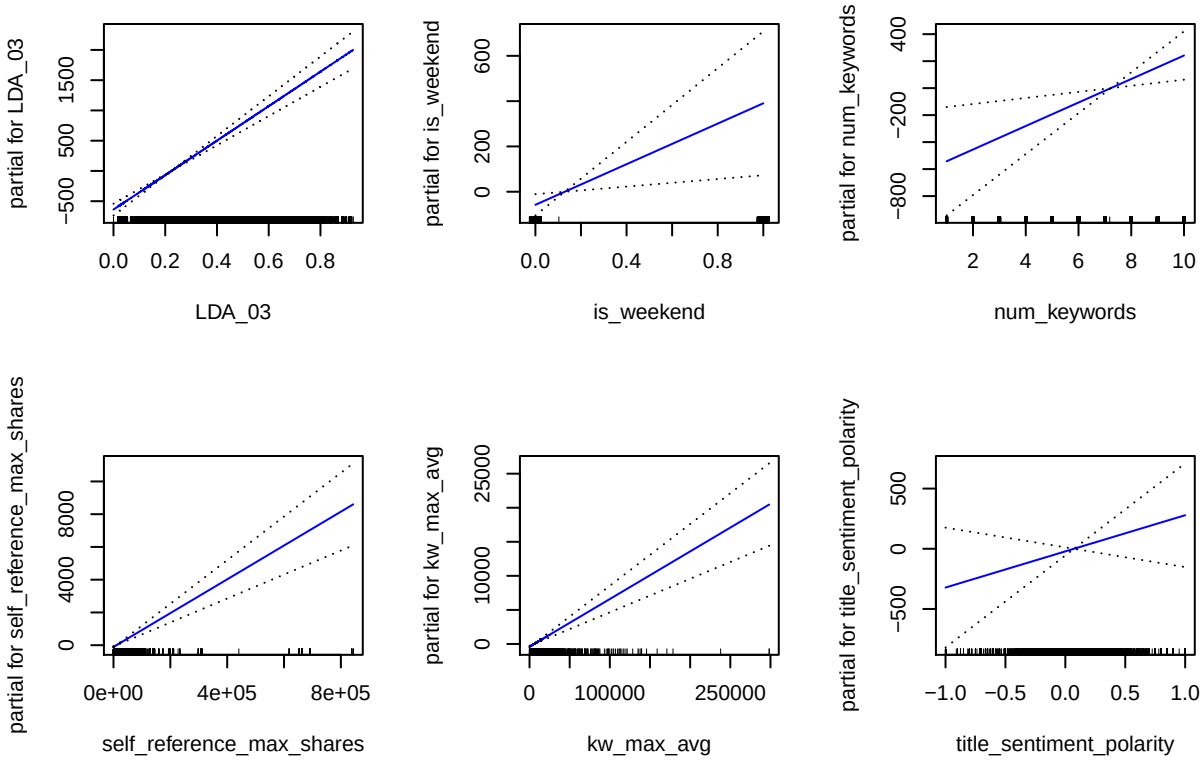
Finally, it is also important to note, polynomial regression was tested with a lot fewer variables, even though it wasn't shown in the above results. Three variables, num_videos and LDA_03, were used in comparison to the current 11. The results were actually surprisingly different. I found that the cross-validation MSE stays fairly constant as the degree of the polynomial increases, until it reaches degree 6. The degree 5 polynomial has the lowest MSE, but a polynomial with degree 2 produces similar MSE with significantly lower model complexity. As a result, the three polynomial regression tests produce differing degrees of a polynomial, some high and some low. I think it would be better to use the degree 1 polynomial for two reasons. First, we used cross-validation to reduce variance of results in error. Also, the polynomial is less complex, so more statistical inference can be done. Although progress has been made in this area, results could be somewhat specific to the features I chose. In the above examples, I used num_videos as the lone feature for initial polynomial regression, but I also used LDA_03 and kw_max_avg for the same test and they produced similar results so I figured it's probably safe to assume these results are standard across the dataset. Then, I used two variables for multiple polynomial regression, as well as eleven variables, so tests were pretty extensive. It would simply take me too long and it would be too annoying to create the graphs for all 59 predictors.

Next, splines were tested. In the fifth plot, the model with 6 degrees of freedom produced the lowest

cross-validation MSE. As such, 6 d.f. produces an even balance between being too complex and being too simple. The resulting MSE is 1.348e+8.

