ISLR - Ch7 - Moving Beyond Linearity

Darshan Patel

2/13/2019

The following set of problems are from the applied exercises section in ISLR Chapter 7: Moving Beyond Linearity

```
rm(list = ls())
library(MASS)
library(ISLR)
library(tidyverse)
```

```
## Warning: package 'tibble' was built under R version 3.4.4
```

```
## Warning: package 'tidyr' was built under R version 3.4.4
```

```
## Warning: package 'purrr' was built under R version 3.4.4
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
library(gridExtra)
library(boot)
library(GGally)
```

```
## Warning: package 'GGally' was built under R version 3.4.4
```

```
## Warning: replacing previous import 'ggplot2::empty' by 'plyr::empty' when
## loading 'GGally'
```

```
library(gam)
```

```
## Warning: package 'gam' was built under R version 3.4.4
```

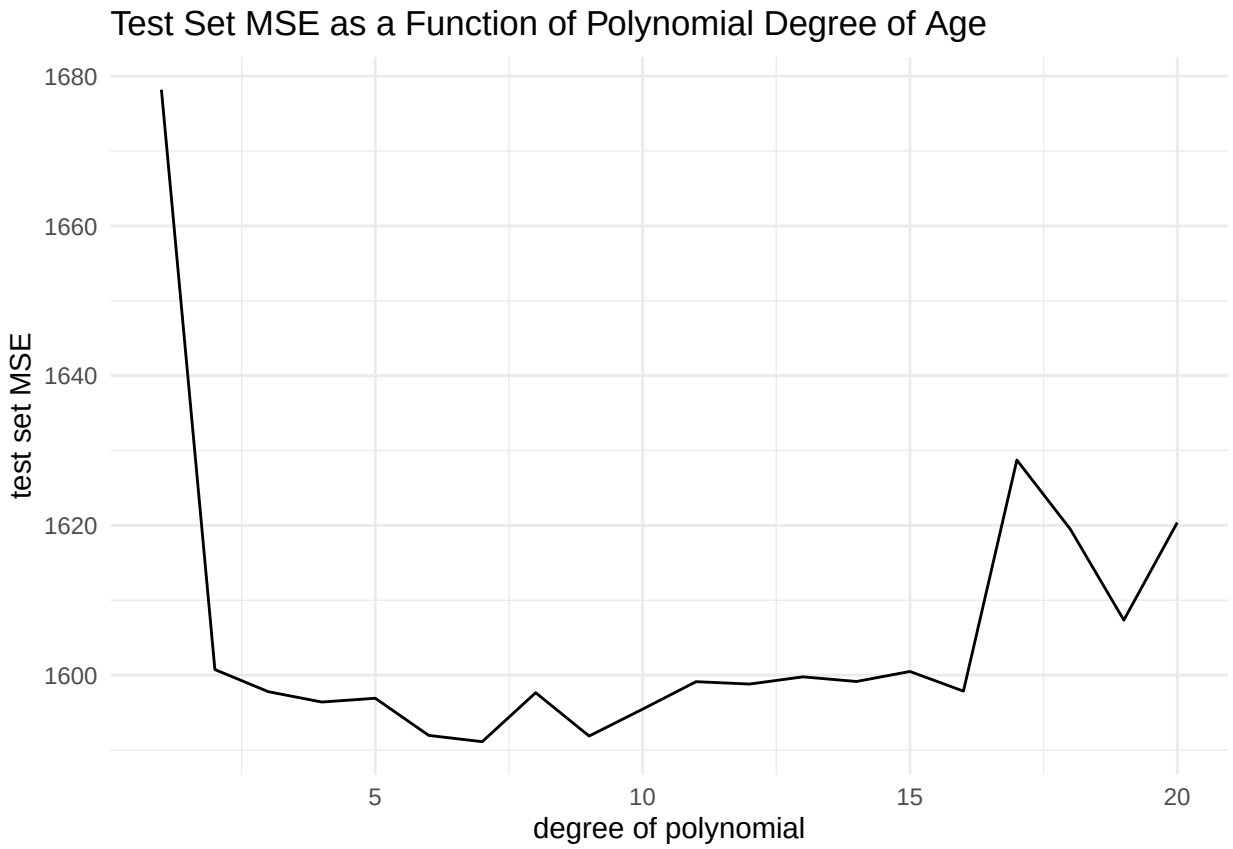
```
library(splines)
library(leaps)
```

Question 6: In this exercise, you will further analyze the `Wage` dataset considered throughout this chapter.

(a) Perform polynomial regression to predict `wage` using `age`. Use cross-validation to select the optimal degree d for the polynomial. What degree was chosen and how does this compare to the results of hypothesis testing using ANOVA? Make a plot of the resulting polynomial fit to the data.

```
df = Wage
set.seed(20)

mses = c()
for(i in 1:20){
  model = glm(data = df, wage ~ poly(age, i))
  mses = c(mses, cv.glm(df, model, K = 5)$delta[1])
}

ggplot(data.frame(mses), aes(x=1:20, y = mses)) + geom_path() +
  ggtitle("Test Set MSE as a Function of Polynomial Degree of Age") +
```

```
  labs(x = "degree of polynomial", y = "test set MSE") +
  theme_minimal()
```



From degree 2 to 16, the test set MSE hovers around a low value and then jump up.

```
which.min(mses)
```

```
## [1] 7
```

The optimal degree d for the polynomial is 7.

Compare this model to models of less degree using the ANOVA test.

```
model7 = lm(wage ~ poly(age, 7), data = df)
summary(model7)
```

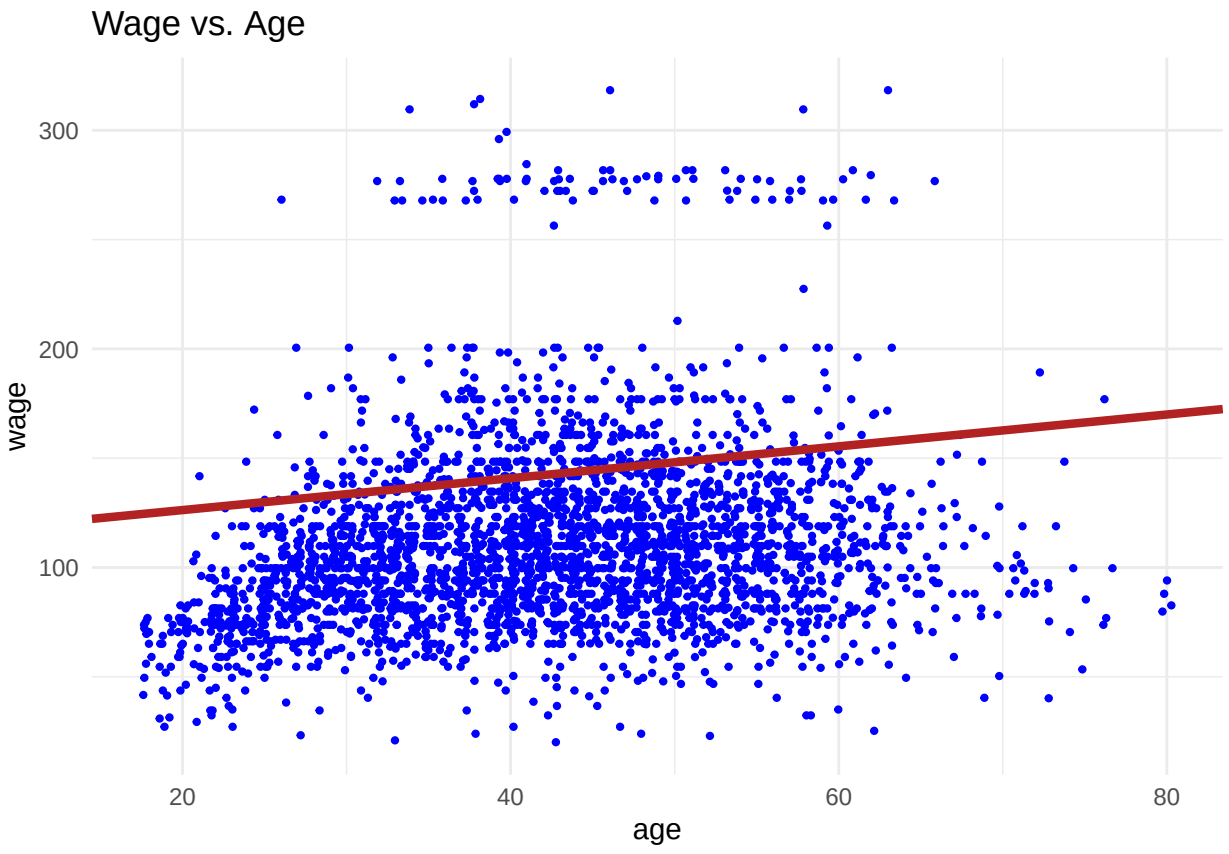
```
##
## Call:
## lm(formula = wage ~ poly(age, 7), data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -99.201 -24.393  -4.913  15.510 200.372
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    111.7036     0.7285 153.331  < 2e-16 ***
## poly(age, 7)1  447.0679    39.9023  11.204  < 2e-16 ***
## poly(age, 7)2 -478.3158    39.9023 -11.987  < 2e-16 ***
```

```
## poly(age, 7)3  125.5217    39.9023   3.146  0.00167 **
## poly(age, 7)4  -77.9112    39.9023  -1.953  0.05097 .
## poly(age, 7)5  -35.8129    39.9023  -0.898  0.36952
## poly(age, 7)6   62.7077    39.9023   1.572  0.11616
## poly(age, 7)7   50.5498    39.9023   1.267  0.20531
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.9 on 2992 degrees of freedom
## Multiple R-squared:  0.08775,    Adjusted R-squared:  0.08562
## F-statistic: 41.12 on 7 and 2992 DF,  p-value: < 2.2e-16
```

According to the ANOVA test, the p-value comparing a model of degree 1 to degree 2 is 0, meaning the linear fit is not significant. Moving along, it can be seen that comparing a model of degree 3 to degree 4 creates significance. Therefore a model with the polynomial degree of 3 is sufficient for a reasonable fit to the data.

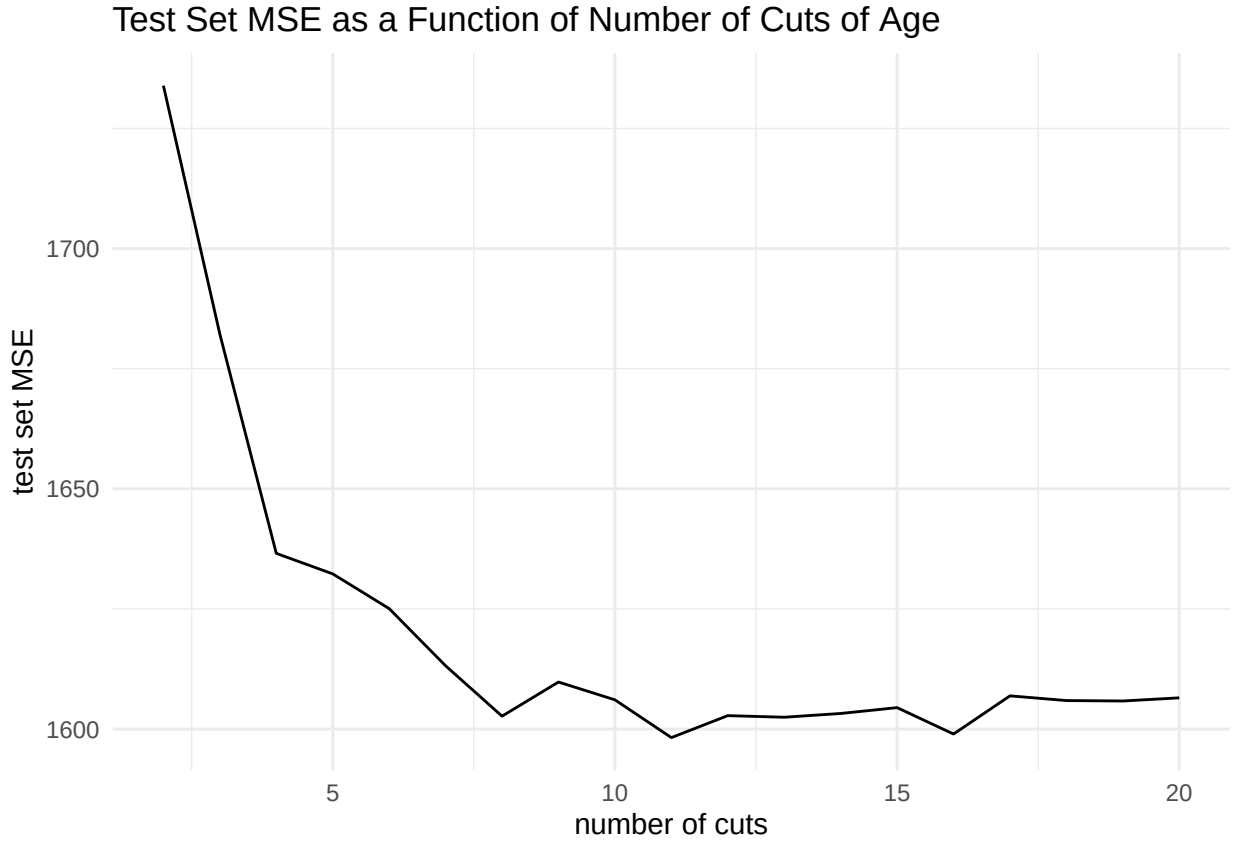
This is a plot of the fit on the dataset.

```
ggplot(df, aes(x = age, y = wage)) + geom_jitter(color = "blue", size = 0.8) +
  ggtitle("Wage vs. Age") +
  geom_abline(intercept = summary(model7)$coef[1,1],
              slope = summary(model7)$coef[1,2],
              color = "firebrick",
              size = 1.5) +
  theme_minimal()
```



(b) Fit a step function to predict `wage` using `age` and perform cross validation to choose the optimal number of cuts. Make a plot of the fit obtained.

```
mses = c()
for(j in 2:20){
  df$cut_age = cut(df$age, j)
  model = glm(data = df, wage ~ cut_age)
  mses = c(mses, cv.glm(df, model, K = 5)$delta[1])
}

ggplot(data.frame(mses), aes(x=2:20, y = mses)) + geom_path() +
  ggtitle("Test Set MSE as a Function of Number of Cuts of Age") +
  labs(x = "number of cuts", y = "test set MSE") +
  theme_minimal()
```



As more splits were made, the test set declined and hovered around the same range of error after 8 cuts.

