Stats 330 Assignment 4

Hasnain Cheena

07/06/2020

Question 1

1a)

```
feuro.df = read.csv('feuro.csv', header = TRUE)

#create BMI
feuro.df <- transform(feuro.df, BMI = weight / height^2)

# Sort by age
sort.order <- with(feuro.df, order(age))
feuro.df <- feuro.df[sort.order, ]

#fit linear model with quadratic in age
quadbmi.fit = lm(BMI ~ poly(age, 2, raw=TRUE), feuro.df)
```

The model equation is shown below:

$$BMI_i = \beta_2 age_i^2 + \beta_1 age_i + \beta_0$$

Where BMI_i is the ith European woman's body mass index and age_i is the ith European woman's age in years.
To find the age at which a European woman's BMI is maximized the first derivative of the model equation is found and equated to 0 as shown below:

$$\frac{d}{dx}(\beta_2 age_i^2 + \beta_1 age_i + \beta_0) = 0$$

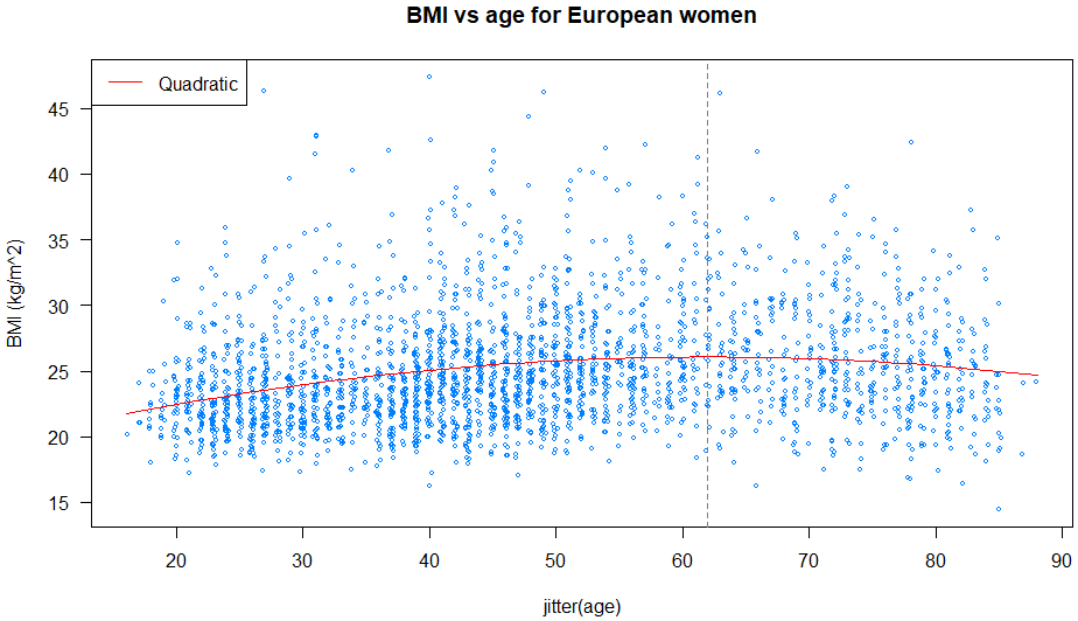
$$2\beta_2 age_i + \beta_1 = 0$$

$$age_i = \frac{-\beta_1}{2\beta_2}$$

```
obtain age where bmi is maximised
theta.hat = (-1*quadbmi.fit$coefficients[2])/(2*quadbmi.fit$coefficients[3])
names(theta.hat) = "Age at which bmi is maximised"
theta.hat

## Age at which bmi is maximised
##                       61.92375
```

Therefore, we estimate the age at which BMI is highest for a typical European woman is 61.9 years old.

```
azure  <- "#007FFF"
plot(BMI ~ jitter(age), data = feuro.df, col = azure, cex = 0.5,las = 1, main
= "BMI vs age for European women",  ylab = "BMI (kg/m^2)")
lines(with(feuro.df, age), fitted(quadbmi.fit), col = "red", lwd=1)
abline(v=c(theta.hat), lty="dashed", col="gray50")
legend("topleft",legend=c("Quadratic"),col=c("red"),lty=rep(1))
```



To produce a high-quality scatterplot the jitter command was used. The vertical dashed line in the scatterplot shows the age at which BMI is maximized for European women.