Finally, with forward selection, we found all 9 tested predictors as being useful.

```
# GAMs
gam.fit = gam(shares ~ LDA_03 + is_weekend + num_keywords + self_reference_max_shares + kw_max_avg + nu

par(mfrow = c(2, 3))
plot(gam.fit, se = T, col = "blue")
```




```
gam.pred = predict(gam.fit, College[test, ])
gam.mse = mean((newsData[test, "shares"] - gam.pred)^2)
gam.mse
```

```
## [1] 174170753
```

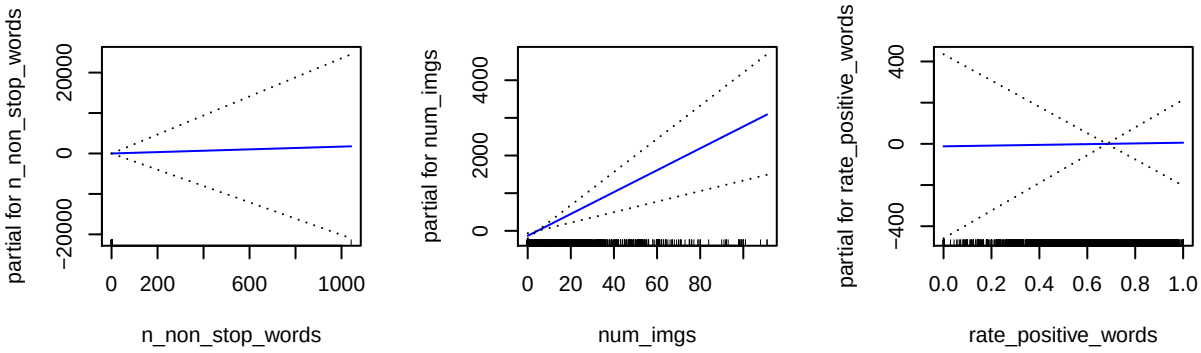
```
gam.tss = mean((newsData[test, "shares"] - mean(College[test, "shares"]))^2)
test.rss = 1 - gam.mse / gam.tss
test.rss
```

```
## [1] NA
```

```
summary(gam.fit)
```

```
##
## Call: gam(formula = shares ~ LDA_03 + is_weekend + num_keywords + self_reference_max_shares +
##     kw_max_avg + num_keywords + title_sentiment_polarity + n_non_stop_words +
##     num_imgs + rate_positive_words, data = newsData[train, ])
```

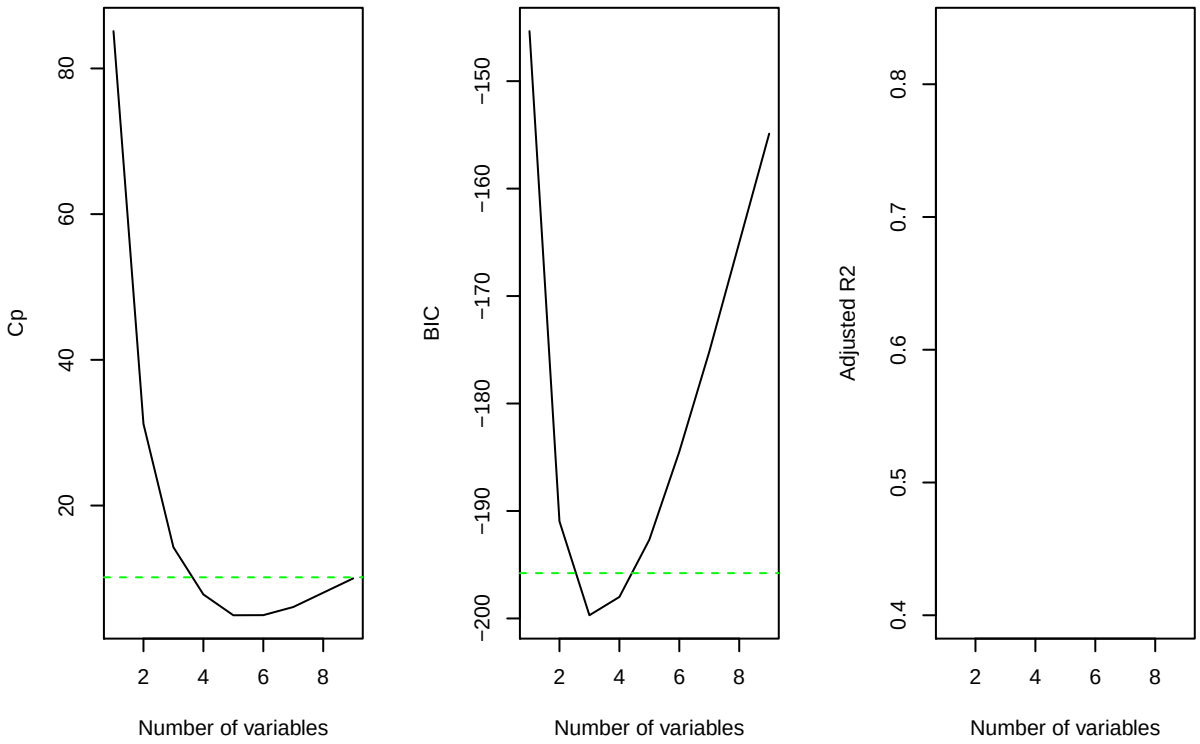
```
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -27633.2  -2181.1  -1458.3   -359.4 838242.1
##
## (Dispersion Parameter for gaussian family taken to be 129488626)
##
##     Null Deviance: 4.586421e+12 on 34999 degrees of freedom
## Residual Deviance: 4.530807e+12 on 34990 degrees of freedom
## AIC: 753106.3
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                               Df     Sum Sq    Mean Sq  F value    Pr(>F)
## LDA_03                         1 3.5041e+10 3.5041e+10 270.6138 < 2.2e-16
## is_weekend                     1 1.1382e+09 1.1382e+09   8.7902 0.0030306
## num_keywords                   1 1.9699e+09 1.9699e+09  15.2126 9.624e-05
## self_reference_max_shares      1 9.2328e+09 9.2328e+09  71.3016 < 2.2e-16
## kw_max_avg                     1 6.0145e+09 6.0145e+09  46.4485 9.558e-12
## title_sentiment_polarity       1 2.7528e+08 2.7528e+08   2.1259 0.1448382
## n_non_stop_words               1 1.0477e+07 1.0477e+07   0.0809 0.7760665
## num_imgs                       1 1.9315e+09 1.9315e+09  14.9162 0.0001126
## rate_positive_words            1 3.7267e+05 3.7267e+05   0.0029 0.9572168
## Residuals                  34990 4.5308e+12 1.2949e+08
##
## LDA_03                     ***
## is_weekend                 **
## num_keywords               ***
## self_reference_max_shares ***
## kw_max_avg                 ***
## title_sentiment_polarity
## n_non_stop_words
## num_imgs                   ***
## rate_positive_words
## Residuals
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



With the GAMs, we don't have any notable results across all 9 predictors. As mentioned earlier in this exercise, most of these graphs have a very weak correlation with number of article shares. We can clearly see from the model that n_non_stop_words is not significant, as well as title_sentiment_polarity, and rate_positive_words. Therefore, if we removed these three predictors, we would have a reduced model with 6 predictors, similar to the polynomial test earlier where we found degree 5 polynomials working with a large number of predictors or the spline determing 6 d.f. being the best model. I don't think I agree with continuing with the use of a GAM since it only marginally reduces model complexity. For future statistical inference, if we want to figure out which variables can potentially be important in predicting future improvements for a given news article, we want to stay on the conservative side and include variables even if we have an inkling of them being useful, such as with the degree 10 polynomial or 9 predictors found in forward selection.