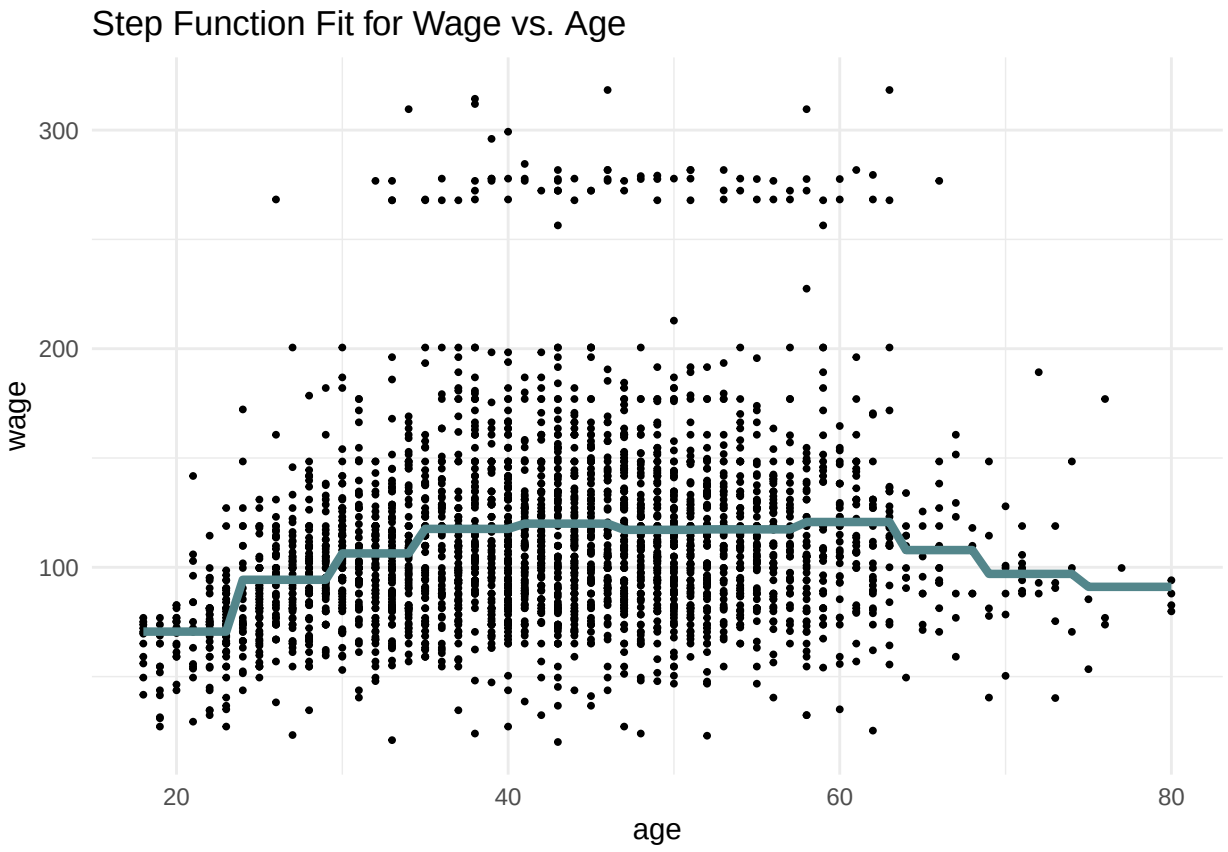
The optimal number of cuts is

```
which.min(mses) + 1
```

```
## [1] 11
```

This is the plot of the fit on top of the data.

```
model11 = lm(data = df, wage ~ cut(age, 11))

ggplot(df, aes(x = age, y = wage)) + geom_point(size = 0.7) +
  geom_line(aes(y = predict(model11)), color = "cadetblue4", size = 1.5) +
  ggtitle("Step Function Fit for Wage vs. Age") +
  theme_minimal()
```



Question 7: The `Wage` dataset contains a number of other features not explored in this chapter, such as marital status (`maritl`), job class (`jobclass`), and others. Explore the relationships between some of these other predictors and `wage` and use non-linear fitting techniques in order to fix flexible models to the data. Create plots of the results obtained, and write a summary of the findings.

```
df = Wage
```

The variables that will be looked at are `age`, `education`, `maritl` and `race`.

```
model1 = gam(wage ~ s(age, 3), data = df)
model2 = gam(wage ~ s(age, 3) + education, data = df)
model3 = gam(wage ~ s(age, 3) + education + maritl, data = df)
model4 = gam(wage ~ s(age, 3) + education + maritl + race, data = df)
anova(model1, model2, model3, model4)
```

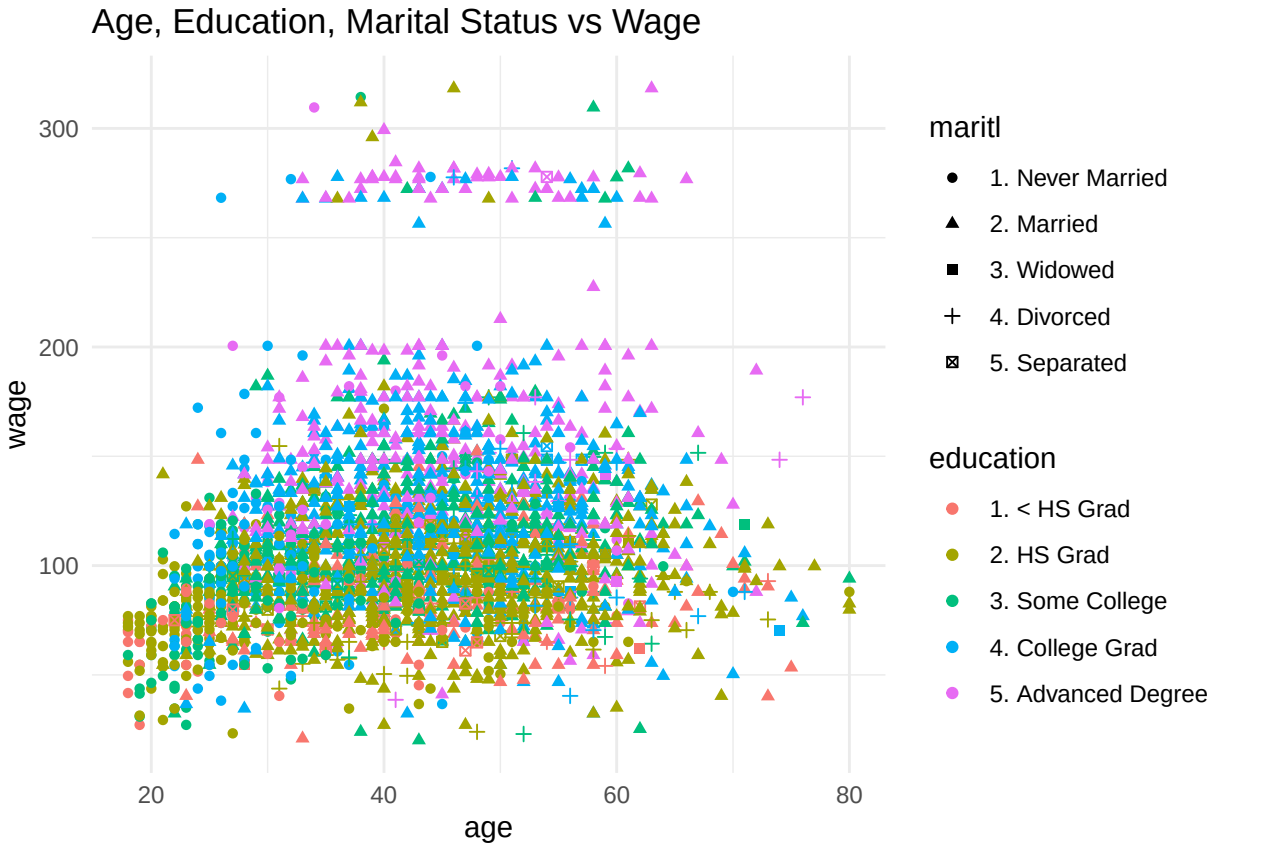
```
## Analysis of Deviance Table
##
## Model 1: wage ~ s(age, 3)
## Model 2: wage ~ s(age, 3) + education
## Model 3: wage ~ s(age, 3) + education + maritl
## Model 4: wage ~ s(age, 3) + education + maritl + race
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      2996    4781503
## 2      2992    3719674  4  1061829 < 2.2e-16 ***
## 3      2988    3624560  4    95114 3.615e-16 ***
```

```
## 4      2985    3616648  3     7912   0.08849 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

When using natural splines of `age` of degree 3, along with other variables: `education`, `maritl` and `race` as found that not all 4 variables will create the best model. When the `race` variable is added, the p-value associated with adding it to the model is 0.08. This means that to estimate `wage`, a natural spline of degree 3 for `age`, `education` and `maritl` is sufficient.

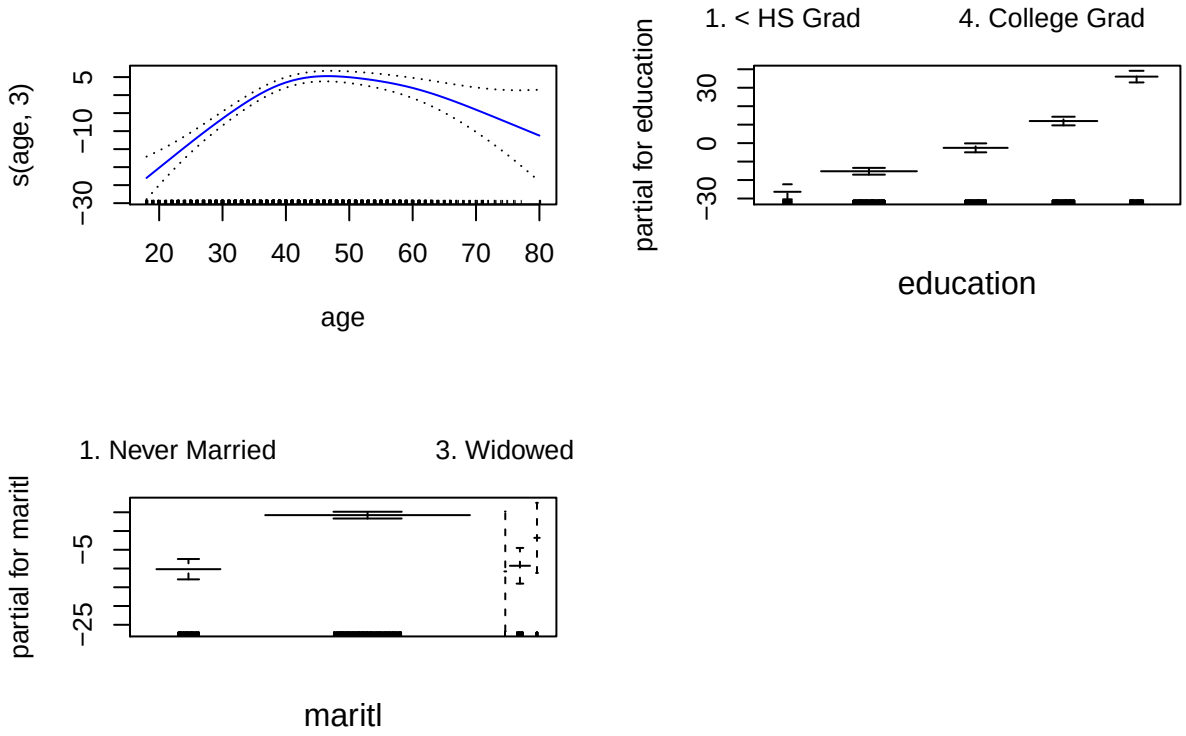
This is how the data looks.

