STAT2450: Assignment 5

Tasneem Hoque, Banner ID: B00841761

2022-06-17

Instructions

Code reuse

I encourage you to reuse the code in the lectures. Fill free to reuse R codes from the lectures of Module 5. However, if you need to copy-paste some code here, you may need to modify my codes to take into account the instructions given here. For example, should an instruction listed here tell you to use the name 'bootdist' for the variable storing a bootstrap distribution, please respect this and modify the original code of the lecture to comply with this requirement.

Data

In this assignment, we will use the following data. We assume that y is the response of interest and x is a predictor of y, and that we have 19 independent observations of x and y:

```
rm(list=ls())
x = c(1,1.5,2,3,4,4.5,5,5.5,6,6.5,7,8,9,10,11,12,13,14,15)
y = c(21.9,27.8,42.8,48.0,52.5,52.0,53.1,46.1,42.0,39.9,38.1,34.0,33.8,30.0,26.1,24.0,20.0
,11.1,6.3)
length(x)
```

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

Evaluation and dates

Two problems, 10 pts each. Due date: given on Brightspace.

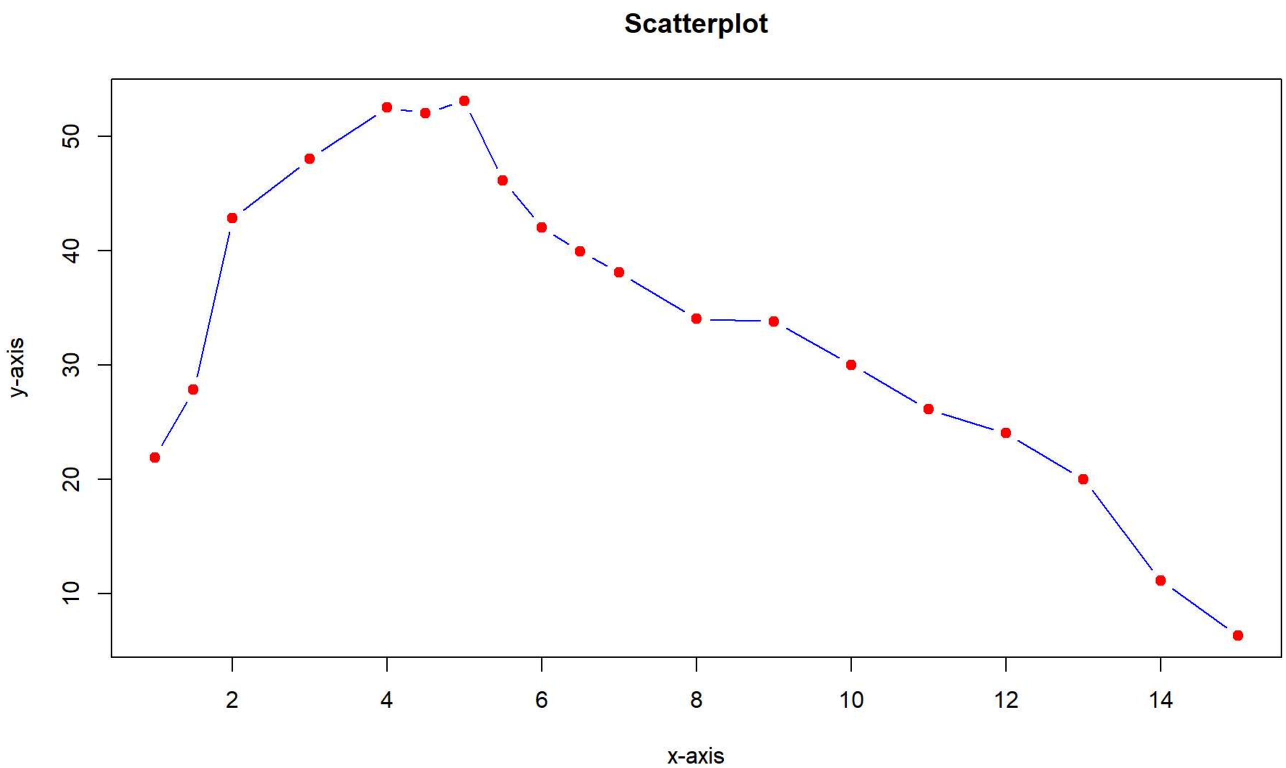
Problem 1 [10 pts]

The goal of this problem is to calculate a bootstrap distribution of the slope of the linear regression of y on x. Then we will use the bootstrap distribution to estimate the uncertainty of the slope.