The scatterplot shows that for European women BMI increases until about 61.9 years old, after which it declines. Furthermore, the plot shows that the data is right skewed. This is because majority of the observations relate to women that are 55 years old or younger.

1b)

```
#reproducible
set.seed(106)

n.sims = 10000
#create vectors to store estimates
est.theta = numeric(n.sims)
#number of observations
n = nrow(feuro.df)
#calculate expected values
feuro.df$expected.values = quadbmi.fit$coefficients[1] + quadbmi.fit$coeffici
ents[2]*feuro.df$age + quadbmi.fit$coefficients[3]*(feuro.df$age)^2
#sigma hat estimate
```

```r
sigma.hat = summary(quadbmi.fit)$sigma

#simulate
for (i in 1:n.sims){
  #simulate responses
  bmi.simulated = rnorm(n, feuro.df$expected.values, sigma.hat)

  #refit
  sim.fit = lm(bmi.simulated ~ poly(feuro.df$age, 2, raw=TRUE))

  #calculate theta and save
  est.theta[i] = (-1*sim.fit$coefficients[2])/(2*sim.fit$coefficients[3])
}

#quantiles to obtain 95% interval
a = theta.hat - quantile(est.theta, prob=0.025)
b = quantile(est.theta, prob=0.975) - theta.hat
ci = c(theta.hat-b, theta.hat+a)
names(ci) = c("2.5%", "97.5%")
ci
```

```
##     2.5%    97.5%
## 57.52950 64.94484
```

As a result of parametric bootstrapping, we estimate that the expected age at which a European woman's BMI is highest is between 57.5 and 64.9 years old.

1c)

```r
#reproducible
set.seed(106)

n.sims = 10000
#create vectors to store estimates
est.theta = numeric(n.sims)
#number of observations
n = nrow(feuro.df)

#simulate
for (i in 1:n.sims){
  #resample dataframe
  samp = sample(1:n, replace=TRUE)
  boot.df = feuro.df[samp,]

  #fit the model
  sim.fit = lm(BMI ~ poly(age, 2, raw=TRUE), boot.df)

  #calculate theta and save
  est.theta[i] = (-1*sim.fit$coefficients[2])/(2*sim.fit$coefficients[3])
}
```

```r
#quantiles to obtain 95% interval
a = theta.hat - quantile(est.theta, prob=0.025)
b = quantile(est.theta, prob=0.975) - theta.hat
ci = c(theta.hat-b, theta.hat+a)
names(ci) = c("2.5%", "97.5%")
ci
```

```
##     2.5%    97.5%
## 57.44098 64.95676
```

As a result of non-parametric bootstrapping, we estimate that the expected age at which a European woman's BMI is highest is between 57.4 and 64.9 years old.

Therefore, the confidence intervals created by parametric and non-parametric bootstrapping are very similar to one another.

1d)

```r
library(msm)

#using original model fit from above
quadbmi.fit = lm(BMI ~ poly(age, 2, raw=TRUE), feuro.df)

#delta method
dm.sd = deltamethod(~x1/(2*x2), coef(quadbmi.fit), vcov(quadbmi.fit))
dm.CI = -coef(quadbmi.fit)[2]/(2*coef(quadbmi.fit)[3]) + 1.96 * c(-1,1) * dm.
sd
names(dm.CI) = c("2.5%", "97.5%")
dm.CI
```

```
##     2.5%    97.5%
## 52.22546 71.62204
```

As a result of the delta-method, we estimate that the expected age at which a European woman's BMI is highest is between 52.2 years old and 71.6 years old.