```
ggplot(df, aes(x = age, y = wage,
               color = education,
               pch = maritl)) + geom_point() +
  ggtitle("Age, Education, Marital Status vs Wage") +
  theme_minimal()
```



This is how the model looks.

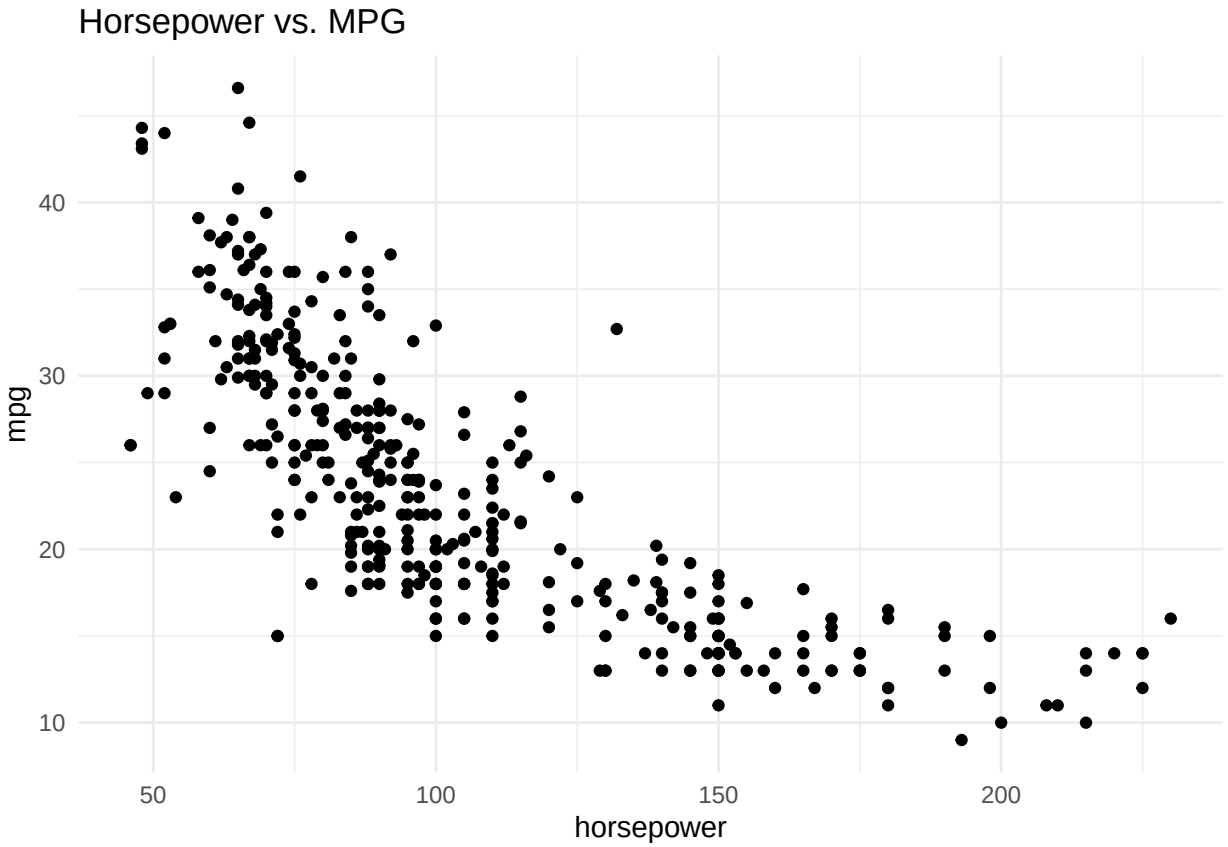
```
par(mfrow=c(2,2))
plot(model3, se=TRUE,col="blue")
```



Question 8: Fit some of the non-linear models investigated in this chapter to the `Auto` dataset. Is there evidence for non-linear relationships in this dataset? Create some informative plots to justify the answer.

The relationship that will be looked at is `mpg` vs. `horsepower`.

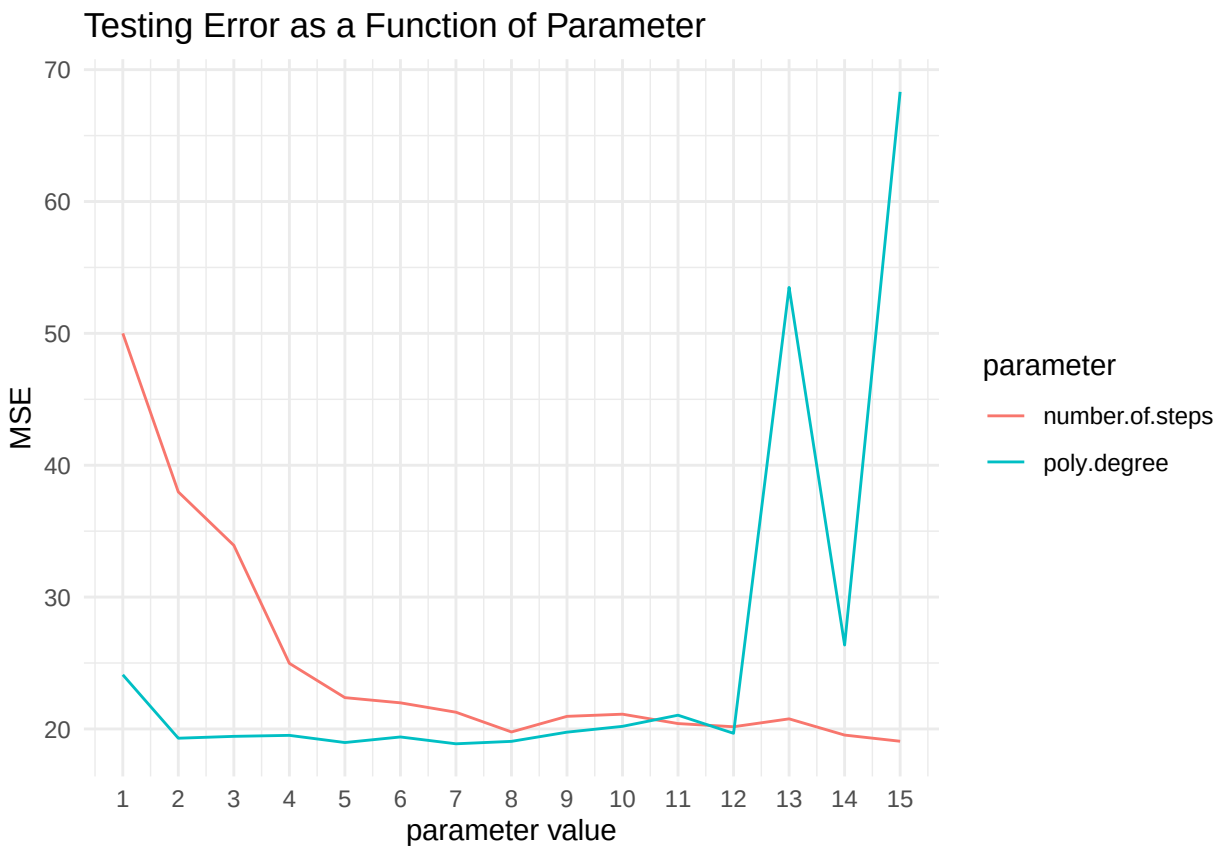
```
df = Auto
ggplot(df, aes(x = horsepower, y = mpg)) + geom_point() +
  ggtitle("Horsepower vs. MPG") +
  theme_minimal()
```



This relationship is quite nonlinear. Try using polynomial functions and step functions to find the best regression.

```
set.seed(8)
poly_mses = c()
for(i in 1:15){
  model = glm(data = df, mpg ~ poly(horsepower, i))
  poly_mses = c(poly_mses, cv.glm(df, model, K = 5)$delta[1])
}
cut_mses = c(50)
for(i in 2:15){
  df$hp_cut = cut(df$horsepower, i)
  model = glm(data = df, mpg ~ hp_cut)
  cut_mses = c(cut_mses, cv.glm(df, model, K=5)$delta[1])
}
mses_df = data.frame(x = 1:15,
                     "poly degree" = poly_mses,
                     "number of steps" = cut_mses)
mses_df %>% gather(parameter, value,
                   poly.degree, number.of.steps) %>%
  ggplot(aes(x = x, y = value,
             color = parameter)) +
  geom_path() +
  ggtitle("Testing Error as a Function of Parameter") +
  labs(x = "parameter value", y = "MSE") +
  scale_x_continuous(limits = c(1,15),
                     breaks = 1:15) +
```

```
  theme_minimal()
```



Too many polynomial degrees and too many steps does not result in lower error. After a polynomial degree of 12, the models had a much higher error. On the other hand, the number of cuts for the step function did not make a huge difference after 8 cuts. In fact, the lowest degree and number of steps occured at

```
which.min(mses_df$poly.degree)
```

```
## [1] 7
```

and

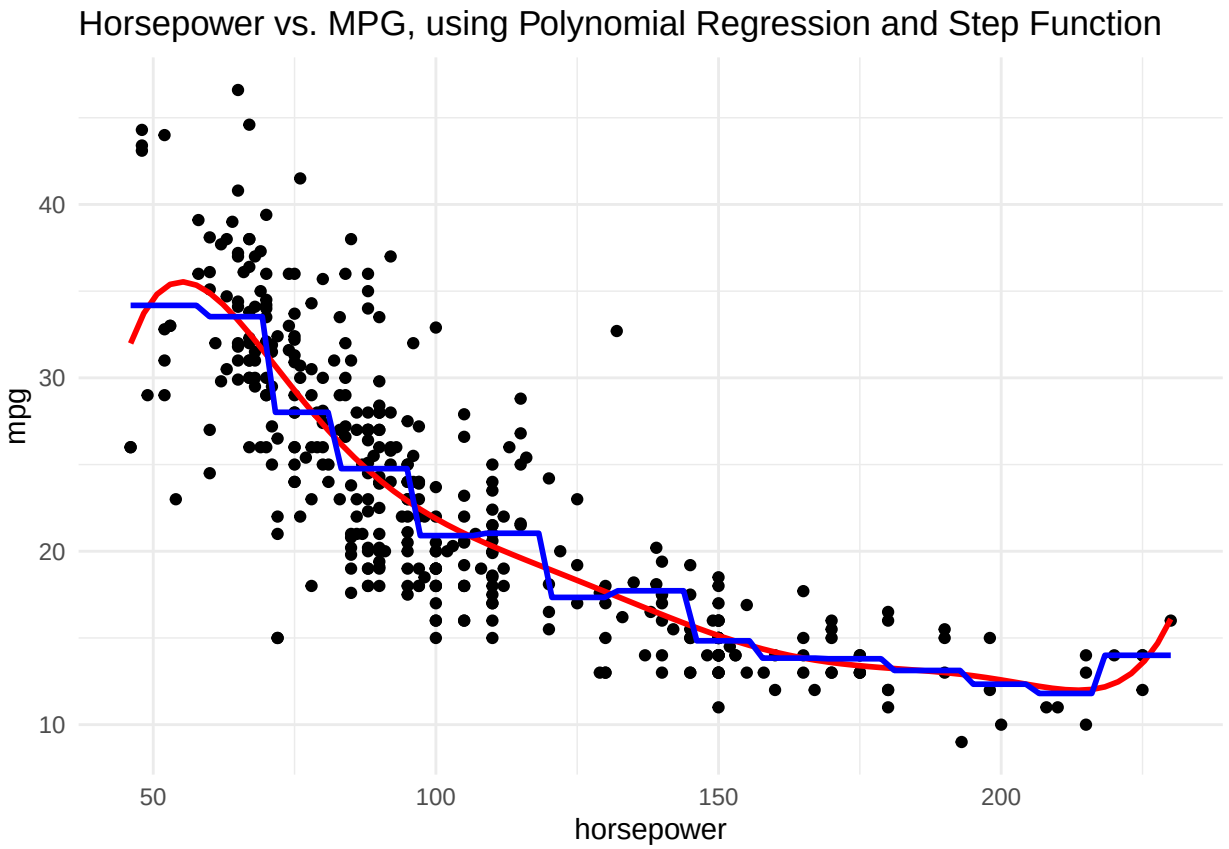
```
which.min(mses_df$number.of.steps)
```

```
## [1] 15
```

respectively.

This can be plotted on the dataset itself.

```
ggplot(df, aes(x = horsepower, y = mpg)) + geom_point() +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ poly(x, 7), color = "red") +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ cut(x, 15), color = "blue") +
  ggtitle("Horsepower vs. MPG, using Polynomial Regression and Step Function") +
  theme_minimal()
```



Question 9: This question uses the variable `dim` (the weighted mean of distances to five Boston employment centers) and `nox` (nitrogen oxides concentration in parts per 10 million) from the `Boston` data. Treat `dis` as the predictor and `nox` as the response.

(a) Use the `poly()` function to fit a cubic polynomial regression to predict `nox` using `dis`. Report the regression output and plot the resulting data and polynomial fits.