Plot [1 pt]

Make a scatterplot of the x,y data. Overlay a continuous series of linear segments joining the points (hence your plot should show the raw data points and the line). Use a point choice (value of argument ‘pch’ of the function ‘plot’) equal to 19, and a red color for plotting the points. Use a blue color for plotting the lines.

```
plot(x,y,xlab="x-axis",ylab="y-axis",main="Scatterplot", pch = 19, col="blue",type="b")
points(x, y, col="red", pch=19)
```



Bootstrap distribution [5 pts]

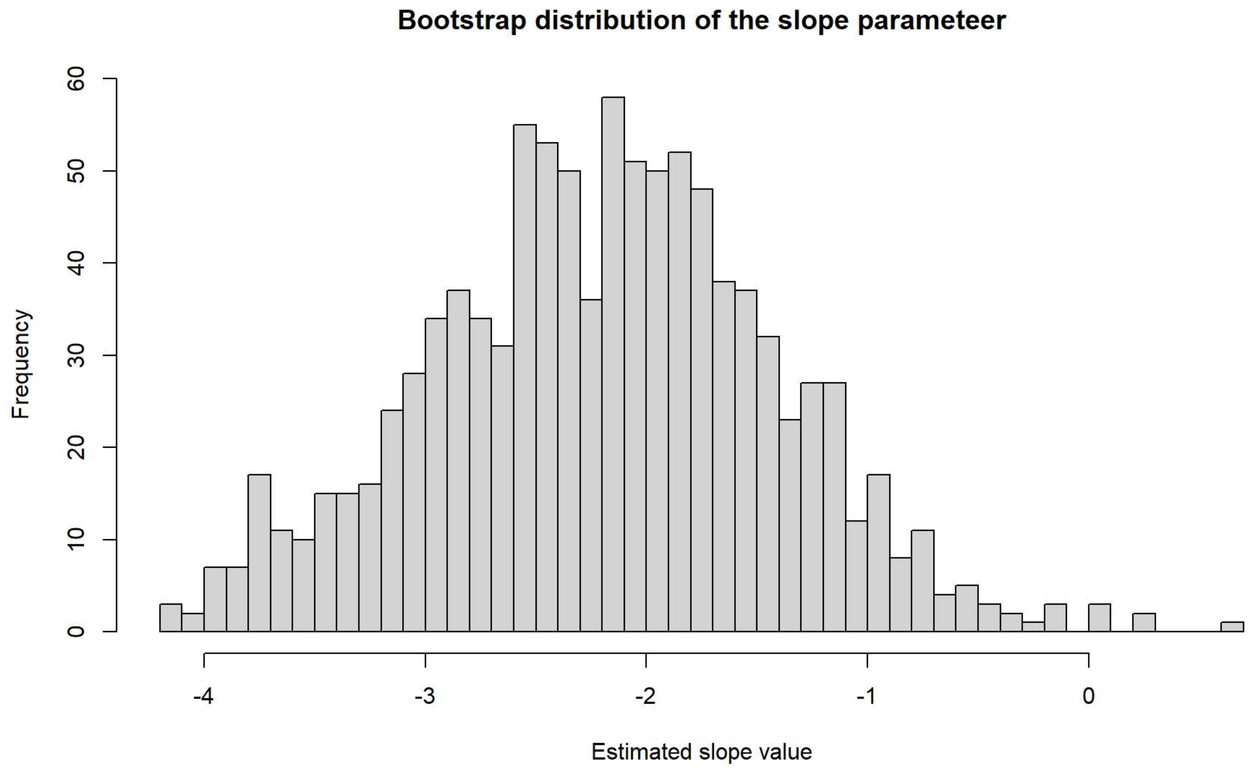
Write a code that produces a vector named ‘bootdist’ that contains a bootstrap distribution of 1000 values of the slope of the linear regression of y on x. You can mimic/reuse code used in the lectures. You will need to use a for loop, produce bootstrap sample of the data, apply the linear regression model, extract the slope and append it to the distribution.

```
set.seed(999123) # keep this line
n=length(x)

index=1:n
lmout0=lm(x~y)
htheta=coef(lmout0)[2]  # estimated slope

bootdist=NULL
Nboot=1000

for (i in 1:Nboot){
  index=sample(1:n,n,replace = TRUE)
  sample(1:n,n,replace = TRUE)
  xb=x[index]
  yb=y[index]
  lmout=lm(yb~xb)
  b=coef(lmout)[2]
  bootdist=c(bootdist,b) #calculate the stat
}
```

Histogram [1 pts]

Make a histogram of the bootstrap distribution ‘bootdist’. Use the main argument to add the title ‘Bootstrap distribution of the slope parameter’. Use the xlab argument to give add the x-axis label: ‘Estimated slope value’.

```
hist(bootdist,nclass=40,main="Bootstrap distribution of the slope parameteer", xlab="Estim
ated slope value")
```



Confidence interval [2 pts]

Use the quantile function to calculate a 92% percentile bootstrap confidence interval (make sure that you leave out a lower tail of probability 4% and an upper tail of probability 4%). Store this confidence interval in a variable named bootci and print this variable.

```
n=length(x) ; alpha=0.05
bootci=mean(x)+c(-1,1)*qt(1-alpha/2,n-1)*sd(x)/sqrt(n)
paste("(",round(bootci[1],3),",",round(bootci[2],3),")")
```

```
## [1] "( 5.191 , 9.335 )"
```

A 92% percentile interval for the slope is (5.191 , 9.335).