Therefore, the confidence interval created by the delta-method is relatively wider than the confidence intervals created by both parametric and non-parametric bootstrapping. In contrast, the confidence intervals created by parametric and non-parametric bootstrapping are very close to one another.

1e)

The results obtained above will not be able to generalize well here and now. This is because:

1. The original study was conducted around 25 years ago, in the mid-1990s. Factors that affect BMI such as the amount and type of food we consume have drastically changed since the mid-1990s. Therefore, the population in the original study is not representative of the population today.

2. The original study contained both male and female participants. However, our results were based on female participants only. Thus, as the results do not consider the male population, they may not be representative of the male population. Therefore, the results may not generalize.

3. The original study contained 4 major ethnic groups: “Europeans”, “Maori”, “Polynesian” and “Others”. Our results were only based on the “European” group and did not consider the other ethnicities. As a result, our results may not generalize to any other ethnicities.

Question 2

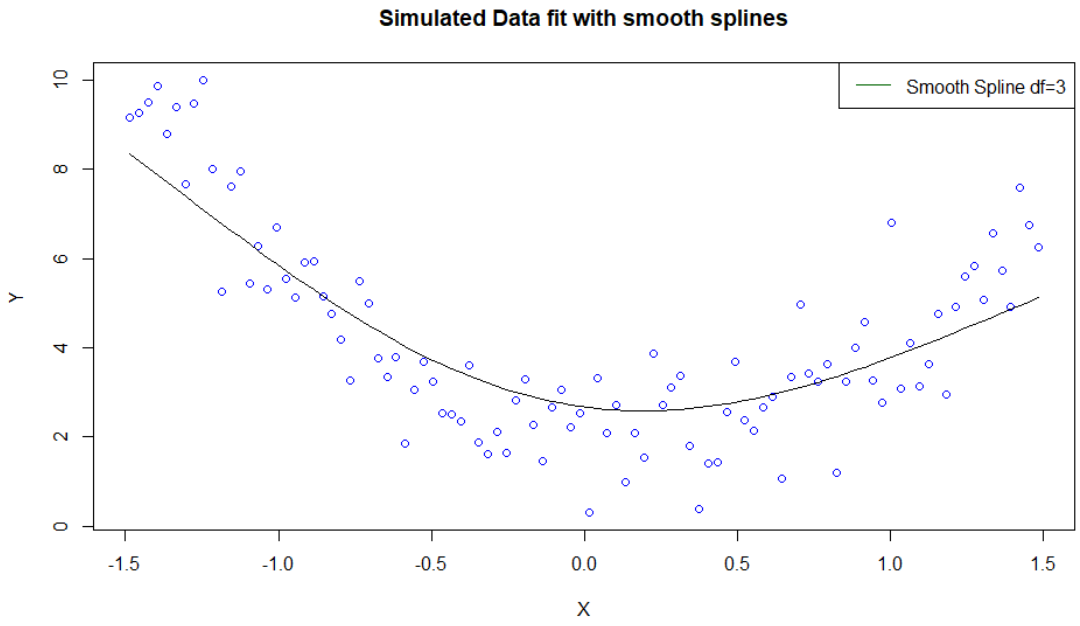
```
library(datasets)
library(MuMIn)

fm1 <- lm(sr ~ pop15 + pop75 + dpi + ddpi, data = LifeCycleSavings)

#use dredge to get a model selected from a table of models
#only keep models that include models with ddpi and use BIC to rank models
options(na.action="na.fail", width=120)
dredge(fm1,rank="BIC", subset=with(ddpi))

## Fixed term is "(Intercept)"

## Global model call: lm(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = Li
feCycleSavings)
## ---
## Model selection table
##    (Intrc)   ddpi        dpi   pop15   pop75 df   logLik   BIC delta weigh
t
## 6   15.600 0.4428            -0.2164          4 -136.943 289.5  0.00  0.43
9
## 14  28.120 0.4278            -0.4518 -1.8350  5 -135.171 289.9  0.37  0.36
6
## 8   19.280 0.3929 -0.0008704 -0.2884          5 -136.416 292.4  2.86  0.10
5
## 16  28.570 0.4097 -0.0003369 -0.4612 -1.6910  6 -135.098 293.7  4.13  0.05
6
## 10   5.470 0.4636                     1.0730  4 -140.211 296.1  6.54  0.01
7
## 4    6.360 0.5292  0.0011950                  4 -141.022 297.7  8.16  0.00
7
## 2    7.883 0.4758                             3 -142.990 297.7  8.18  0.00
7
## 12   5.487 0.4738  0.0001972          0.9529  5 -140.190 299.9 10.41  0.00
2
## Models ranked by BIC(x)
```