```
df = Boston
model1 = lm(nox ~ poly(dis, 3), data = df)
summary(model1)
```

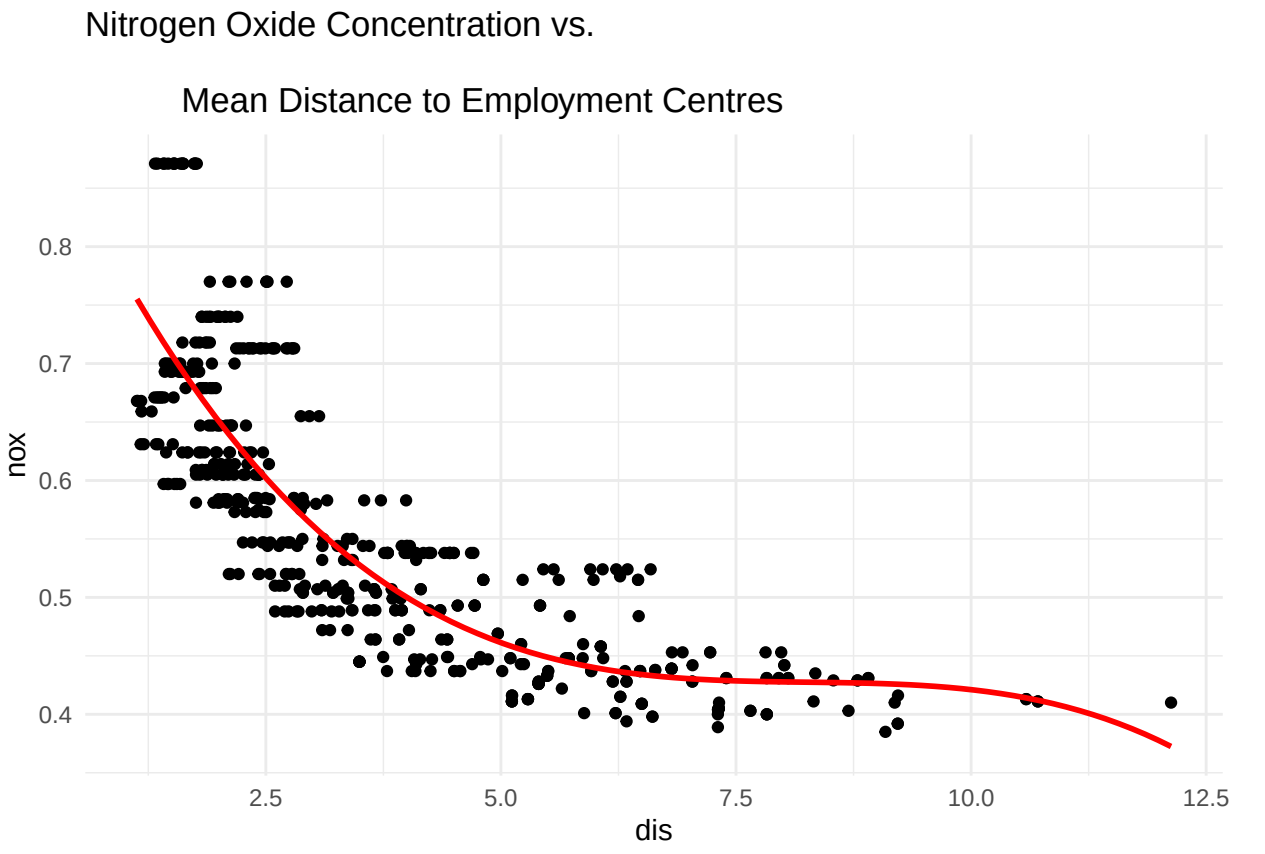
```
##
## Call:
## lm(formula = nox ~ poly(dis, 3), data = df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.121130 -0.040619 -0.009738  0.023385  0.194904
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.554695   0.002759 201.021  < 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
##
## Residual standard error: 0.06207 on 502 degrees of freedom
## Multiple R-squared:  0.7148, Adjusted R-squared:  0.7131
## F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

It can be seen that all the coefficient estimates are statistically significant. Furthermore, the adjusted R^2 is 0.7131, meaning that 71.31% of the variation in `nox` is explained by the regression curve.

Below is the polynomial fit on the dataset.

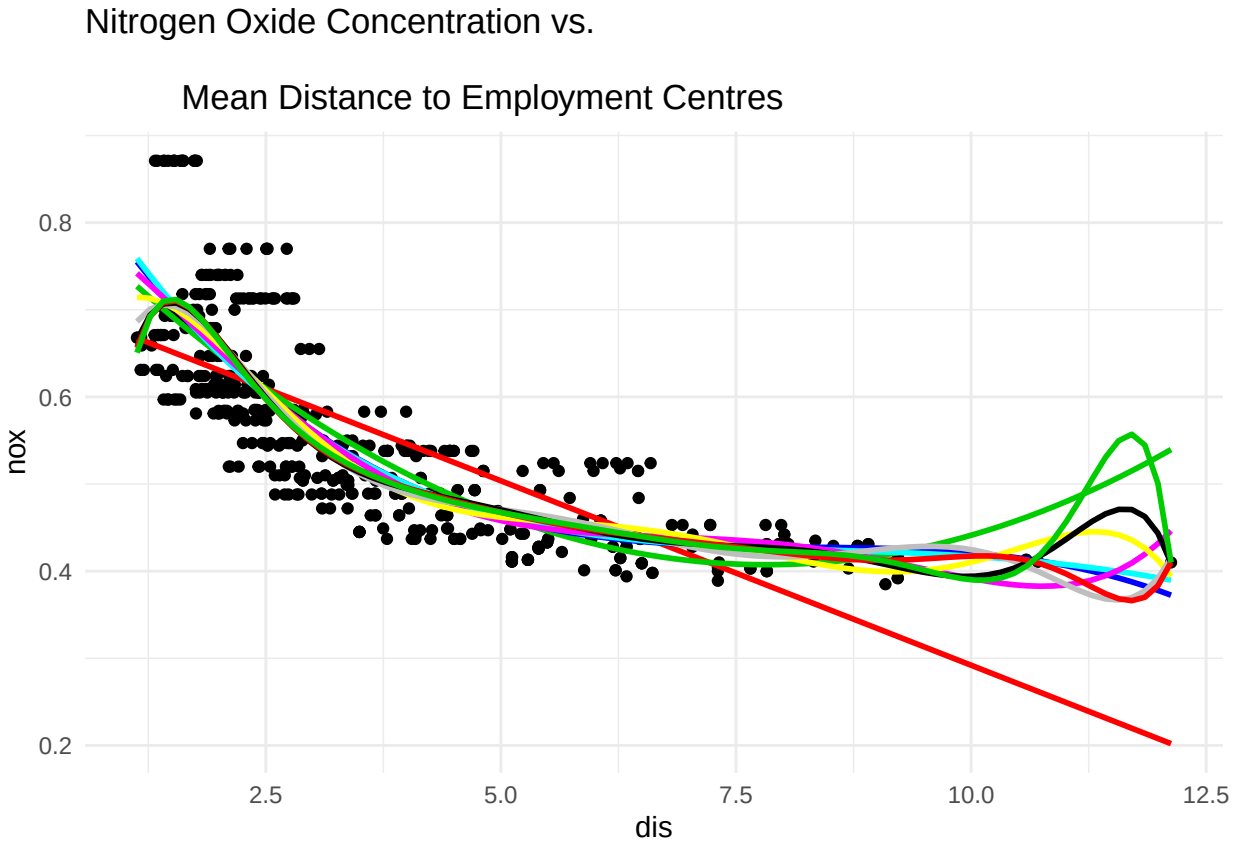
```
ggplot(df, aes(x = dis, y = nox)) + geom_point() +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ poly(x, 3), color = "red") +
  ggtitle("Nitrogen Oxide Concentration vs. \n
          Mean Distance to Employment Centres") +
  theme_minimal()
```



(b) Plot the polynomial fits for a range of different polynomial degrees and report the associated sum of squares.

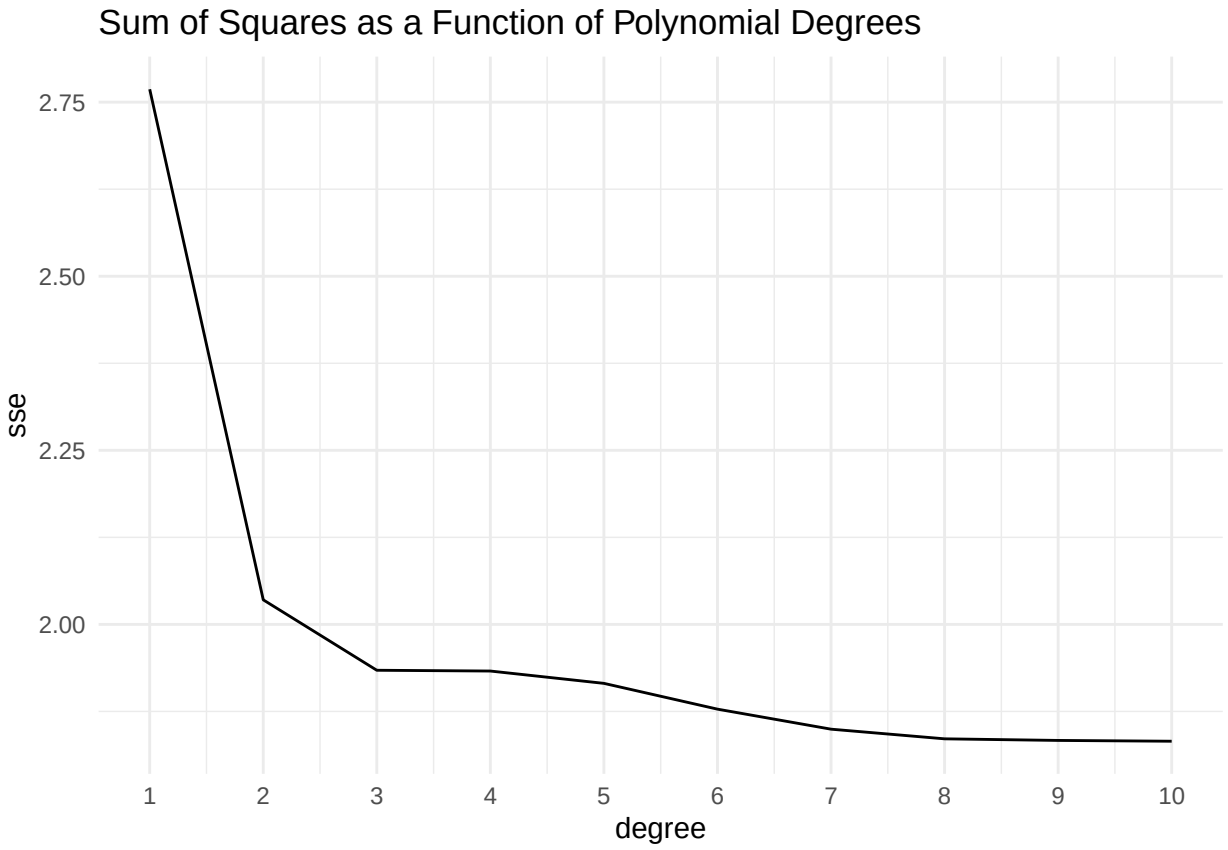
```
sse = c()
for(i in 1:10){
  model = lm(nox ~ poly(dis, i), data = df)
  sse = c(sse, sum(summary(model)$residuals^2))
}

ggplot(df, aes(x = dis, y = nox)) + geom_point() +
```

```
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ poly(x, 1), color = "450") +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ poly(x, 2), color = "451") +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ poly(x, 3), color = "452") +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ poly(x, 4), color = "453") +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ poly(x, 5), color = "454") +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ poly(x, 6), color = "455") +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ poly(x, 7), color = "456") +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ poly(x, 8), color = "457") +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ poly(x, 9), color = "458") +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ poly(x, 10), color = "459") +
  ggtitle("Nitrogen Oxide Concentration vs. \n
          Mean Distance to Employment Centres") +
  theme_minimal()
```



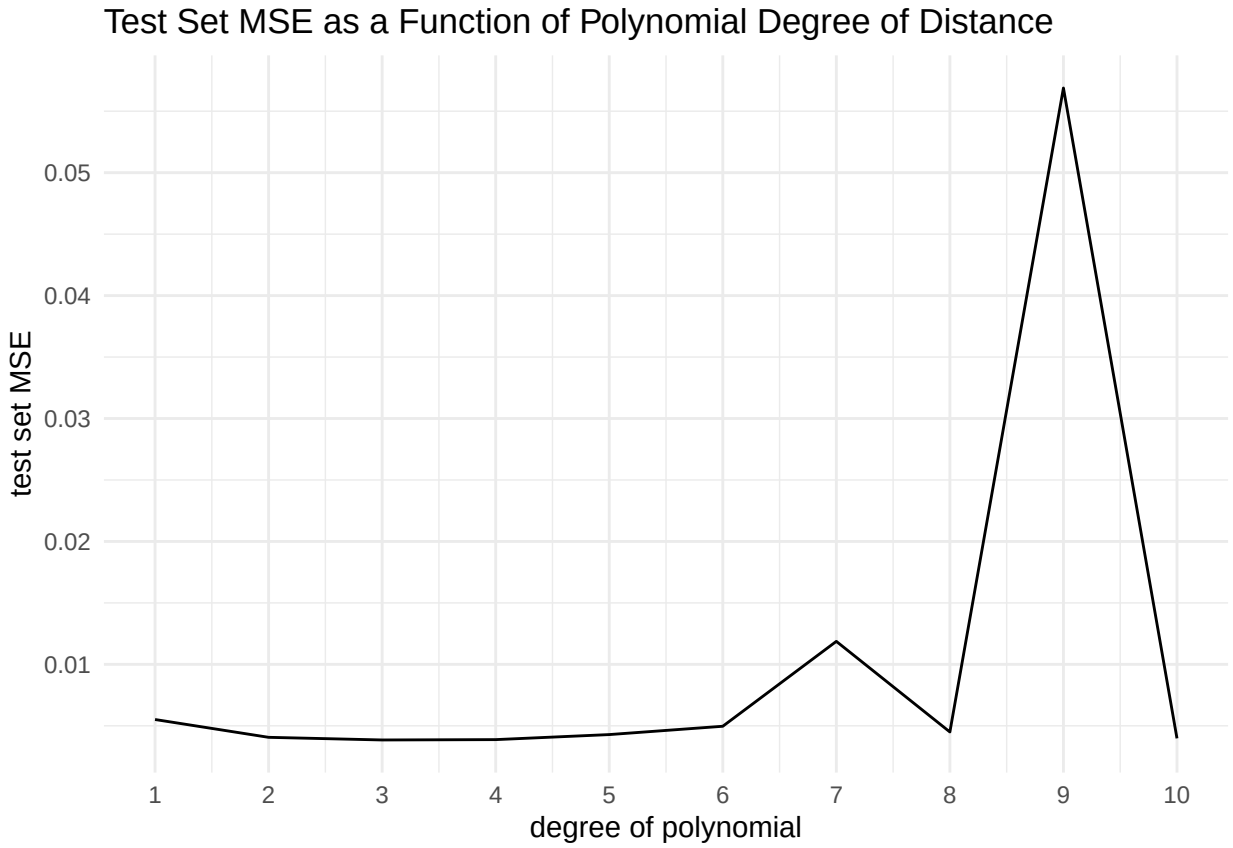
Above is the plot of the polynomial fits on the data. Below is the plot of the sum of squares.

```
sse_df = data.frame(degree = 1:10, sse = sse)
ggplot(sse_df, aes(x = degree, y = sse)) + geom_path() +
  scale_x_continuous(limits = c(1,10),
                     breaks = 1:10) +
  ggtitle("Sum of Squares as a Function of Polynomial Degrees") +
  theme_minimal()
```