Variance of slope [1 pt]

Note that the variance of the bootstrap distribution provides an estimate pf the variance of the estimate of the slope of the regression:

$$Var(\hat{\beta}_1)$$

Now, use the 'var' function to calculate the variance of the bootstrap distribution. Store the value of the variance in a variable named 'varbootdist'.

```
varbootdist=var(bootdist)
round(varbootdist,3)
```

```
## [1] 0.612
```

Estimated variance of $\hat{\beta}_1$ is 0.612.

Problem 2 [10 pts]

In this problem, we will explore polynomial regression models.

Cubic model fitting [2 pts]

Fit a cubic polynomial to the data x,y. To do so, you will need to define and compute two extra variables x2 and x3 to store the square and the cube of x. Then use lm to fit a full cubic regression model:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

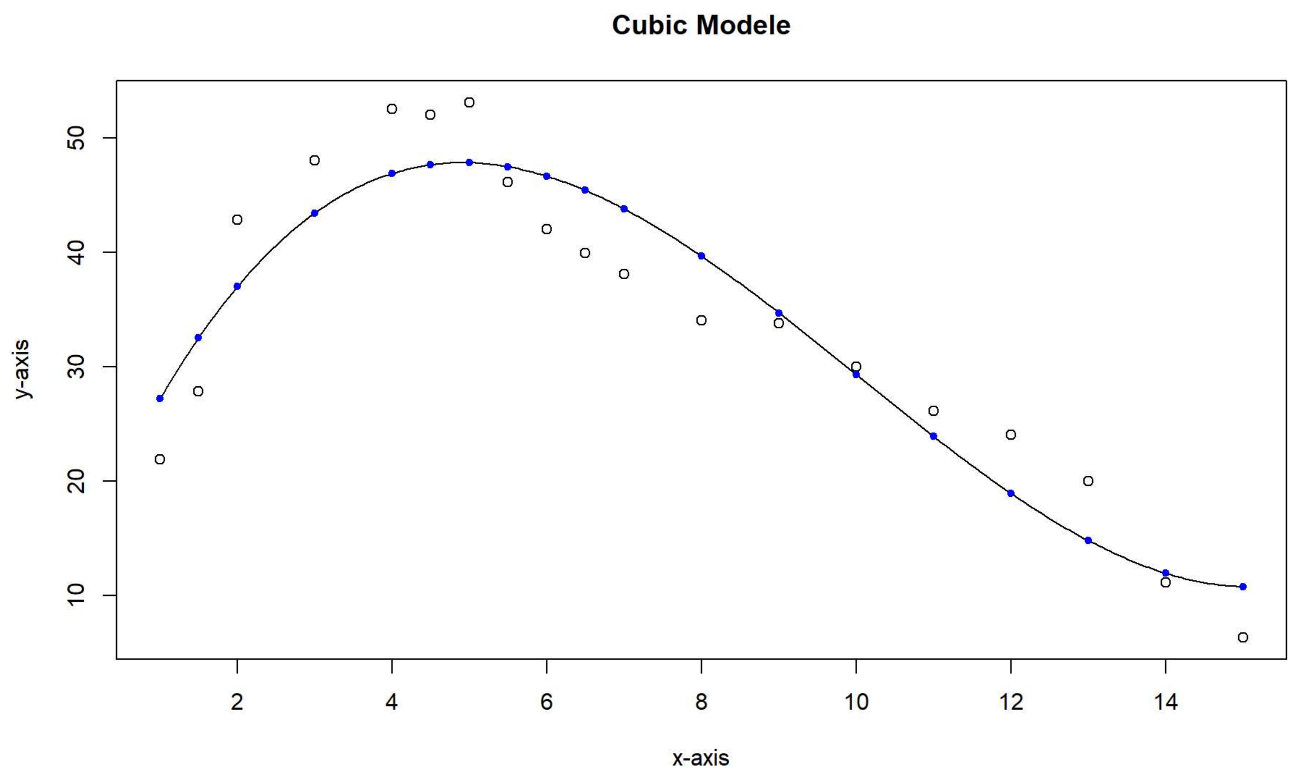
```
x2=x^2
lm2.out=lm(y~x+x2) # yep try to use x and the square of x as predictors!
coef(lm2.out) #least squares estimates
```

```
## (Intercept)           x          x2
##  29.2689914   5.2104091  -0.4686497
```

```
xplot=seq(min(x),max(x),length.out=300)
fits2=predict(lm2.out,data.frame(x=xplot,x2=xplot^2))
ypred2=predict(lm2.out) #predicted values


x3=x^3
lm3.out=lm(y~x+x2+x3) # even richer nonlinearities adding the cube of x as predictor
coef(lm3.out) #least squares estimates
```

```
## (Intercept)           x          x2          x3
## 13.56504744 15.64205734 -2.11031628  0.07033125
```

```
xplot=seq(min(x),max(x),length.out=300)
fits3=predict(lm3.out,data.frame(x=xplot,x2=xplot^2,x3=xplot^3))
ypred3=predict(lm3.out) #predicted values

plot(x,y,xlab="x-axis",ylab="y-axis", main="Cubic Modele")
lines(list(x=xplot,y=fits3))
points(x,ypred3,col="Blue",pch=20)
```