```
# Cp, BIC, Adjusted R2 measurements
train <- sample(length(self_reference_max_shares), length(self_reference_max_shares) - (length(self_refe
test <- -train
newsData.train <- newsData[train, ]
newsData.test <- newsData[test, ]

fit <- regsubsets(shares ~ LDA_03 + is_weekend + num_keywords + self_reference_max_shares + kw_max_avg +
fit.summary <- summary(fit)
par(mfrow = c(1, 3))
plot(fit.summary$cp, xlab = "Number of variables", ylab = "Cp", type = "l")

min.cp <- min(fit.summary$cp)
std.cp <- sd(fit.summary$cp)
abline(h = min.cp + 0.2 * std.cp, col = "green", lty = 2)
abline(h = min.cp - 0.2 * std.cp, col = "green", lty = 2)
plot(fit.summary$bic, xlab = "Number of variables", ylab = "BIC", type='l')
```

```
min.bic <- min(fit.summary$bic)
std.bic <- sd(fit.summary$bic)
abline(h = min.bic + 0.2 * std.bic, col = "green", lty = 2)
abline(h = min.bic - 0.2 * std.bic, col = "green", lty = 2)
plot(fit.summary$adjr2, xlab = "Number of variables", ylab = "Adjusted R2", type = "l", ylim = c(0.4, 0

max.adjr2 <- max(fit.summary$adjr2)
std.adjr2 <- sd(fit.summary$adjr2)
abline(h = max.adjr2 + 0.2 * std.adjr2, col = "green", lty = 2)
abline(h = max.adjr2 - 0.2 * std.adjr2, col = "green", lty = 2)
```



In general, these results don't differ significantly from previous homework problems and tests. In previous investigations, LASSO found four useful features, such as rate_positive_polarity and num_images, which were both used in the above models with splines, GAMs and polynomial regression. As such, there are not any revelations in these results and we don't have an amazing MSE or RSS in any of the models, which is to be expected since none of the relationships between predictors and the response are directly linear or polynomial in nature. As such, it is extremely difficult to model this problem. With that being said, the model with 10 or so predictors provides for a lot of flexibility while also provind significant predictors and possible improvements for RSS/MSE. After evaluating BIC, C_p, and R^2, we can say somewhere around 5 or 6 variables produces the best values. As such, somewhere between the range of 6-9 predictors produces good results in terms of MSE, RSS, BIC, and across different models.

In the future, since we have a good estimate of the number of predictors that produces good error rates, I plan to do a bit more feature selection since we can now choose somewhere around a couple million different models rather than 59^{59} different models, and that number can be reduced even further since we have recognized useful variables with LASSO and forward selection. As such, this investigative work with applied exercises

9 and 10 haven't produced substantially better models, but have provided useful insights into how many predictors are appropriate to solve our problem and optimize for both error and model complexity, which turns out to be around 6 predictors. This remaining feature selection will be a point of future investigation for both me and our team in the remaining weeks.