(c) Perform cross-validation or another approach to select the optimal degree for the polynomial and explain the results.

```
mses = c()
for(i in 1:10){
  model = glm(data = df, nox ~ poly(dis, i))
  mses = c(mses, cv.glm(df, model, K = 5)$delta[1])
}
ggplot(data.frame(mses), aes(x=1:10, y = mses)) + geom_path() +
  ggtitle("Test Set MSE as a Function of Polynomial Degree of Distance") +
  labs(x = "degree of polynomial", y = "test set MSE") +
  scale_x_continuous(breaks = 1:10) +
  theme_minimal()
```



By cross validation, the optimal degree for the polynomial is

```
which.min(mses)
```

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

(d) Use the `bs()` function to fit a regression spline to predict `nox` using `dis`. Report the output for the fit using four degrees of freedom. How were the knots chosen? Plot the resulting fit.

```
model = lm(nox~bs(dis, df = 4), data = df)
summary(model)
```

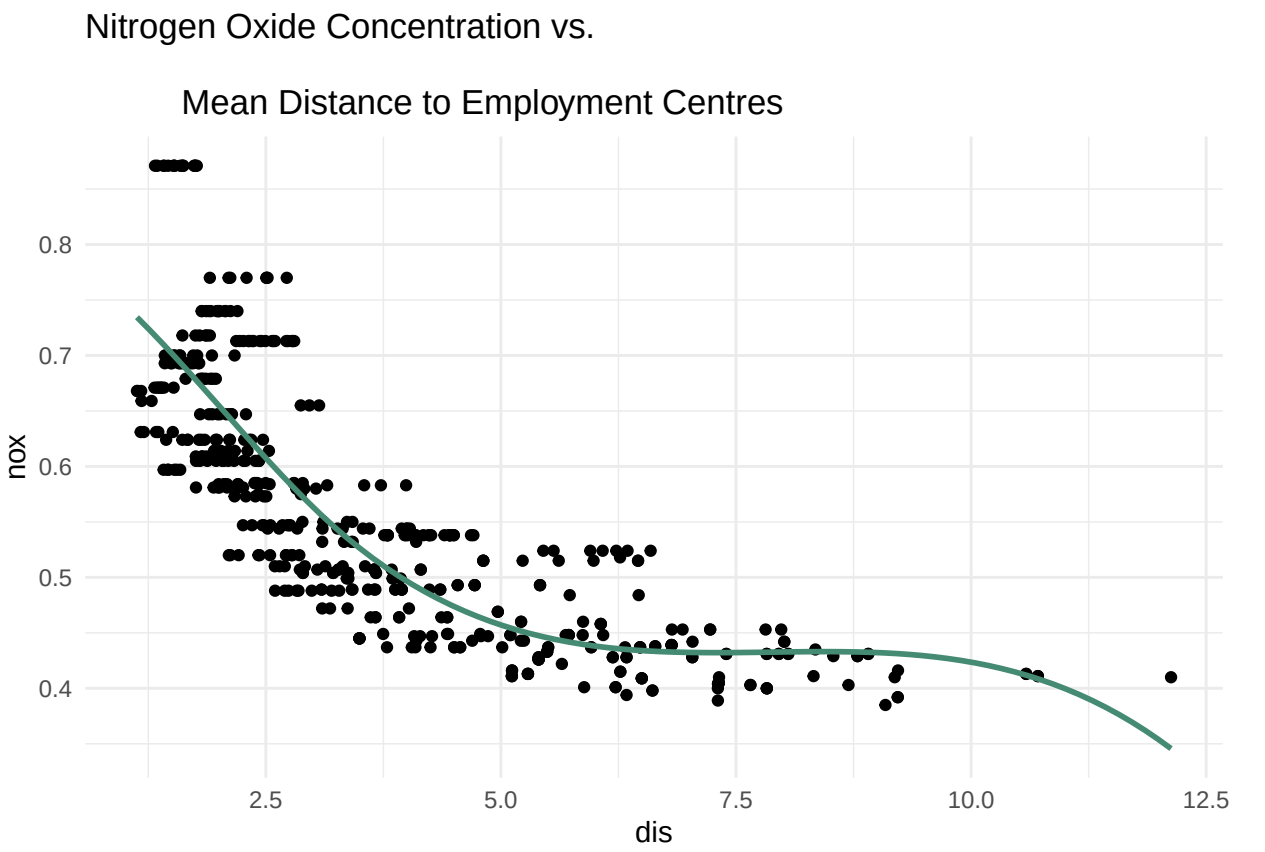
```
##
## Call:
## lm(formula = nox ~ bs(dis, df = 4), data = df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.124622 -0.039259 -0.008514  0.020850  0.193891
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.73447    0.01460  50.306  < 2e-16 ***
## bs(dis, df = 4)1 -0.05810    0.02186  -2.658  0.00812 **
## bs(dis, df = 4)2 -0.46356    0.02366 -19.596  < 2e-16 ***
## bs(dis, df = 4)3 -0.19979    0.04311  -4.634 4.58e-06 ***
## bs(dis, df = 4)4 -0.38881    0.04551  -8.544  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.06195 on 501 degrees of freedom
## Multiple R-squared:  0.7164, Adjusted R-squared:  0.7142
## F-statistic: 316.5 on 4 and 501 DF,  p-value: < 2.2e-16
```

This model has an adjusted R^2 of 0.7142, meaning that 71.42% of the variation in `nox` is explained by the regression model. Furthermore, all of the coefficient estimates are statistically significant. The knots were chosen by dividing the data into four uniform quantiles.

This is a plot of the resulting fit.

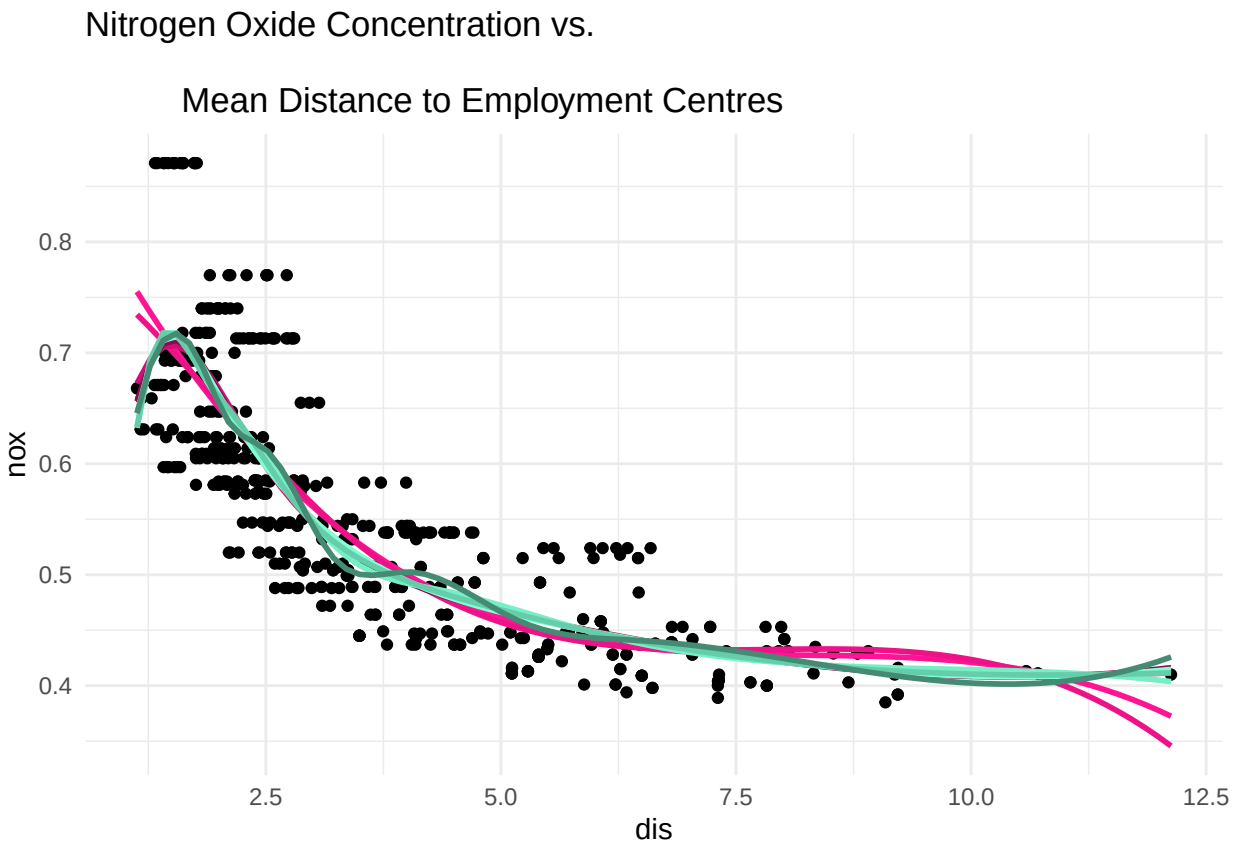
```
ggplot(df, aes(x = dis, y = nox)) + geom_point() +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ bs(x, df = 4), color = "aquamarine4") +
  ggtitle("Nitrogen Oxide Concentration vs. \n
          Mean Distance to Employment Centres") +
  theme_minimal()
```



(e) Now fit a regression spline for a range of degrees of freedom and plot the resulting fits and report the resulting RSS. Describe the results obtained.

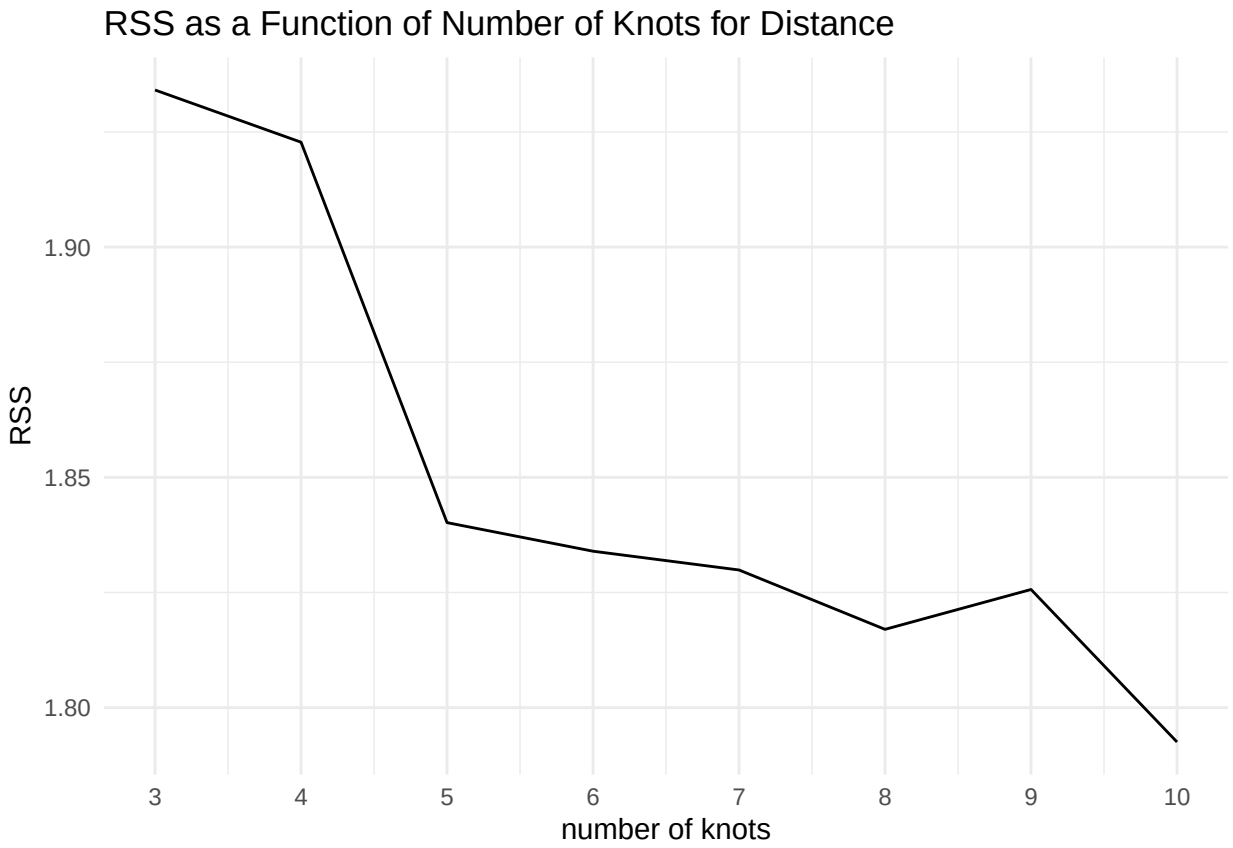
```
ggplot(df, aes(x = dis, y = nox)) + geom_point() +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ bs(x, df = 3), color = "deeppink1") +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ bs(x, df = 4), color = "deeppink2") +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ bs(x, df = 5), color = "deeppink3") +
  stat_smooth(method = "lm", se = FALSE,
```

```
              formula = y ~ bs(x, df = 6), color = "deeppink4") +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ bs(x, df = 7), color = "aquamarine1") +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ bs(x, df = 8), color = "aquamarine2") +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ bs(x, df = 9), color = "aquamarine3") +
  stat_smooth(method = "lm", se = FALSE,
              formula = y ~ bs(x, df = 10), color = "aquamarine4") +
  ggtitle("Nitrogen Oxide Concentration vs. \n
          Mean Distance to Employment Centres") +
  theme_minimal()
```



As the number of knots increases, the fit of the data becomes more flexible. The RSSs are plotted below.