Produce a summary of the model:

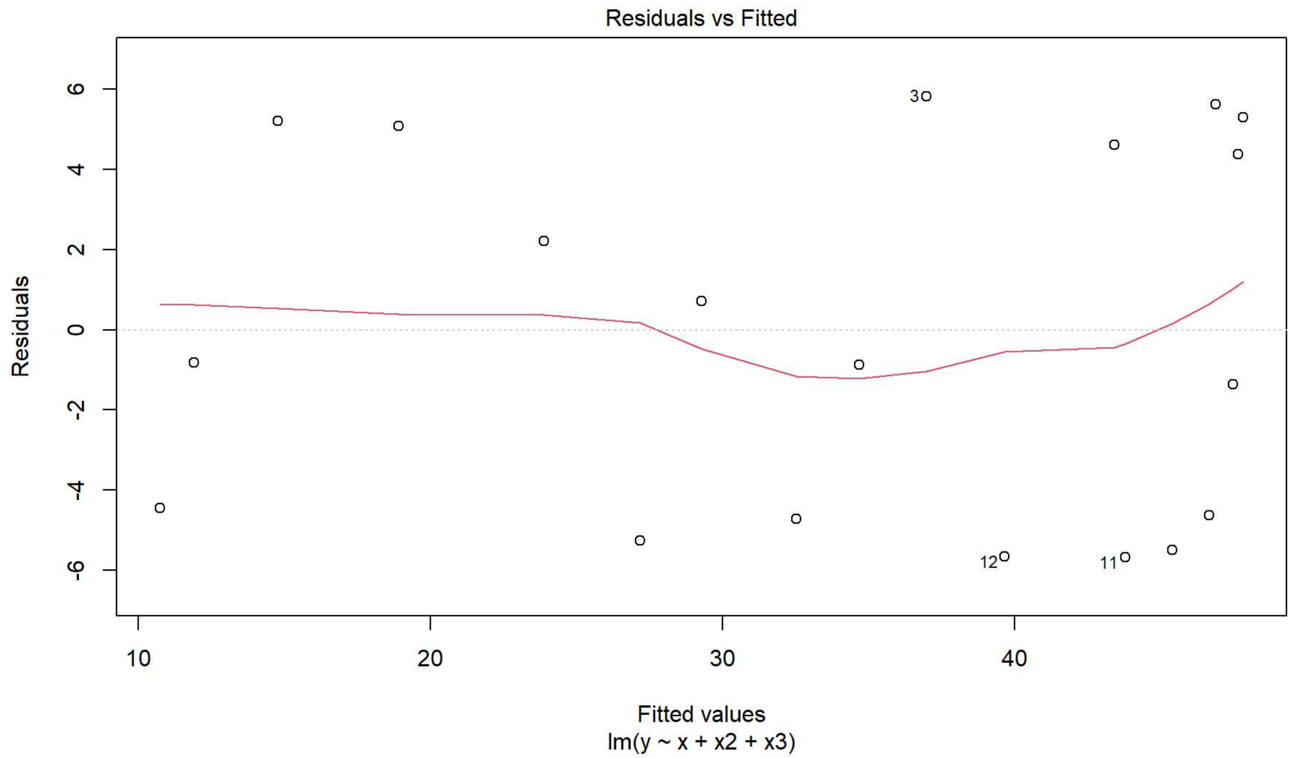
```
summary(lm3.out)
```

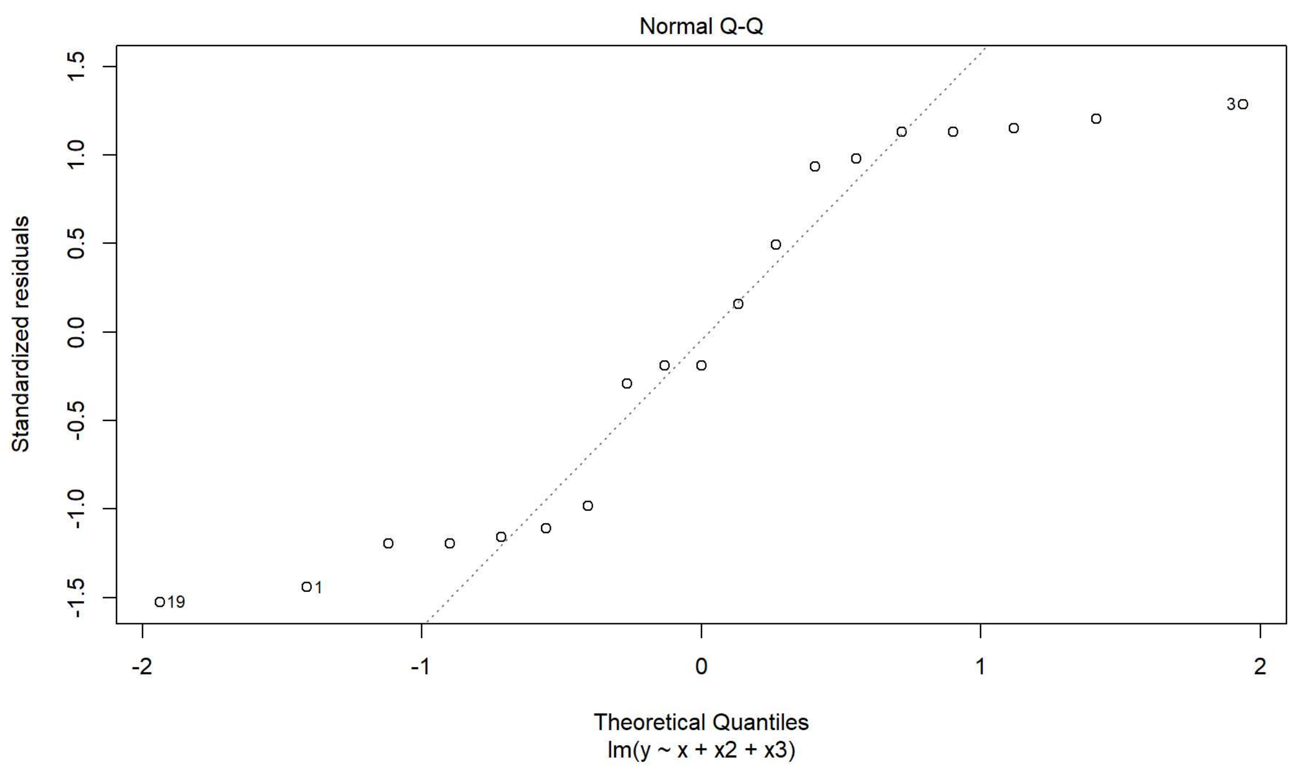
```
##
## Call:
## lm(formula = y ~ x + x2 + x3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.6776 -4.6774 -0.8208  4.8430  5.8295
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.56505    5.76242   2.354 0.032620 *
