BMishra Hw7

Bijesh Mishra

```r
rm(list = ls())
library(ISLR, warn.conflicts = F)
library(splines, warn.conflicts = F)
library(foreach, warn.conflicts = F)
library(gam, warn.conflicts = F)
```

```
## Loaded gam 1.16.1
```

```r
data("Auto")
attach(Auto, warn.conflicts = F)
# library(help = "splines")
```

Q1: KNOTS AND DEGREE

```r
table.q1a = matrix(0, 4, 6)
for (i in 1:4) {
  for (j in 1:6) {
    disp.q1 = bs(displacement, # Variable
           degree = i, # Degree of piecewise polynomial.
           knots = quantile(displacement, 1 : j/(j + 1))) # Knots
    fit.q1 = lm(mpg ~ disp.q1)
    table.q1a[i,j] = round(summary(fit.q1)$adj.r.squared, 5)
  }
}
which.max(table.q1a) # Maximum adjusted R-Squared value position in table
```

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

```r
table.q1a[3,5] # Maximum adjusted R-Squared value is from 3 degree polynomial with 5 knots.
```

```
## [1] 0.71576
```

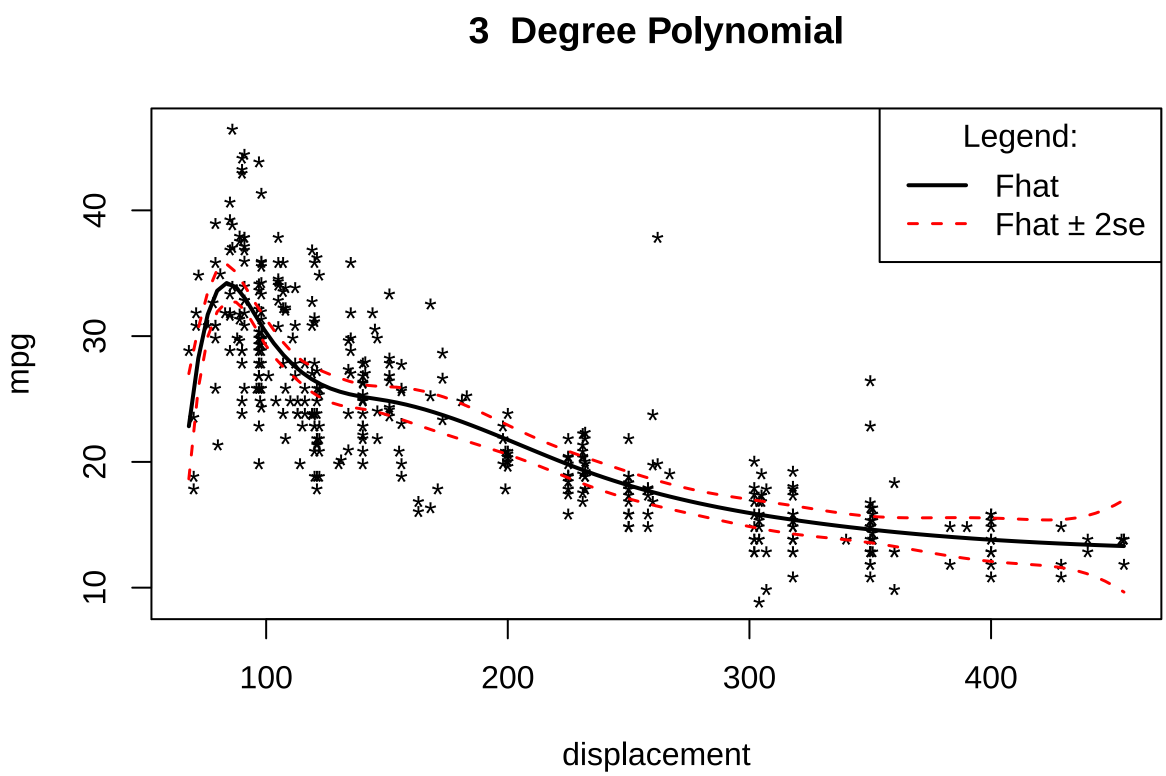
```r
table.q1a = as.data.frame(table.q1a,
                          row.names = c("Degree 1", "Degree 2",
                                        "Degree 3", "Degree 4"))
colnames(table.q1a) = c("Knot 1", "Knot 2", "Knot 3",
                        "Knot 4", "Knot 5", "Knot 6")
print(table.q1a)
```

```
##            Knot 1  Knot 2  Knot 3  Knot 4  Knot 5  Knot 6
## Degree 1 0.68109 0.68845 0.68648 0.69860 0.69929 0.69319
## Degree 2 0.68744 0.68757 0.70160 0.71394 0.71447 0.71435
## Degree 3 0.68676 0.69769 0.71467 0.71543 0.71576 0.71568
## Degree 4 0.69185 0.70824 0.71569 0.71268 0.71526 0.71368
```