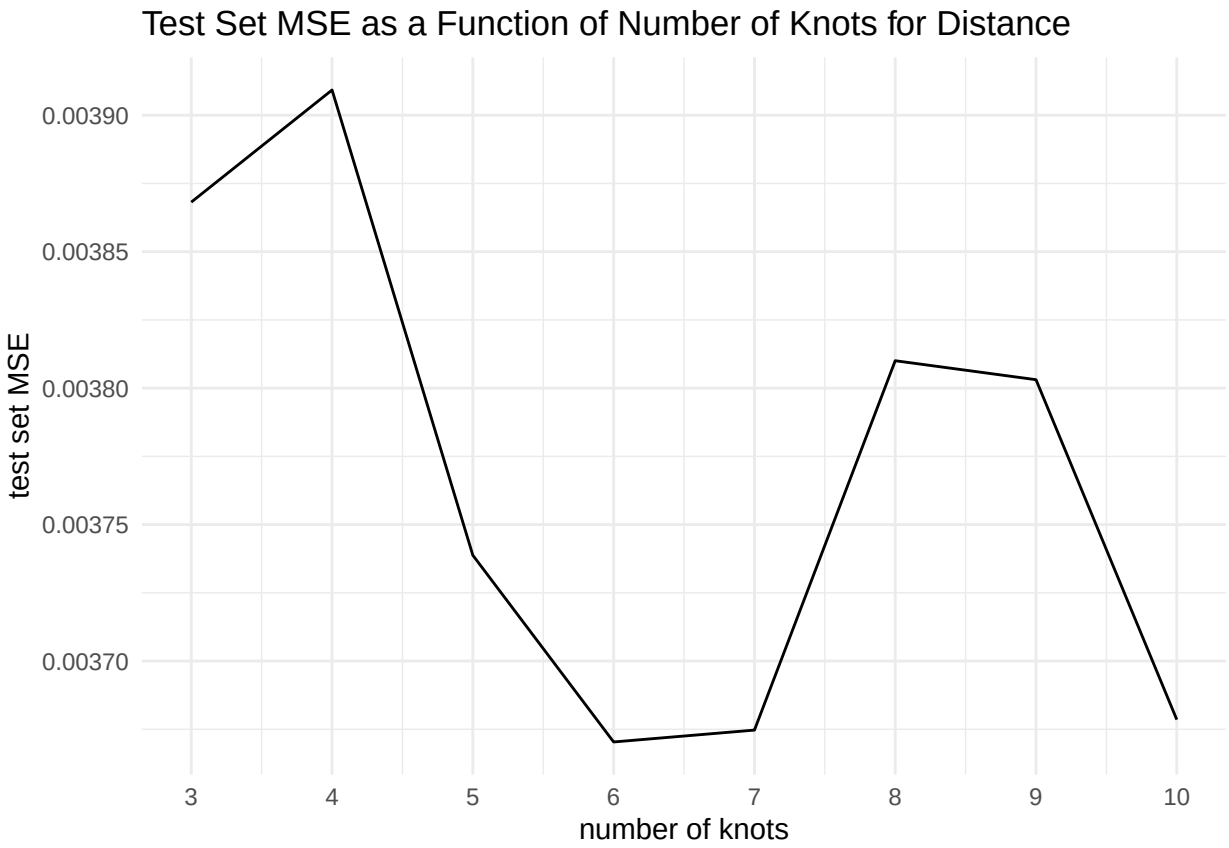
```
rss = c()
for(i in 3:10){
  model = lm(nox ~ bs(dis, df = i), data = df)
  rss = c(rss, sum((summary(model)$resid)^2))
}
ggplot(data.frame(rss), aes(x=3:10, y = rss)) + geom_path() +
  ggtitle("RSS as a Function of Number of Knots for Distance") +
  labs(x = "number of knots", y = "RSS") +
  scale_x_continuous(breaks = 3:10) +
  theme_minimal()
```



The lowest RSS occurs when there are 10 degrees of freedom, or number of knots.

(f) Perform cross-validation or another approach in order to select the best degrees of freedom for a regression spline on this data. Describe the results.

```
set.seed(6)
mses = c()
for(i in 3:10){
  model = glm(data = df, nox ~ bs(dis, df = i))
  mses = c(mses, cv.glm(df, model, K = 5)$delta[1])
}

ggplot(data.frame(mses), aes(x=3:10, y = mses)) + geom_path() +
  ggtitle("Test Set MSE as a Function of Number of Knots for Distance") +
  labs(x = "number of knots", y = "test set MSE") +
  scale_x_continuous(breaks = 3:10) +
  theme_minimal()
```



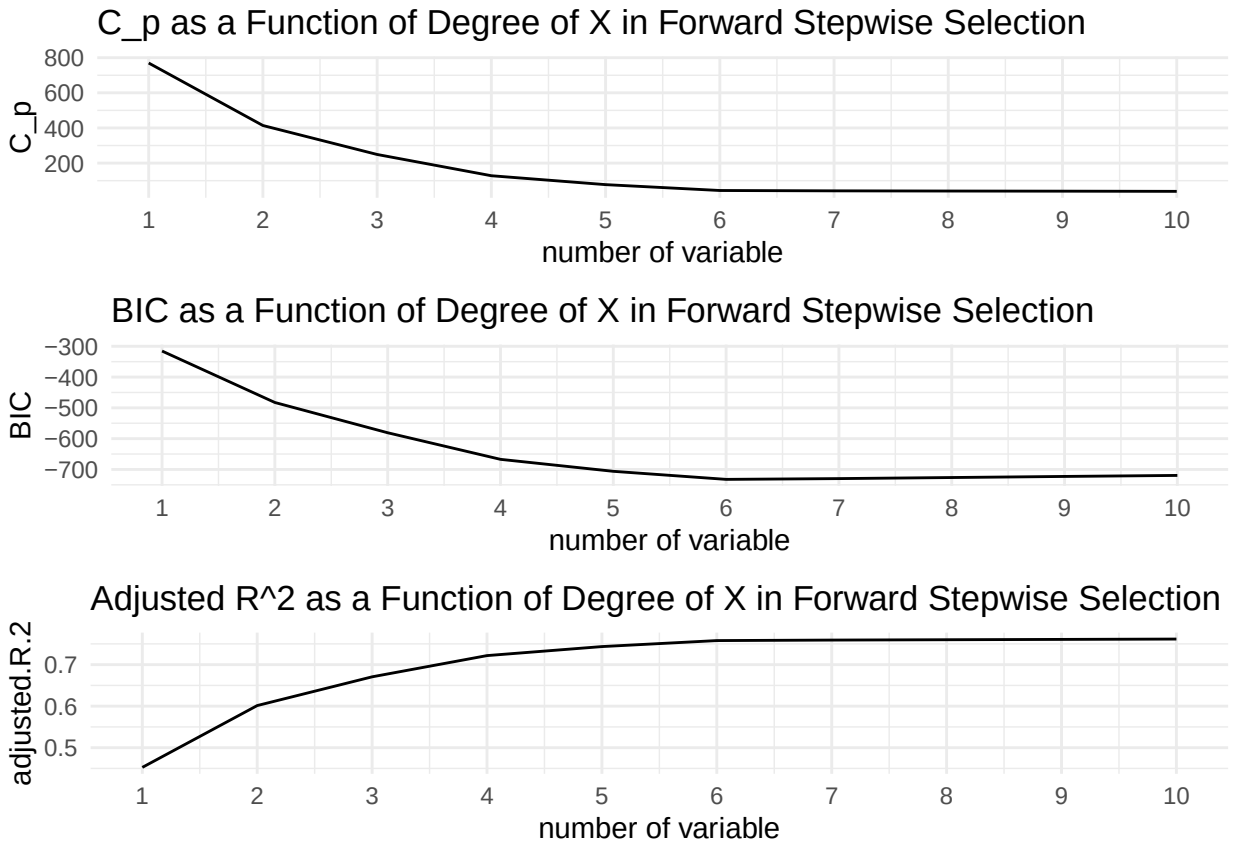
The lowest test set MSE occurs when the degrees of freedom is 6. After that, test set MSE steadily goes up and then down. This gives less degrees of freedom than when a model is created from the entire dataset.

Question 10: This question relates to the `College` dataset.

(a) Split the data into a training set and a test set. Using out-of-state tuition as the response and the other variables as the predictors, perform forward stepwise selection on the training set in order to identify a satisfactory model that uses just a subset of the predictors.

```
set.seed(10)
df = College
indices = sample(1:nrow(df), size = 0.7*nrow(df))
train = df[indices,]
test = df[indices,]

model = regsubsets(Outstate~., data = train, method = "forward", nvmax = 10)
model_stats = data.frame("num_vars" = seq(1,10),
                         "C_p" = summary(model)$cp,
                         "BIC" = summary(model)$bic,
                         "adjusted R^2" = summary(model)$adjr2)

g1 = ggplot(model_stats, aes(x = num_vars, y = C_p)) + geom_path() +
  scale_x_continuous(breaks = seq(1, 10, by = 1)) +
  labs(x = "number of variable") +
  ggtitle("C_p as a Function of Degree of X in Forward Stepwise Selection") +
  theme_minimal()
```

```
g2 = ggplot(model_stats, aes(x = num_vars, y = BIC)) + geom_path() +
  scale_x_continuous(breaks = seq(1, 10, by = 1)) +
  labs(x = "number of variable") +
  ggtitle("BIC as a Function of Degree of X in Forward Stepwise Selection") +
  theme_minimal()
g3 = ggplot(model_stats, aes(x = num_vars, y = adjusted.R.2)) + geom_path() +
  scale_x_continuous(breaks = seq(1, 10, by = 1)) +
  labs(x = "number of variable") +
  ggtitle("Adjusted R^2 as a Function of Degree of X in Forward Stepwise Selection") +
  theme_minimal()

grid.arrange(g1,g2,g3,ncol=1)
```



The best number of variables to use is

```
min(which.min(summary(model)$cp),
    which.min(summary(model)$bic))
```