## x           15.64206    3.05385   5.122 0.000125 ***
## x2          -2.11032    0.45116  -4.678 0.000298 ***
## x3           0.07033    0.01909   3.684 0.002210 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.042 on 15 degrees of freedom
## Multiple R-squared:  0.8884, Adjusted R-squared:  0.8661
## F-statistic: 39.81 on 3 and 15 DF,  p-value: 2.214e-07
```

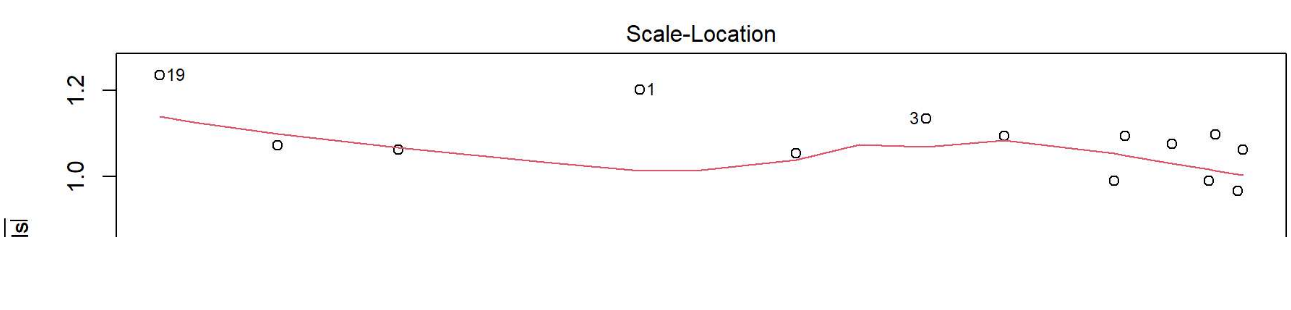
Diagnostic plot [1 pt]