The table above shows a selection of models ranked by the BIC criteria that contain the variable *ddpi*.

Question 3

3a)

```
# Generate the 'original' data
#ID: 190411106
set.seed(106)

n <- 100
X <- scale(3 * (1:n)/n, scale = FALSE)
myfun <- function(x)
  2 - x + 3*x*x
Y <- myfun(X) + rnorm(n)

plot(X, Y, col = "blue", main="Simulated Data fit with smooth splines")
fit <- smooth.spline(X, Y, df = 3 , all = TRUE)
lines(fit, lty = 1, col = "black", lwd = 1)
legend("topright",legend=c("Smooth Spline df=3"),col=c("darkgreen"),lty=rep(1
))
```

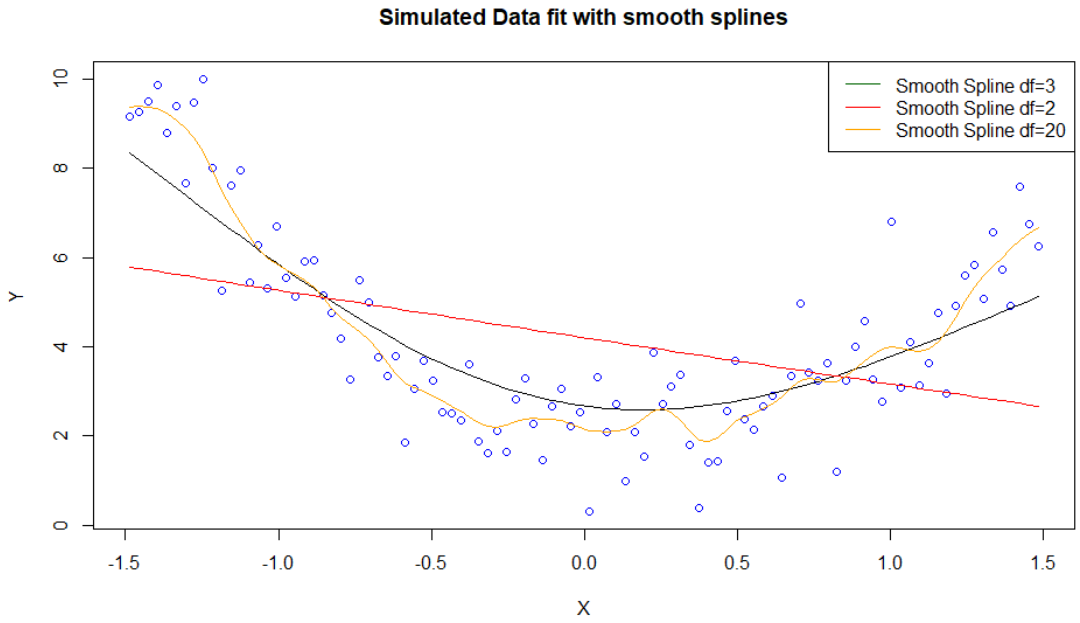


The scatterplot shows that the spline overlaid (smooth spline of df=3) is underfitting the data.