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

So,

```
summary(model)
```

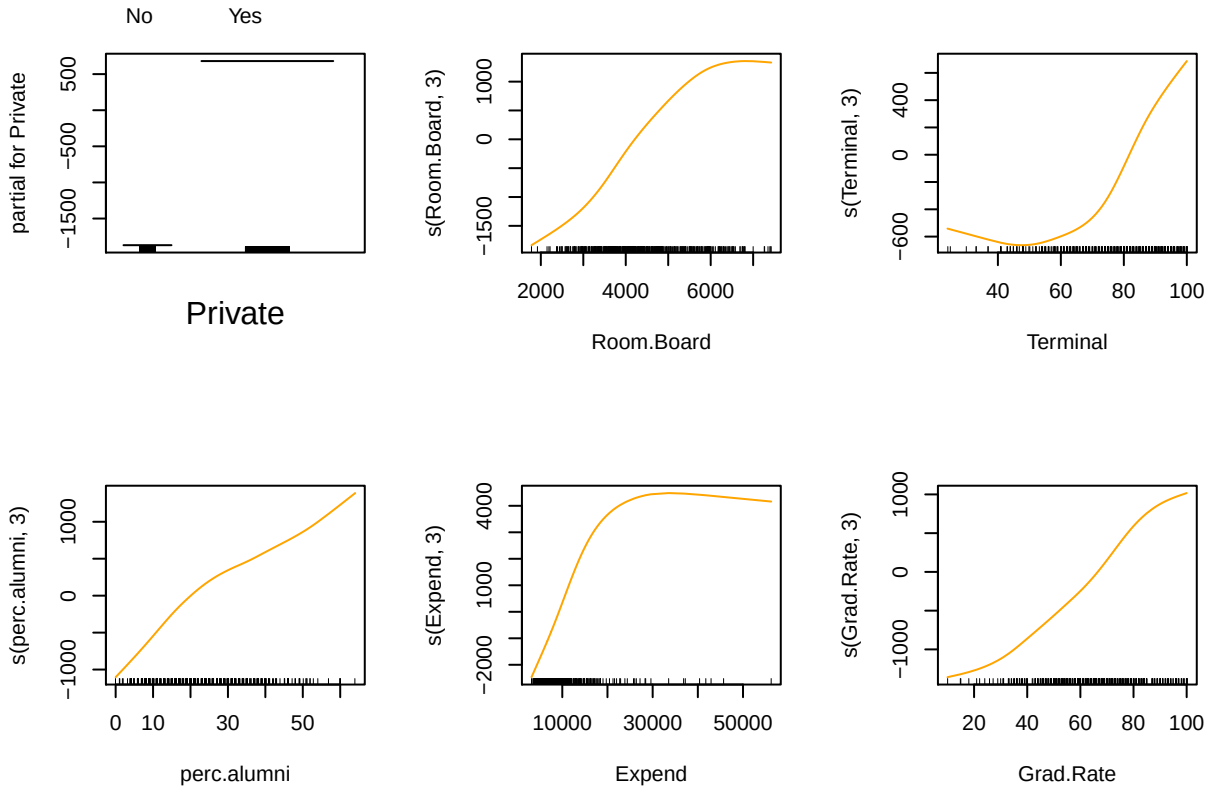
```
## Subset selection object
## Call: regsubsets.formula(Outstate ~ ., data = train, method = "forward",
##     nvmax = 10)
## 17 Variables  (and intercept)
##             Forced in Forced out
```

```
## PrivateYes      FALSE      FALSE
## Apps            FALSE      FALSE
## Accept          FALSE      FALSE
## Enroll          FALSE      FALSE
## Top10perc       FALSE      FALSE
## Top25perc       FALSE      FALSE
## F.Undergrad     FALSE      FALSE
## P.Undergrad     FALSE      FALSE
## Room.Board      FALSE      FALSE
## Books           FALSE      FALSE
## Personal        FALSE      FALSE
## PhD             FALSE      FALSE
## Terminal        FALSE      FALSE
## S.F.Ratio       FALSE      FALSE
## perc.alumni     FALSE      FALSE
## Expend          FALSE      FALSE
## Grad.Rate       FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: forward
##           PrivateYes Apps Accept Enroll Top10perc Top25perc F.Undergrad
## 1  ( 1 )  " "        " "  " "    " "    " "        " "        " "
## 2  ( 1 )  " "        " "  " "    " "    " "        " "        " "
## 3  ( 1 )  " "        " "  " "    " "    " "        " "        " "
## 4  ( 1 )  "*"        " "  " "    " "    " "        " "        " "
## 5  ( 1 )  "*"        " "  " "    " "    " "        " "        " "
## 6  ( 1 )  "*"        " "  " "    " "    " "        " "        " "
## 7  ( 1 )  "*"        " "  " "    " "    " "        " "        " "
## 8  ( 1 )  "*"        " "  " "    " "    "*"        " "        " "
## 9  ( 1 )  "*"        " "  " "    " "    "*"        " "        " "
## 10  ( 1 ) "*"        " "  "*"    " "    "*"        " "        " "
##           P.Undergrad Room.Board Books Personal PhD Terminal S.F.Ratio
## 1  ( 1 )  " "          "*"        " "   " "      " " " "      " "
## 2  ( 1 )  " "          "*"        " "   " "      " " " "      " "
## 3  ( 1 )  " "          "*"        " "   " "      " " " "      " "
## 4  ( 1 )  " "          "*"        " "   " "      " " " "      " "
## 5  ( 1 )  " "          "*"        " "   " "      " " "*"      " "
## 6  ( 1 )  " "          "*"        " "   " "      " " "*"      " "
## 7  ( 1 )  " "          "*"        " "   "*"      " " "*"      " "
## 8  ( 1 )  " "          "*"        " "   "*"      " " "*"      " "
## 9  ( 1 )  " "          "*"        " "   "*"      " " "*"      "*"
## 10  ( 1 ) " "          "*"        " "   "*"      " " "*"      "*"
##           perc.alumni Expend Grad.Rate
## 1  ( 1 )  " "         " "    " "
## 2  ( 1 )  "*"         " "    " "
## 3  ( 1 )  "*"         "*"    " "
## 4  ( 1 )  "*"         "*"    " "
## 5  ( 1 )  "*"         "*"    " "
## 6  ( 1 )  "*"         "*"    "*"
## 7  ( 1 )  "*"         "*"    "*"
## 8  ( 1 )  "*"         "*"    "*"
## 9  ( 1 )  "*"         "*"    "*"
## 10  ( 1 ) "*"         "*"    "*"
```

The 6 variables are: `Private`, `Room.Board`, `Terminal`, `perc.alumni`, `Expend` and `Grad.Rate`.

(b) Fit a GAM on the training data, using out-of-state tuition as the response and the features selected in the previous step as the predictors. Plot the results and explain the findings.

```
model1 = gam(Outstate ~ Private + s(Room.Board, 3) + s(Terminal, 3) +
               s(perc.alumni, 3) + s(Expend, 3) +
               s(Grad.Rate, 3), data = train)
par(mfrow = c(2, 3))
plot(model1, col = "orange")
```



(c) Evaluate the model obtained on the test set and explain the results obtained.

```
mean((test$Outstate - predict(model1, newdata = test))^2)
```

```
## [1] 3282792
```

The SSE is 3282792.

```
tss = mean((df$Outstate - mean(test$Outstate))^2)
1 - (3282792 / tss)
```

```
## [1] 0.7969051
```

and the R^2 is 0.7969, meaning that 79.69% of the variation in the `Outstate` is explained by the model.

(d) For which variables, if any, is there evidence of a non-linear relationship with the response?

```
summary(model1)
```

```
##
## Call: gam(formula = Outstate ~ Private + s(Room.Board, 3) + s(Terminal,
##     3) + s(perc.alumni, 3) + s(Expend, 3) + s(Grad.Rate, 3),
##     data = train)
## Deviance Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -7103.05 -1153.72    45.31  1298.46  4444.05
##
## (Dispersion Parameter for gaussian family taken to be 3388934)
##
##     Null Deviance: 9072439678 on 542 degrees of freedom
## Residual Deviance: 1782578206 on 525.9996 degrees of freedom
## AIC: 9724.258
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                     Df     Sum Sq    Mean Sq F value    Pr(>F)
## Private              1 2436052354 2436052354 718.826 < 2.2e-16 ***
## s(Room.Board, 3)     1 1829961693 1829961693 539.981 < 2.2e-16 ***
## s(Terminal, 3)       1  595666247  595666247 175.768 < 2.2e-16 ***
## s(perc.alumni, 3)    1  354887126  354887126 104.719 < 2.2e-16 ***
## s(Expend, 3)         1  677735407  677735407 199.985 < 2.2e-16 ***
## s(Grad.Rate, 3)      1  121349491  121349491  35.808 4.031e-09 ***
## Residuals          526 1782578206    3388934
## ---
## 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)          2  4.924  0.007606 **
## s(Terminal, 3)            2  2.921  0.054756 .
## s(perc.alumni, 3)         2  1.079  0.340751
## s(Expend, 3)              2 42.564 < 2.2e-16 ***
## s(Grad.Rate, 3)           2  1.748  0.175082
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

By the ANOVA test, there seems to be a highly non-linear relationship between the response and percent of alumni as well a moderate one between the response and `Terminal` as well as graduation rate.

Question 11: In Section 7.7, it was mentioned that GAMs are generally fit using a backfitting approach. The idea behind backfitting is actually quite simple. We will now explore backfitting in the context of multiple linear regression.

Suppose that we would like to perform multiple linear regression but we do not have software to do so. Instead, we only have software to perform simple linear regression. Therefore, we take the following iterative approach: we repeatedly hold all but one coefficient estiamte fixed at its current value and update only that coefficient estimate using a simple linear regression. The process is continued until convergence - that is, until the coefficient estimates stop changing.

We will now try this out on a toy example.

(a) Generate a response Y and two predictors X_1 and X_2, with $n = 100$.

```
set.seed(11)
x1 = rnorm(100)
x2 = rnorm(100)
```

```
eps = rnorm(100, sd = 0.3)
y = 2 + (7 * x1) - (5 * x2) + eps
```

(b) Initialize $\hat{\beta}_1$ to take on a value of your choice.

```
beta0 = rep(NA, 1000)
beta1 = rep(NA, 1000)
beta2 = rep(NA, 1000)
beta1[1] = 3
```

(c) Keeping $\hat{\beta}_1$ fixed, fit the model

$$Y - \hat{\beta}_1 X_1 = \beta_0 + \beta_2 X_2 + \varepsilon$$

You can do this as follows:

```
a = y - beta1[1]*x1
beta2[1] = lm(a~x2)$coef[2]
```

(d) Keeping $\hat{\beta}_2$ fixed, fit the model

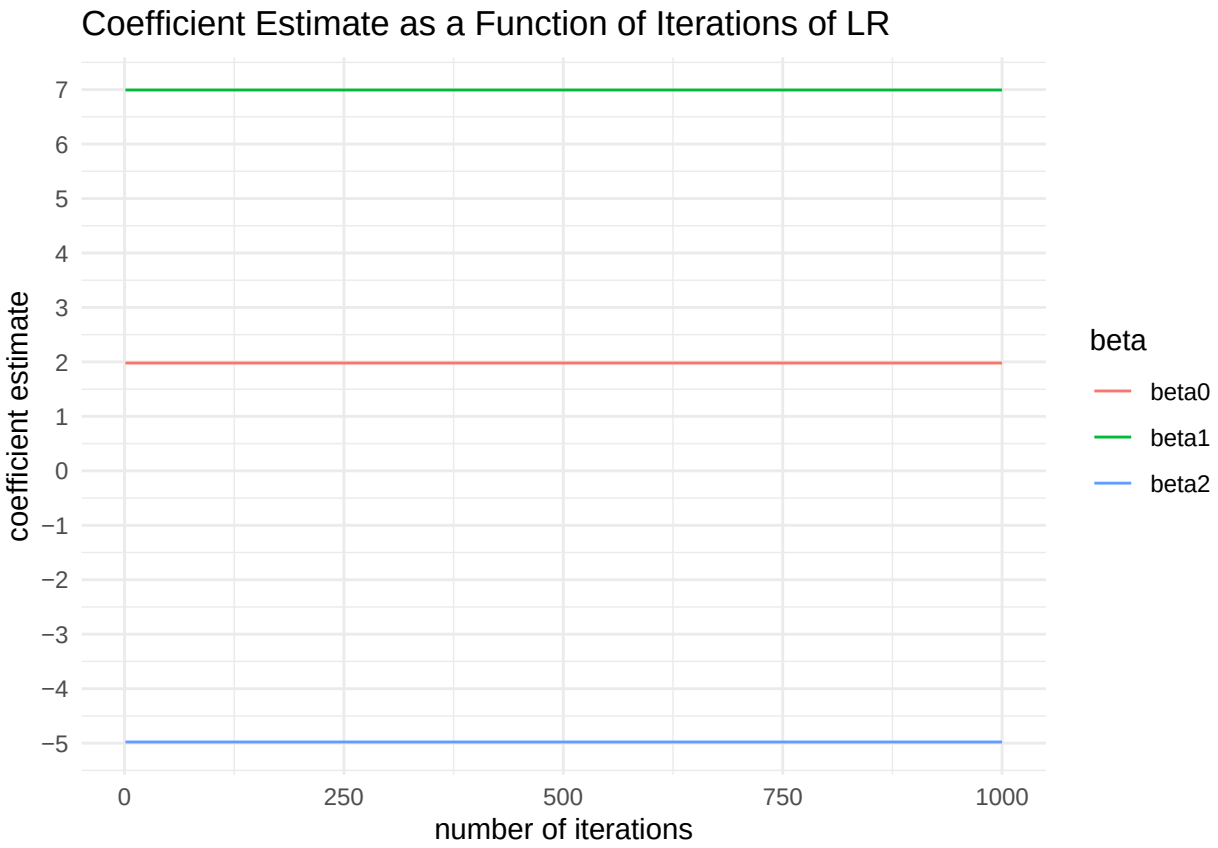
$$Y - \hat{\beta}_2 X_2 = \beta_0 + \beta_1 X_1 + \varepsilon$$

You can do this as follows:

```
a = y - beta2[1]*x2
beta1[1] = lm(a~x1)$coef[2]
```

(e) Write a for loop to repeat (c) and (d) 1,000 times. Report the estimates of $\hat{\beta}_0$, $\hat{\beta}_1$ and $\hat{\beta}_2$ at each iteration of the for loop. Create a plot in which each of these values is displayed, with $\hat{\beta}_0$, $\hat{\beta}_1$ and $\hat{\beta}_2$ each shown in a different color.

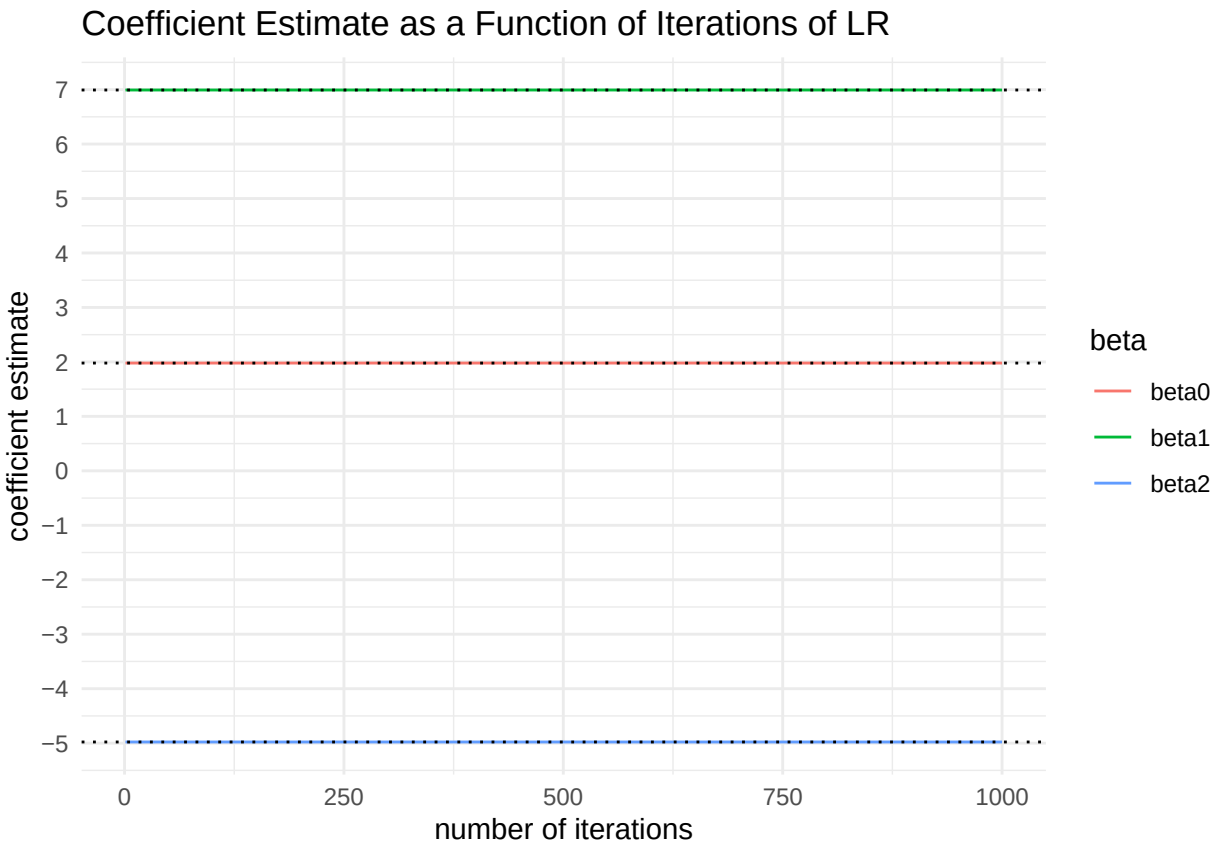
```
for (i in 1:1000) {
    a = y - beta1[i] * x1
    beta2[i] = lm(a ~ x2)$coef[2]
    a = y - beta2[i] * x2
    temp = lm(a ~ x1)
    if (i < 1000) {
        beta1[i + 1] = temp$coef[2]
    }
    beta0[i] = temp$coef[1]
}
beta_df = data.frame(x = 1:1000, beta0, beta1, beta2)
beta_df %>% gather(beta, coef, beta0, beta1, beta2) %>%
  ggplot(aes(x = x, y = coef, color = beta)) + geom_path() +
  scale_y_continuous(breaks = -5:8) +
  ggtitle("Coefficient Estimate as a Function of Iterations of LR") +
  labs(x = "number of iterations", y = "coefficient estimate") +
  theme_minimal()
```



The coefficient estimates were found quickly.

(f) Compare the above answer to the results of simply performing multiple linear regression to predict Y using X_1 and X_2. Add the coefficient estimates on top of the plot above.

```
model_real = lm(y ~ x1 + x2)
beta_df %>% gather(beta, coef, beta0, beta1, beta2) %>%
  ggplot(aes(x = x, y = coef, color = beta)) + geom_path() +
  scale_y_continuous(breaks = -5:8) +
  geom_hline(yintercept = summary(model_real)$coef[1], linetype = "dotted") +
  geom_hline(yintercept = summary(model_real)$coef[2], linetype = "dotted") +
  geom_hline(yintercept = summary(model_real)$coef[3], linetype = "dotted") +
  ggtitle("Coefficient Estimate as a Function of Iterations of LR") +
  labs(x = "number of iterations", y = "coefficient estimate") +
  theme_minimal()
```



(g) On this data set, how many backfitting iterations were required in order to obtain a "good" approximation to the multiple regression coefficient estimates?

It appears to be that one backfitting iteration was enough to obtain a "good" approximation to the multiple regression coefficient estimates.

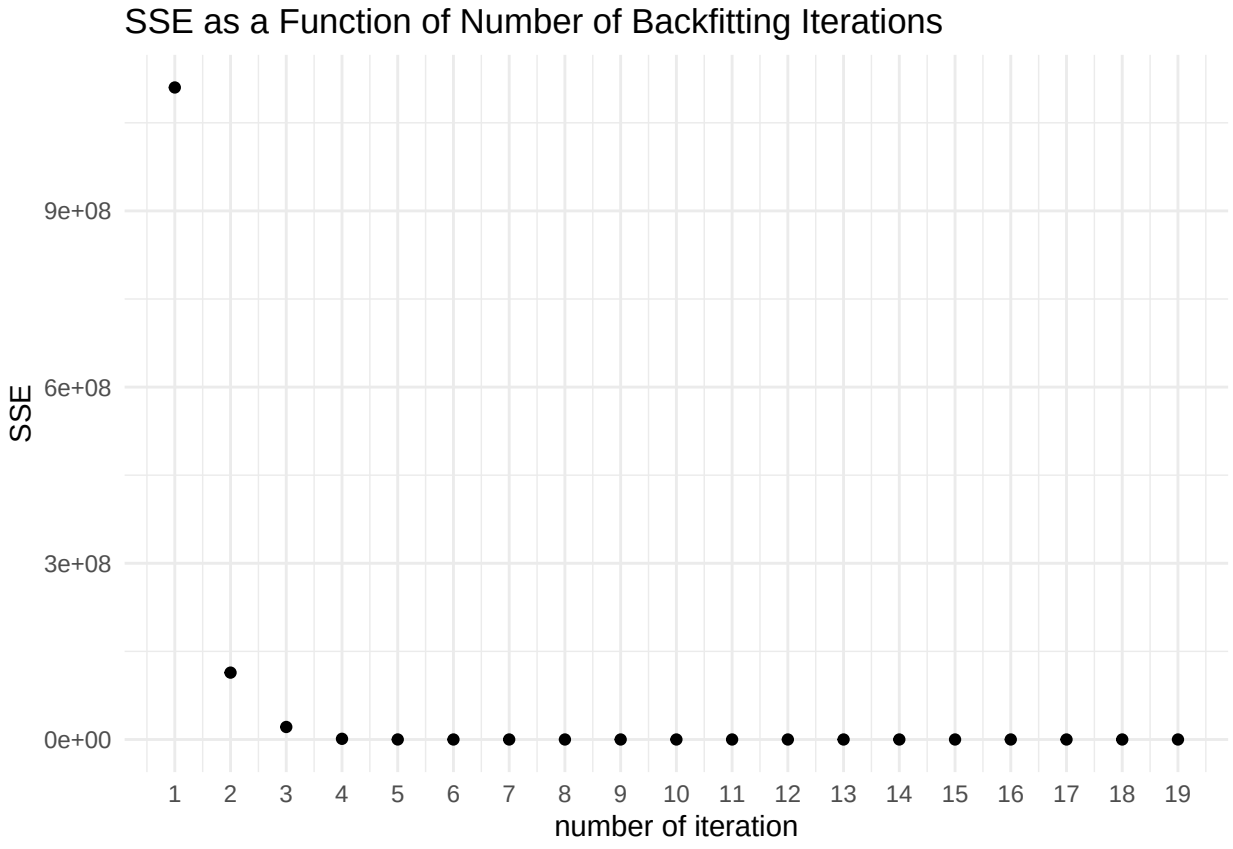
Question 12: This problem is a continuation of the previous exercise. In a toy example with $p = 100$, show that one can approximate the multiple linear regression coefficient estimates by repeatedly performing simple linear regression in a backfitting procedure. How many backfitting iterations are required in order to obtain a "good" approximation to the multiple regression coefficient estimates? Create a plot to justify this answer.

```
set.seed(12)
p = 100
n = 1000
x = matrix(ncol = p, nrow = n)
coeff = c()
for (i in 1:p) {
    x[, i] = rnorm(n)
    coeff = c(coeff, rnorm(1) * 100)
}
y = x %*% coeff + rnorm(n)

beta = rep(0, p)
```

```
sse = c(sum(y - x %*% beta)^2)

while(i < 300){
  for(j in 1:p){
    a = y - x %*% beta + beta[j] * x[, j]
    beta[j] = lm(a ~ x[, j])$coef[2]
  }
  sse = c(sse, sum(y - x %*% beta)^2)
  if(abs(diff(tail(sse, 2))) < 1e-4){
    break
  }
  else{
    i = i + 1
  }
}
sse_df = data.frame(x = 1:length(sse), sse)
```

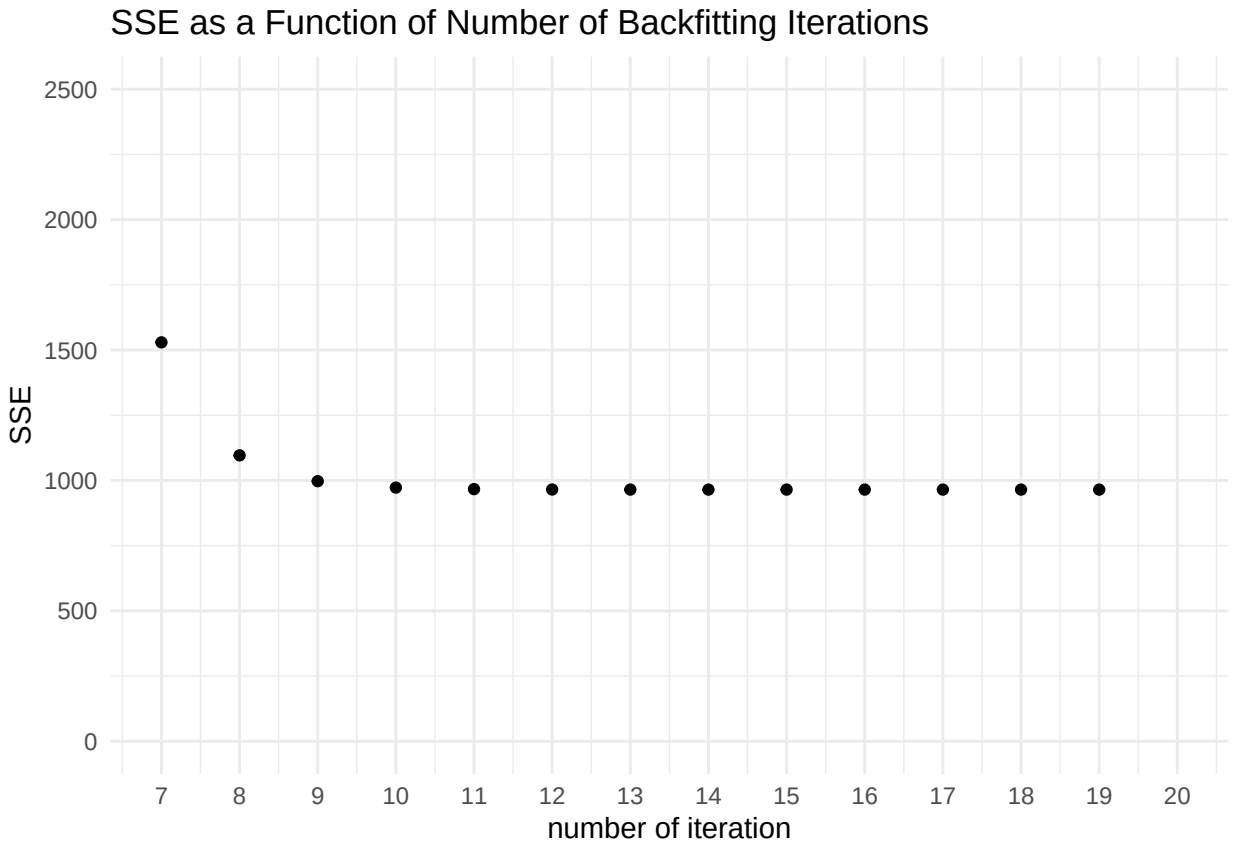
This is the plot of the SSE vs iterations.

```
ggplot(sse_df, aes(x = x, y = sse)) + geom_point() +
  ggtitle("SSE as a Function of Number of Backfitting Iterations") +
  labs(x = "number of iteration", y = "SSE") +
  scale_x_continuous(breaks = 1:19) +
  theme_minimal()
```



A "good" approximation to the multiple regression coefficient estimate was found relatively quickly, after 19 iterations! In fact, the SSE after the 8th iteration do not change by a lot.

```r
ggplot(sse_df, aes(x = x, y = sse)) + geom_point() +
  ggtitle("SSE as a Function of Number of Backfitting Iterations") +
  labs(x = "number of iteration", y = "SSE") +
  scale_x_continuous(limits = c(7, 20), breaks = 7:20) +
  scale_y_continuous(limits = c(0, 2500)) +
  theme_minimal()
```



All of the practice applied exercises in this document are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.