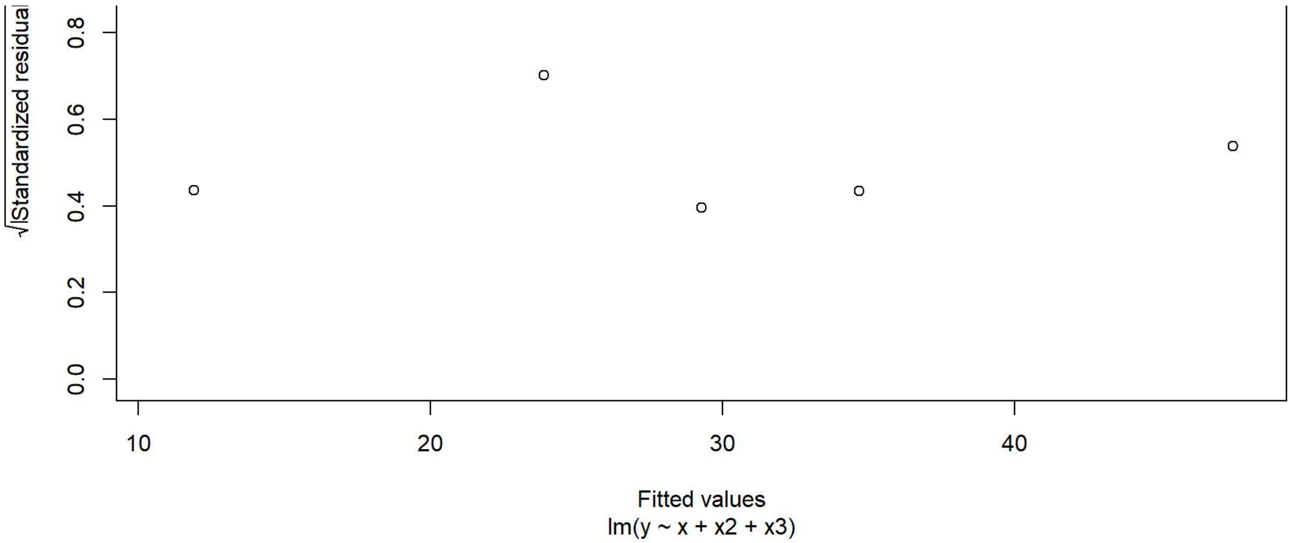
Produce a plot showing 4 diagnostics of the validity of the model. Your code should not be more than one line long and should show 4 plots: 1. Residual vs Fitted 2. Normal Q-Q plot 3. Scale-location plot 4. Residual vs Leverage

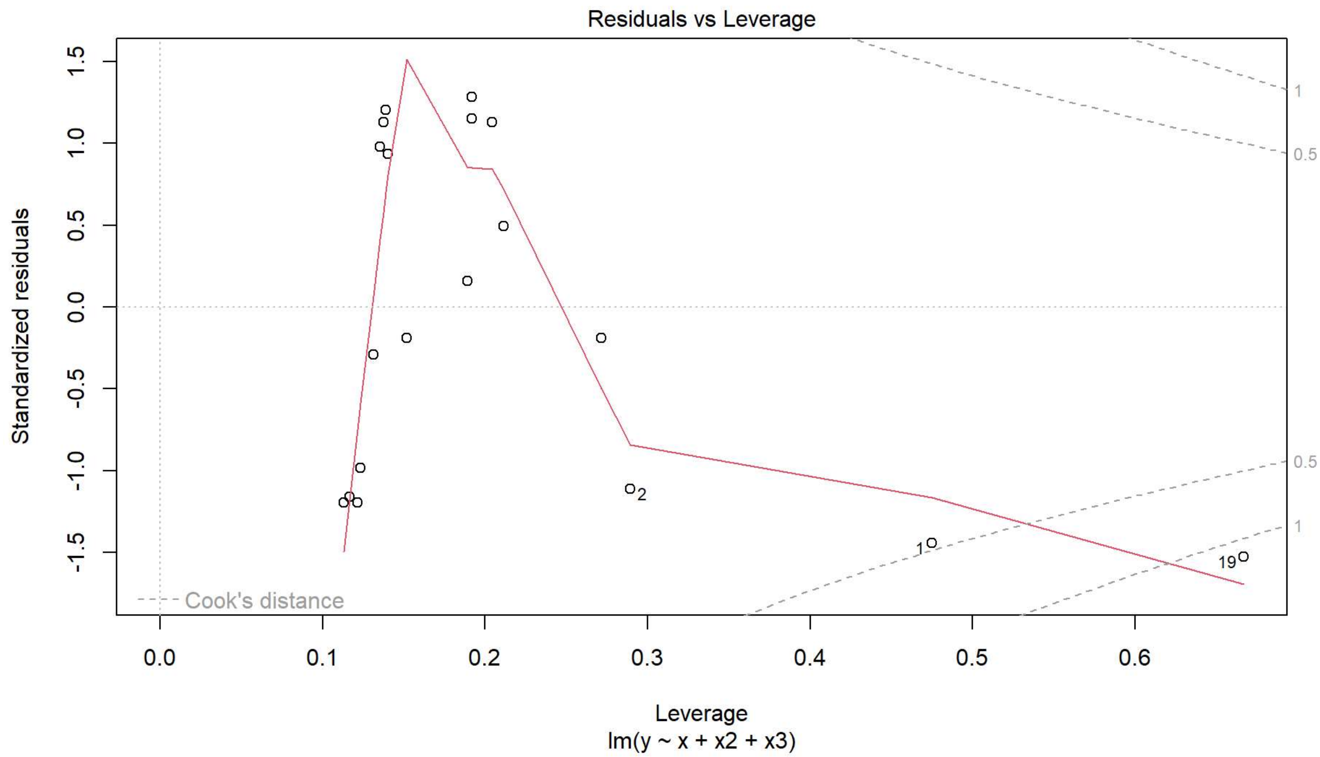
```
plot(lm3.out)
```


Residuals vs Fitted
Residuals
Fitted values
lm(y ~ x + x2 + x3)
3
12
11


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(y ~ x + x2 + x3)
3
19
1


Scale-Location
19
1
3





Compute the SSE [2 pts]