The model with highest Adj-R^2 value is considered as the best model. From above table, model with 5 knots and 3 degree has Adj-R^2 value = 0.71576 which is highest among all reported Adj-R^2. So, I am going to use model with 5 knots and 3^{rd} degree polynomial in displacement.

Q2:

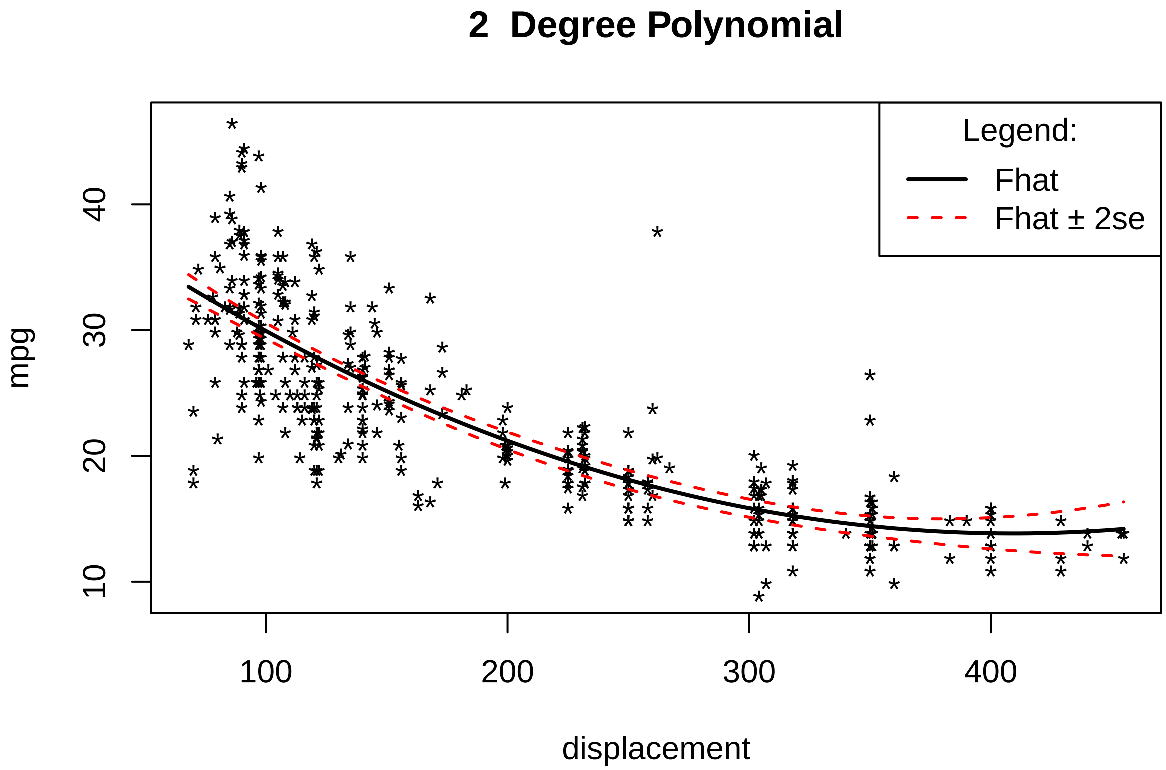
```
fit.q2 = lm(mpg ~ bs(displacement,
                     degree = 3,
                     knots = quantile(displacement,
                                      1 : 5/(5 + 1))))
disp.grid.q2 = seq(from = min(displacement),
                to = max(displacement),
                length = 100)
pred.q2 = predict(fit.q2,
                  newdata = list(displacement = disp.grid.q2),
                  se = TRUE)
plot(displacement, mpg, pch = "*")
lines(disp.grid.q2,
      pred.q2$fit,
      lty = 1,
      col = 1,
      lwd = 2) # fhatt
lines(disp.grid.q2,
      pred.q2$fit + 2*pred.q2$se.fit,
      lty = 2,
      col = 2,
      lwd = 1.5) # fhatt + 2se
lines(disp.grid.q2,
      pred.q2$fit - 2*pred.q2$se.fit,
      lty = 2,
      col = 2,
      lwd = 1.5) # fhatt - 2se.
legend("topright",
       col = c(1, 2),
       lwd = c(2, 1.5),
       lty = c(1, 2),
       legend = c("Fhat","Fhat ± 2se"),
       title = ("Legend:"))
title(paste(3, " Degree Polynomial"))
```