3b)

```
#smooth spline with df=2
spline.df2 <- smooth.spline(X, Y, df = 2 , all = TRUE)

#smooth spline with df=20
spline.df20 <- smooth.spline(X, Y, df = 20 , all = TRUE)
```

```r
plot(X, Y, col = "blue", main="Simulated Data fit with smooth splines")
lines(fit, lty = 1, col = "black", lwd = 1)
lines(spline.df2, lty = 1, col = "red", lwd = 1)
lines(spline.df20, lty = 1, col = "orange", lwd = 1)
legend("topright",legend=c("Smooth Spline df=3", "Smooth Spline df=2", "Smoot
h Spline df=20"),col=c("darkgreen", "red", "orange"),lty=rep(1,3))
```

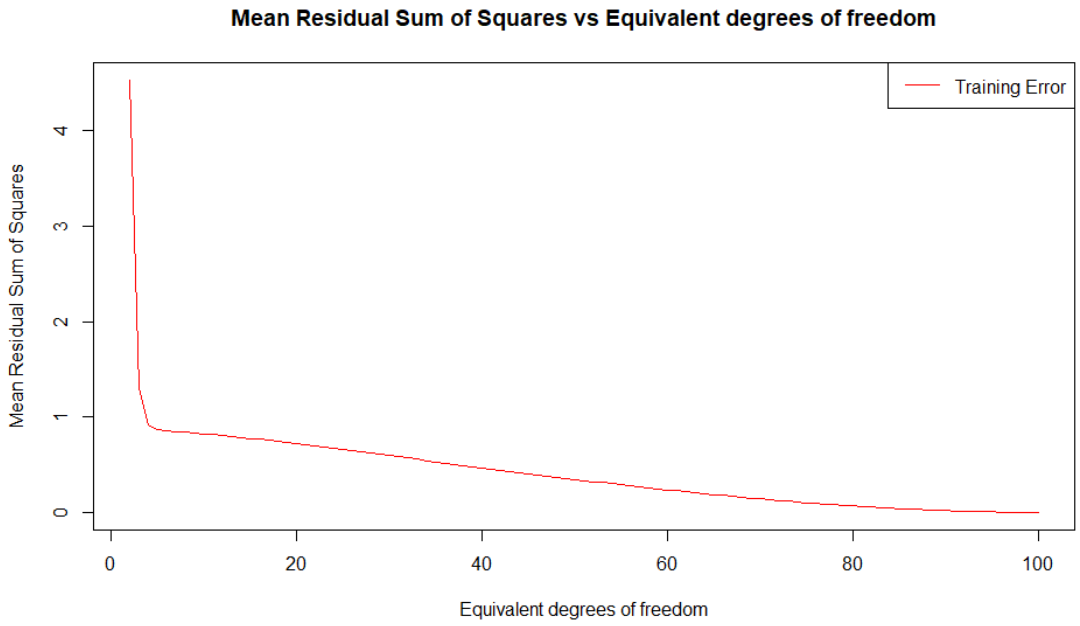


Looking at the scatterplot above it is clear the spline with df=2 is underfitting the data while the spline with df=20 is overfitting the data.

3c)

```r
dfvec = 2:n
results.df = data.frame(df=dfvec, MSE= matrix(0, nrow=n-1))


for (i in dfvec){
  #fit spline
  smooth.fit = smooth.spline(X, Y, df = i , all = TRUE)
  #calculate MSE and add to df
  results.df$MSE[i-1] = (sum((Y - smooth.fit$y)^2))/n

}

#plot using line
plot(MSE~df, results.df, type="n", main="Mean Residual Sum of Squares vs Equi
valent degrees of freedom", xlab="Equivalent degrees of freedom", ylab="Mean
Residual Sum of Squares")
```

```r
lines(MSE~df, results.df, lty = 1, col = "red", lwd = 1)
legend("topright",legend=c("Training Error"),col=c("red"),lty=rep(1))
```