Proceed as in lecture 2 to compute the sum of squared errors (SSE) for model lm3.out. You will have to apply the function predict to calculate a vector ‘ypred3’ containing the values predicted by the fitted cubic model at the observed values of x (hence no need to use the argument newdata here). Then compute and store the SSE in a variable named SSE3 and print the value of SSE3.

```
ypred3=predict(lm3.out) #predicted values
resids3=y-ypred3  #residuals
SSE3=sum(resids3^2) #error some of squares
print(paste("Error sum of squares of quadratic model is ",SSE3))
```

```
## [1] "Error sum of squares of quadratic model is  381.259972630481"
```

Prediction [2 pts]

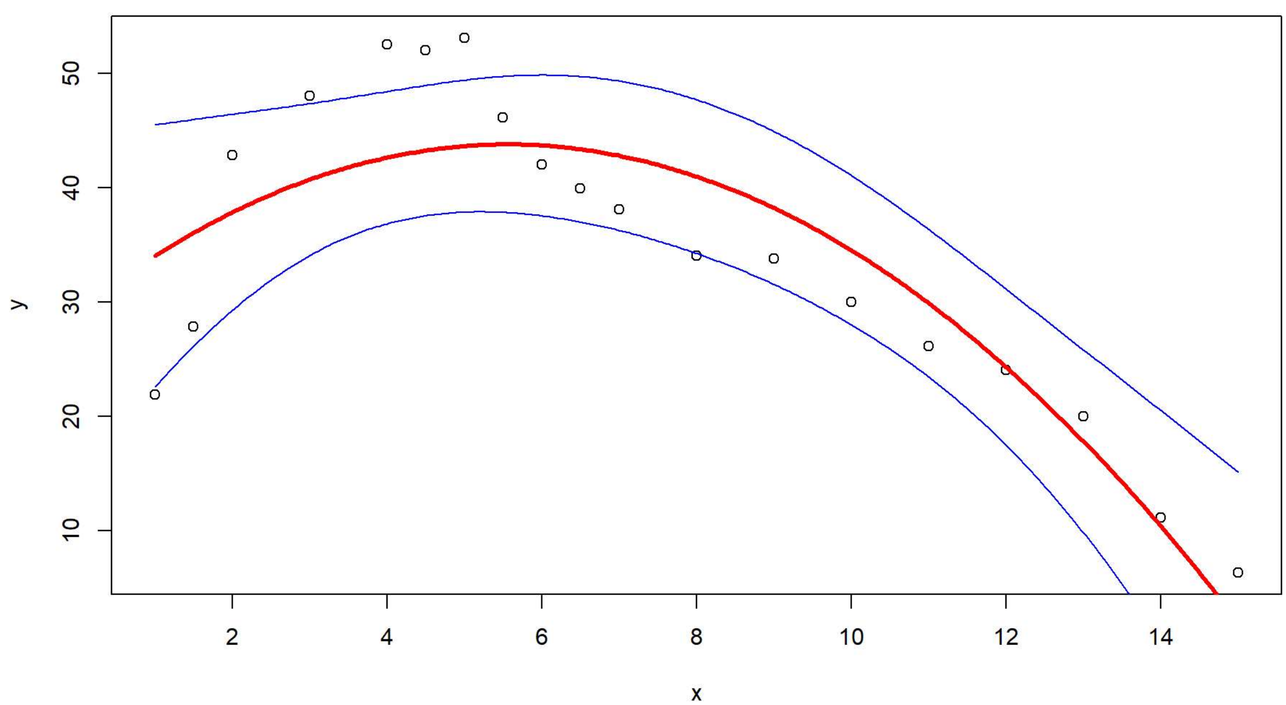
You will now write a code that plots the predictions of the cubic model on a grid of regularly spaced values of x. Use the seq function to define a grid of 300 points that spans the interval from the minimum to the maximum value of x.

Then apply the predict function to compute the predicted response AND use the interval argument and the level argument of 'predict' to request the calculation of confidence interval of 99%.

Then call the plot function on x and y, and overlay 3 lines to show the predictions in red and the confidence interval in blue.

Note that you can reuse the code provided in the FINAL version of M5-L2 (see BS).

```
grid=seq(min(x),max(x),length.out=300)
pmodel <- lm3.out
predicted.intervals <- predict(pmodel,data.frame(x=x),interval='confidence',level=0.99)
xplot=seq(min(x),max(x),length.out=300)

predicted.intervals <-predict(lm2.out,newdata=data.frame(x=xplot,x2=xplot^2),interval='con
fidence',level=0.99)

plot(x,y)
lines(xplot,predicted.intervals[,1],col='red',lwd=3)
lines(xplot,predicted.intervals[,2],col='blue',lwd=1)
lines(xplot,predicted.intervals[,3],col='blue',lwd=1)
```



Quintic model [2 pts]