Q3:

```
fit.q3 = lm(mpg ~ bs(displacement,
                     degree = 2)) # d = 2. and k = 0.
disp.grid.q3 = seq(from = min(displacement),
                   to = max(displacement),
                   length = 100)
pred.q3 = predict(fit.q3,
                  newdata = list(displacement = disp.grid.q3),
                  se = TRUE)
plot(displacement, mpg, pch = "*")
lines(disp.grid.q3,
      pred.q3$fit,
      lty = 1,
      col = 1,
      lwd = 2) # Fhatt
lines(disp.grid.q3,
      pred.q3$fit + 2*pred.q3$se.fit,
      lty = 2,
      col = 2,
      lwd = 1.5) # Fhatt + 2se
lines(disp.grid.q3,
      pred.q3$fit - 2*pred.q3$se.fit,
      lty = 2,
      col = 2,
      lwd = 1.5) # Fhatt - 2se
legend("topright",
       col = c(1, 2),
       lty = c(1, 2),
       lwd = c(2, 1.5),
```

```r
       legend = c("Fhat","Fhat ± 2se"),
       title = ("Legend:"))
title(paste(2, " Degree Polynomial"))
```




```r
summary.q2 = summary(fit.q2)$adj.r.squared
summary.q3 = summary(fit.q3)$adj.r.squared
compare.models = as.data.frame(cbind(round(summary.q2,3),
                                     round(summary.q3,3)))
colnames(compare.models) = c("3 Degree 5 Knots", "3 Degree 0 Knots")
print(compare.models)
```

```
##   3 Degree 5 Knots 3 Degree 0 Knots
## 1            0.716            0.687
```

Even though the Adjusted R^2 of model with 3 Degree and 5 Knots is higher than that with 3 degree and 0 knots, they are very close to each other thus explaining about same amount of variation by both models. But Model with 3 degree and 0 knots is much simpler compared to Model with 3 degree and 5 knots. So, I would use model with 3 degree and 0 knots considering simplicity of model and ready to give up very small additional variation explained by model with 3 degree and 5 knots.