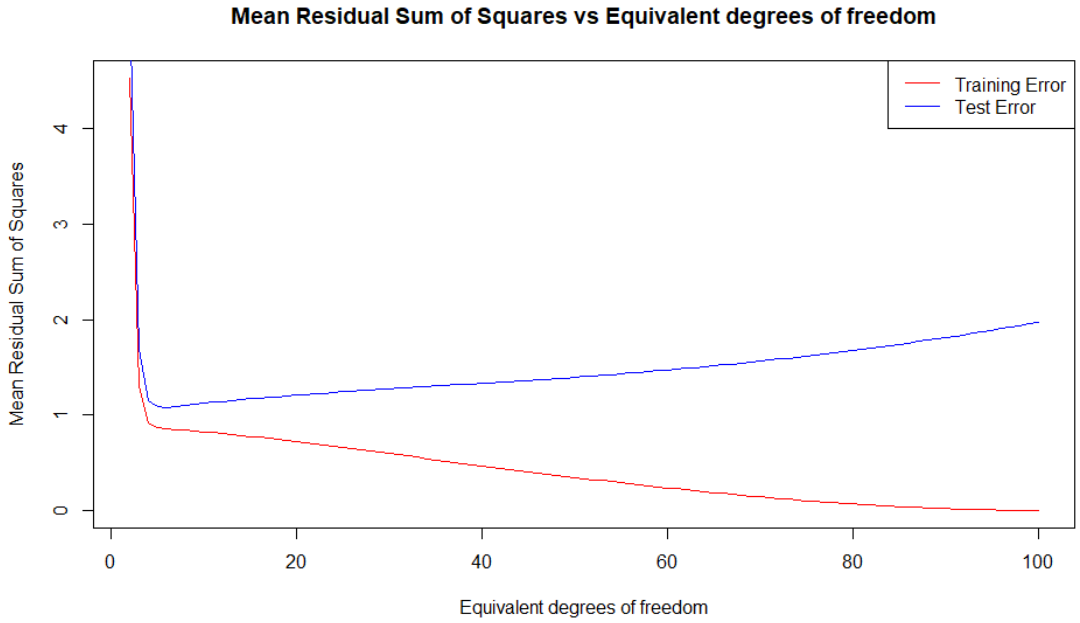


From the plot above it can be observed that as the equivalent degrees of freedom increases, the mean residuals sum of squares computed on the training set decreases. This is because as the equivalent degrees of freedom increases the model becomes more complex and tends to fit to the random variation in the data (overfit to the data).

3d)

```r
#model complexity vs prediction error
#generate new test data
n <- 100
X.test <- scale(3 * (1:n)/n, scale = FALSE)
myfun <- function(x)
  2 - x + 3*x*x
Y.test <- myfun(X.test) + rnorm(n)

dfvec = 2:n
results.train.df = data.frame(df=dfvec, MSE= matrix(0, nrow=n-1))
results.test.df = data.frame(df=dfvec, MSE= matrix(0, nrow=n-1))

#fit splines with training data and fit on test data
for (i in dfvec){
  #fit spline
  smooth.fit = smooth.spline(X, Y, df = i , all = TRUE)
  #calculate MSE and add to training df
  results.train.df$MSE[i-1] = (sum((Y - smooth.fit$y)^2))/n

  #predict test
```

```
  Y.pred = predict(smooth.fit, data=X.test, type="response")
  #calculate MSE
  results.test.df$MSE[i-1] = (sum((Y.test - Y.pred$y)^2))/n
}

#plot using line
plot(MSE~df, results.train.df, type="n", main="Mean Residual Sum of Squares v
s Equivalent degrees of freedom", xlab="Equivalent degrees of freedom", ylab=
"Mean Residual Sum of Squares")
lines(MSE~df,  lty = 1, col = "red", lwd = 1, results.train.df)
lines(MSE~df,  lty = 1, col = "blue", lwd = 1, results.test.df)
legend("topright",legend=c("Training Error", "Test Error"),col=c("red", "blue
"),lty=rep(1,2))
```