Define extra variables , x4 and x5, that contain the fourth and fifth power of x. Use them to fit a polynomial model of degree 5 to explain the response y. Store this quintic (i.e. degree 5) regression model in an object called lmout.5.

```
x4=x^4
lm4.out=lm(y~x+x2+x3+x4) # even richer nonlinearities adding the cube of x as predictor
coef(lm4.out) #least squares estimates
```

```
##  (Intercept)           x          x2          x3          x4
## -10.86992782  39.50881125  -8.36120385   0.66693482  -0.01871265
```

```
xplot=seq(min(x),max(x),length.out=300)
fits4=predict(lm4.out,data.frame(x=xplot,x2=xplot^2,x3=xplot^3,x4=xplot^4))
ypred4=predict(lm4.out) #predicted values

x5=x^5
lm5.out=lm(y~x+x2+x3+x+x5) # even richer nonlinearities adding the cube of x as predictor
coef(lm5.out) #least squares estimates
```

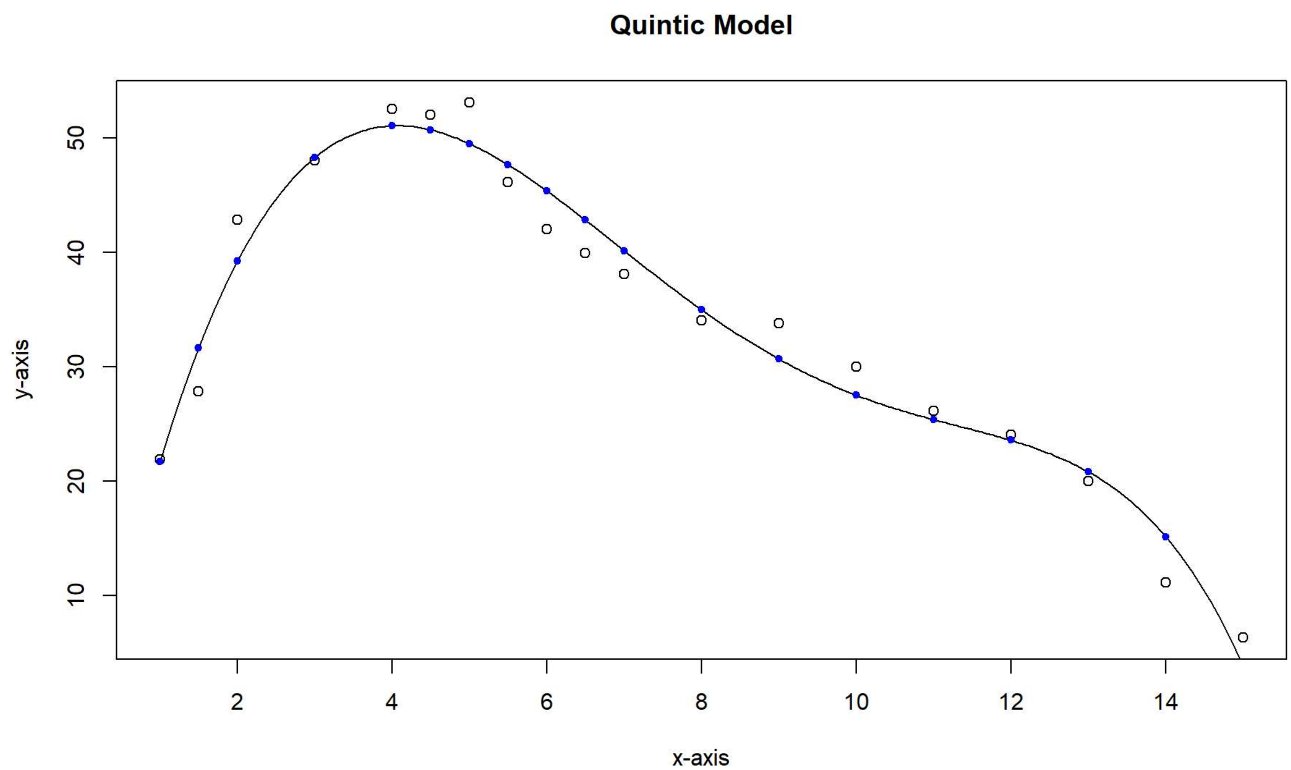
```
##   (Intercept)             x            x2            x3            x5
## -6.5094625637 34.2872476425 -6.4878709215  0.3852897214 -0.0004537483
```

```
xplot=seq(min(x),max(x),length.out=300)
fits5=predict(lm5.out,data.frame(x=xplot,x2=xplot^2,x3=xplot^3,x4=xplot^4,x5=xplot^5))
ypred5=predict(lm5.out) #predicted values
```

Significance of quintic model [1 pt]

Produce a summary of the quintic regression model.

```
plot(x,y,xlab="x-axis",ylab="y-axis", main="Quintic Model")
lines(list(x=xplot,y=fits5))
points(x,ypred5,col="Blue",pch=20)
```



Which of alternatives a. or b. does the summary of the quintic model imply:

a. the regression coefficient of the highest order monomial term (x^5) IS significantly different from zero.
b. the regression coefficient of the highest order monomial term (x^5) IS NOT significantly different from zero.

Note: as usual, we will consider that any test of hypothesis with a p-value at most equal to 0.05 is significant.

The answer is b.