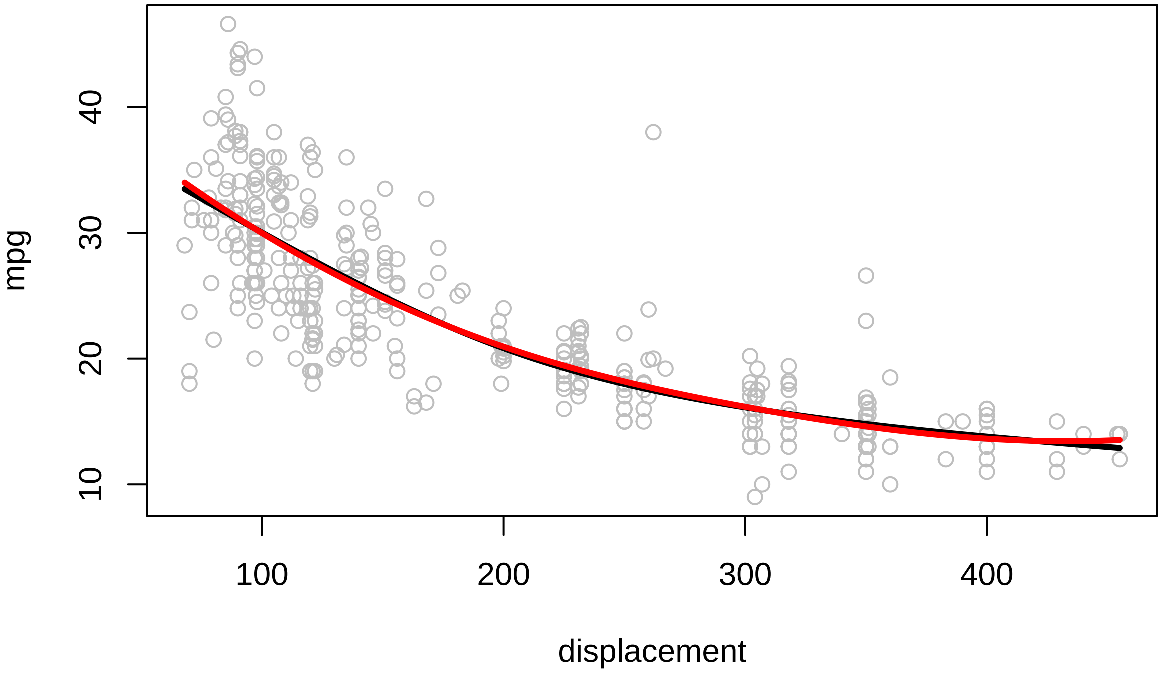
Q4: NATURAL AND BASIS SPLINES

```r
# Natural spline force linearity at the edge (& force unbroken line at Knots).
fit2 = lm(mpg ~ ns(displacement, knots = c(200, 300)))
summary(fit2)$df # gives number of parameters to estimate.
```

```
## [1]   4 388   4
```

```r
disp.grid = seq(from = min(displacement), max(displacement), length = 100)
preds2 = predict(fit2, newdata = list(displacement = disp.grid))
plot(displacement, mpg, col = "grey")
lines(disp.grid, preds2, type = "l", lwd = 3) # Black
```

```r
# Allow edge to swing by using basic spline (& force unbroken line at Knots).
fit3 = lm(mpg ~ bs(displacement, degree = 3, knots = c(200, 300)))
preds3 = predict(fit3, newdata = list(displacement = disp.grid))
lines(disp.grid, preds3, type = "l", lwd = 3, col = 2) # Red
```




```r
summary(fit3)$df # gives number of parameters to estimate.
```

```
## [1]   6 386   6
```

```r
# summary(fit2)
# summary(fit3)
```

Q4.A:

- For fit2 model (black line), Degree = 3 (Cubic spline) and Knots = 2 at 200 and 300.
- For fit3 model (red line), Degree = 3 (Third degree polynomial) and Knots = 2 at 200 and 300.

Q4B: bs does not force the edges of line in the chart to be linear and thus more flexible in the model fitting. However, ns forces edges of line in the chart to be linear and thus have less flexibility in model fitting. More flexible model might have low training error, high test error, thus less bias but more variance. However, less flexible model has less test error, less variance and more bias compared to flexible model. But to obtain more accurate prediction, I am willing to introduce some bias in the modeling process by reducing varianceSo, I may use fit2, since 500 is on the edge of the chart and bs might give unstable prediction for 500 due to its tail-swinging nature (reduce variance error).

Q4C: Natural spline (ns) force the edges of the curve on the figure to be linear thus limiting the swing which was observed in the basis spline (bs). This additional limitation is taking out degree of freedoms from the model and thus reducing the df of model with natural spline (ns). So, we are basically estimating additional two parameters by forcing linear conditions on the edges. However, this constraint is not imposed on model with basic spline (bs) which gives higher degree of freedom comapred to natural spline. Thus, I have to estimate fewer parameters in the ns compared to bs.

Q4D:

- In fit2 model, we have 4 parameters (intercept + 1 Degree + 2 Knots) to estimate.

- In fit3 we have 6 (intercept + 3 Degree + 2 Knots) parameters to estimate.