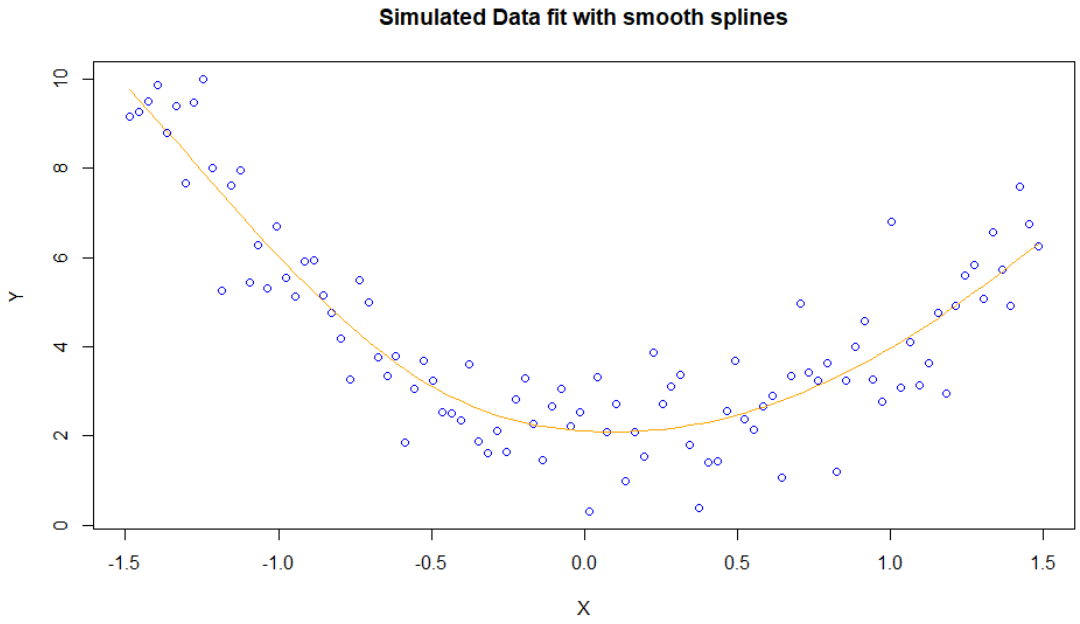


The plot above highlights that as the equivalent degrees of freedom (df) increases, the mean residuals sum of squares computed on the training dataset decreases. In comparison, the mean residuals sum of squares computed on the test dataset decreases up to an optimal df value and then begins to increase. This is because as the equivalent degrees of freedom increases the model becomes more complex. The model then begins to overfit to the training set and doesn't generalize well on the test set.

3e)

```
smooth.autogcv.fit = smooth.spline(X, Y, all = TRUE)
smooth.autogcv.fit

## Call:
## smooth.spline(x = X, y = Y, all.knots = TRUE)
##
## Smoothing Parameter  spar= 0.9508283  lambda= 0.005551461 (14 iterations)
```

```
## Equivalent Degrees of Freedom (Df): 5.125804
## Penalized Criterion (RSS): 86.49381
## GCV: 0.9609235

#plot a scatter plot with the smoother
plot(X, Y, col = "blue", main="Simulated Data fit with smooth splines")
lines(smooth.autogcv.fit, lty = 1, col = "orange", lwd = 1)
```



The optimal equivalent degrees of freedom set by GCV is approximately 5.1. This spline is a good fit and is a relatively better fit than the splines above as it seems to capture the underlying pattern in the data.

3f)

This section highlights the differences between good fitting, underfitting and overfitting models.

Quetions 3b and 3e show the visual differences between a good fitting, underfitting and overfitting model. These questions show that overfitting models fit to the random variation in the data (low bias and high variance) while underfitting models are not able to capture the underlying pattern in the data (high bias and low variance).

Furthermore, Questions 3c and 3d show how overfitting tends to occur when the complexity (equivalent degrees of freedom) of a model increases leading to a failure to generalise well. Moreover, these questions also show that characteristicly underfitting models have both high training and test error while overfitting models have low training error but high test error.