Q5: SMOOTH SPLINES

```
# a:
plot(displacement, mpg,
     col = 1,
     pch = "*",
     xlab = "Displacement",
     ylab = "MPG",
     ylim = c(5, 50),
     xlim = c(100, 500),
     main ="Smooth Splines")
spl1.q5 = smooth.spline(displacement,
                        mpg,
                        cv = TRUE) # LOOCV for estimating df or tuning parameter lambda.
```

```
## Warning in smooth.spline(displacement, mpg, cv = TRUE): cross-validation with
## non-unique 'x' values seems doubtful
```

```
spl1.q5$df
```

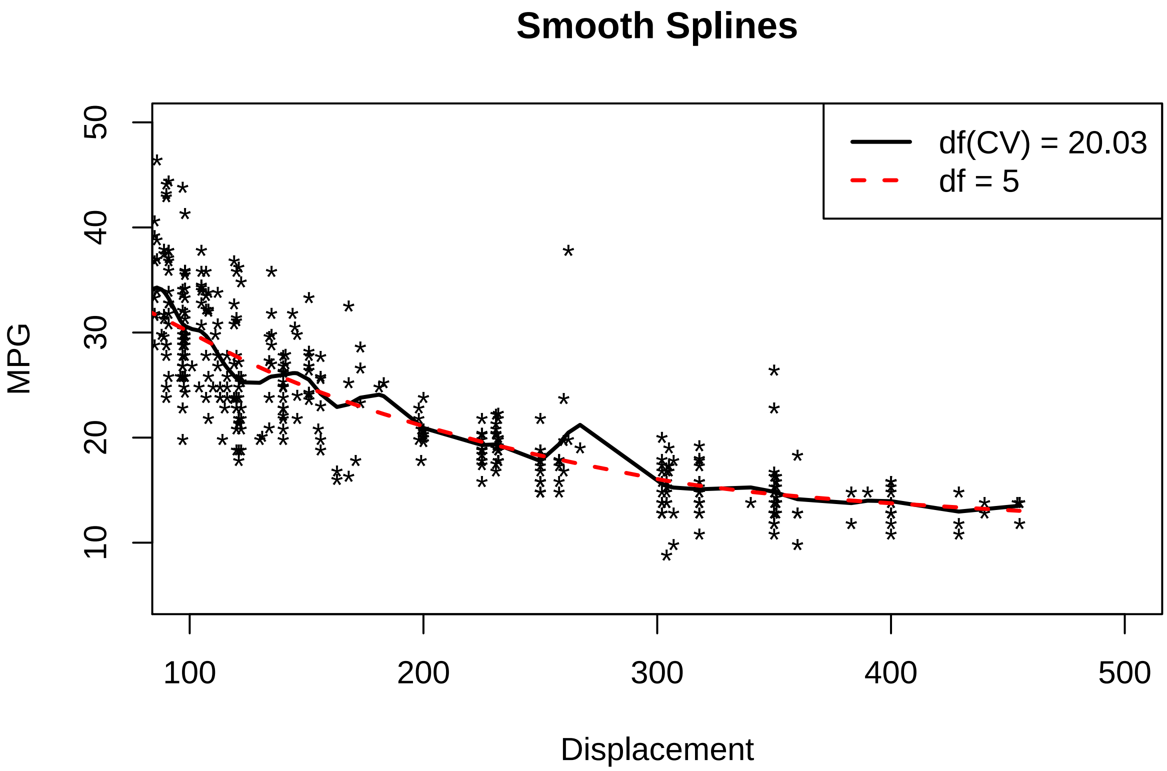
```
## [1] 20.0332
```

```
cat("Q5.A: The effective degree of freedom =", round(spl1.q5$df, 2))
```

```
## Q5.A: The effective degree of freedom = 20.03
```

```
# b:
lines(spl1.q5, lwd = 2, col = 1, lty = 1) # LOOCV (df)

# c:
spl2.q5  = smooth.spline(displacement, mpg, df = 5)
lines(spl2.q5, lwd = 2, col = 2, lty = 2) # df = 5
legend("topright",
       legend = c("df(CV) = 20.03", "df = 5"),
       lty = c(1, 2),
       lwd = c(2, 2),
       col = c(1, 2))
```



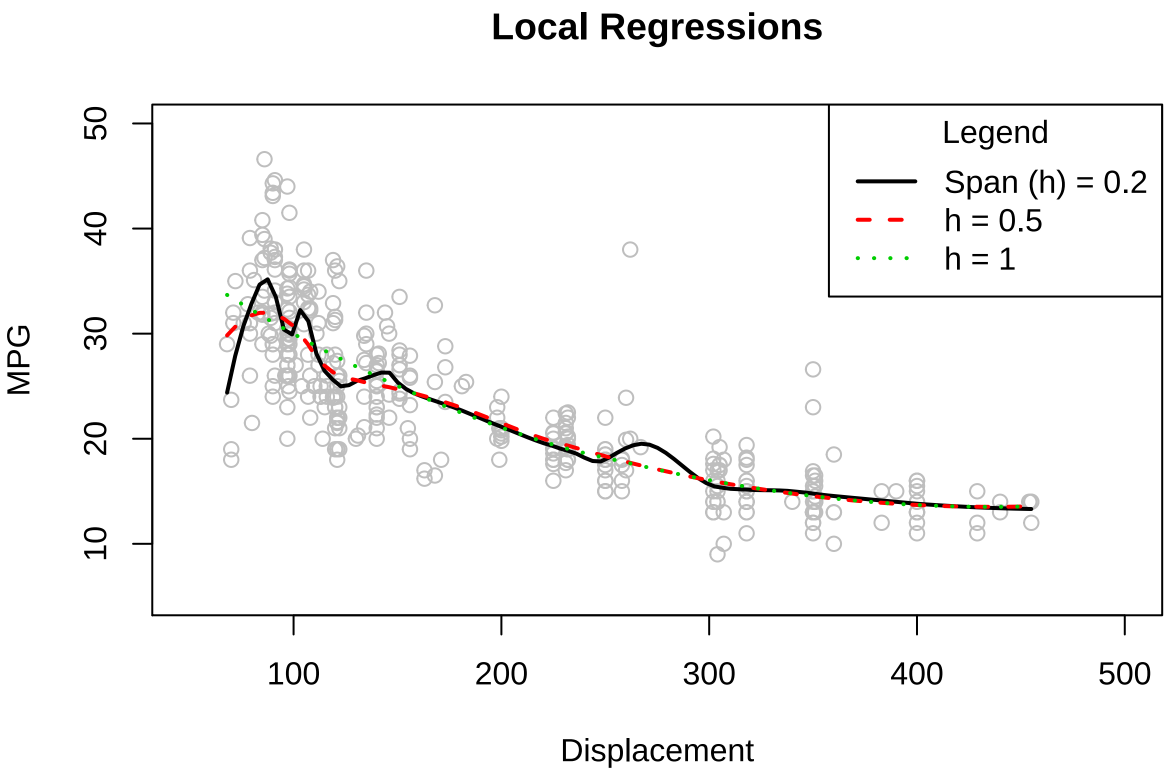
Q6: LOCAL REGRESSION:

```
plot(displacement, mpg,
     col = "grey", ylim = c(5, 50),
     xlim = c(50, 500),
     xlab = "Displacement",
     ylab = "MPG",
     main = "Local Regressions")
disp.grid.q6 = seq(from = min(displacement),
                   to = max(displacement),
                   length = 100)
# Fit 1:
q6.fit1 = loess(mpg ~ displacement,
                span = 0.2) # h
q6.fit1.pred = predict(q6.fit1,
                       data.frame(displacement = disp.grid.q6))
lines(disp.grid.q6,
      q6.fit1.pred,
      col = 1,
      lty = 1,
      lwd = 2)

# Fit 2:
q6.fit2 = loess(mpg ~ displacement,
                span = 0.5) # h
q6.fit2.pred = predict(q6.fit2,
                       data.frame(displacement = disp.grid.q6))
lines(disp.grid.q6,
      q6.fit2.pred,
      col = 2,
```

```
      lty = 2,
      lwd = 2)

# Fit 3:
q6.fit3 = loess(mpg ~ displacement,
                span = 1) # h
q6.fit3.pred = predict(q6.fit3,
                       data.frame(displacement = disp.grid.q6))
lines(disp.grid.q6,
      q6.fit3.pred,
      col = 3,
      lty = 3,
      lwd = 2)
legend("topright",
       legend = c("Span (h) = 0.2", "h = 0.5", "h = 1"),
       title = "Legend",
       lty = c(1, 2, 3),
       lwd = c(2, 2,2),
       col = c(1, 2, 3))
```



Q7: GENERALIZED ADDITIVE MODEL (P Predictors):

```
wowzers = lm(mpg ~
               s(displacement, 4) + # Smoothing splines fit in GAM Formula.
               ns(horsepower, 3) + # Natural Cobic spline.
               cylinders +
               bs(weight, 5)) # B-Spline Basis Polynomial splines.
summary(wowzers)
```

```
##
```

```
## Call:
## lm(formula = mpg ~ s(displacement, 4) + ns(horsepower, 3) + cylinders +
##     bs(weight, 5))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.5180  -2.1536  -0.2912   1.9528  15.6315
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          37.110170   2.684425  13.824  < 2e-16 ***
## s(displacement, 4)   -0.008835   0.009081  -0.973 0.331207
## ns(horsepower, 3)1   -9.171411   1.764140  -5.199 3.28e-07 ***
## ns(horsepower, 3)2 -19.917941   3.197853  -6.229 1.25e-09 ***
## ns(horsepower, 3)3   -8.516326   2.287845  -3.722 0.000227 ***
## cylinders            -0.034459   0.481303  -0.072 0.942962
## bs(weight, 5)1        2.325431   2.965562   0.784 0.433442
## bs(weight, 5)2       -0.360021   2.379800  -0.151 0.879833
## bs(weight, 5)3       -8.398701   3.371241  -2.491 0.013153 *
## bs(weight, 5)4       -6.667495   3.283203  -2.031 0.042970 *
## bs(weight, 5)5       -9.928814   3.675492  -2.701 0.007214 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.902 on 381 degrees of freedom
## Multiple R-squared:  0.7564, Adjusted R-squared:   0.75
## F-statistic: 118.3 on 10 and 381 DF,  p-value: < 2.2e-16
```

- Displacement: s = smooth spline, specifying a Smoothing spline fit in a GAM formula. In this case, we are applying smooth spline for displacement with 4th degree of freedom. The polynomial degree of displacement is one.
- Horsepower: ns = natural spline generates a basis matrix for natural cubic splines. The degree of freedom is 3 and Power = 3.
- Cylinders: Used as it is. Degree = 1.
- Weight: bs = basis spline generates B-spline basis matrix for a ploynomial spline. Degree of freedom = 5, The ploynomial degree of weight is 5.

```r
wowzers1 = lm(mpg ~
               s(displacement, 4) +
               ns(horsepower, 3) +
               weight)
summary(wowzers1)
```

```
##
## Call:
## lm(formula = mpg ~ s(displacement, 4) + ns(horsepower, 3) + weight)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.8998  -2.3076  -0.0932   1.8124  15.5703
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          4.582e+01  1.368e+00  33.502  < 2e-16 ***
```

```
## s(displacement, 4) -1.589e-02  6.252e-03  -2.542  0.01143 *
## ns(horsepower, 3)1 -9.353e+00  1.566e+00  -5.974 5.26e-09 ***
## ns(horsepower, 3)2 -1.925e+01  2.670e+00  -7.210 2.97e-12 ***
## ns(horsepower, 3)3 -5.759e+00  2.029e+00  -2.839  0.00477 **
## weight             -3.478e-03  7.056e-04  -4.930 1.23e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.926 on 386 degrees of freedom
## Multiple R-squared:  0.7502, Adjusted R-squared:  0.747
## F-statistic: 231.9 on 5 and 386 DF,  p-value: < 2.2e-16
```

Even though the Adjusted R^2 of first model (wowzers) is larger compared to second model (wowzers1), The Adjusted R^2 is not significantly large but are very close to each other thus explaining about same amount of variation by both models. In second model, I removed bs from weight which reduce variance (fluctuating tail) instead of using ns to impose linearity which also stabilize the fluctiating tail but increases number of parameters to estimate. In second GAM model (wowzers1) that I have less parameters to estimate compared to "wowzers" model. So, I would use model with smaller Adjusted R^2 considering simplicity of model and ready to give up very small additional variation explained by another model.