Tanmoy Das
Ph.D. in Industrial Engineering
Feb 02, 2020

Question 1

Describe the estimators for linear regression that are implemented in the lmRob function (in R)

lmRob performs a robust linear regression. The estimators of lmRob are as follow: coefficients, T.coefficients, residuals, fitted.values, and many others. The prominent estimators are considered in this discussion.

Coefficients: vector of coefficients for the robust regression; Residuals: the residual vector corresponding to the estimates returned in coefficients; fitted.values: the fitted values corresponding to the estimates returned in coefficients.

According to the lmRob function implemented in a RoadKills_edited dataset, the estimate of intercept and D.PARK are 40.36 and -0.00161, respectively. Since the p-value for both cases are below 0.05, it can be justified that both parameters are statistically significant. The M-estimator, where M for Maximum Likelihood and the LS-estimate are 2.914 and 6.97, respectively.

```
##---- Load relevant libraries
library(robust)   # Required library for lmRob function

data1 <- read.table("../Data/RoadKills_edited.txt", header=TRUE)
head(data1)

##   Sector      X      Y BufoCalamita TOT.N S.RICH OPEN.L  OLIVE MONT.S
## 1      1 260181 256546            5    70      3 22.684 60.333  0.000
## 2      2 259914 256124            1    80      4 24.657 40.832  0.000
## 3      3 259672 255688           40    90      6 30.121 23.710  0.258
## 4      4 259454 255238           27    90      5 50.277 14.940  1.783
## 5      5 259307 254763           67    90      4 43.609 35.353  2.431
## 6      6 259189 254277           56    90      7 31.385 17.666  0.000
##     MONT POLIC SHRUB  URBAN WAT.RES L.WAT.C L.D.ROAD L.P.ROAD D.WAT.RES
## 1  0.653 4.811 0.406  7.787   0.043   0.583 3330.189    1.975   252.113
## 2  0.161 2.224 0.735 27.150   0.182   1.419 2587.498    1.761   139.573
## 3 10.918 1.946 0.474 28.086   0.453   2.005 2149.651    1.250    59.168
## 4 26.454 0.625 0.607  0.831   0.026   1.924 4222.983    0.666   277.842
## 5 11.330 0.791 0.173  2.452   0.000   2.167 2219.302    0.653   967.808
## 6 43.678 0.054 0.325  2.730   0.039   2.391 1005.629    1.309   560.000
##   D.WAT.COUR   D.PARK N.PATCH  P.EDGE L.SDI
## 1    735.000  250.214     122 553.936 1.801
## 2    134.052  741.179      96 457.142 1.886
## 3    269.029 1240.080      67 432.360 1.930
## 4     48.751 1739.885      63 421.292 1.865
## 5    126.102 2232.130      59 407.573 1.818
## 6    344.444 2724.089      49 420.289 1.799

##---- Data Visualization
# plot(data1$D.PARK,data1$TOT.N,xlab="Distance to park", ylab="Road kills") # Doesn't look linear.
# Lets plot in the next chunk of code!
# lmRob modeling
model.lmRob<-lmRob(TOT.N~D.PARK,data=data1)
summary(model.lmRob)
```

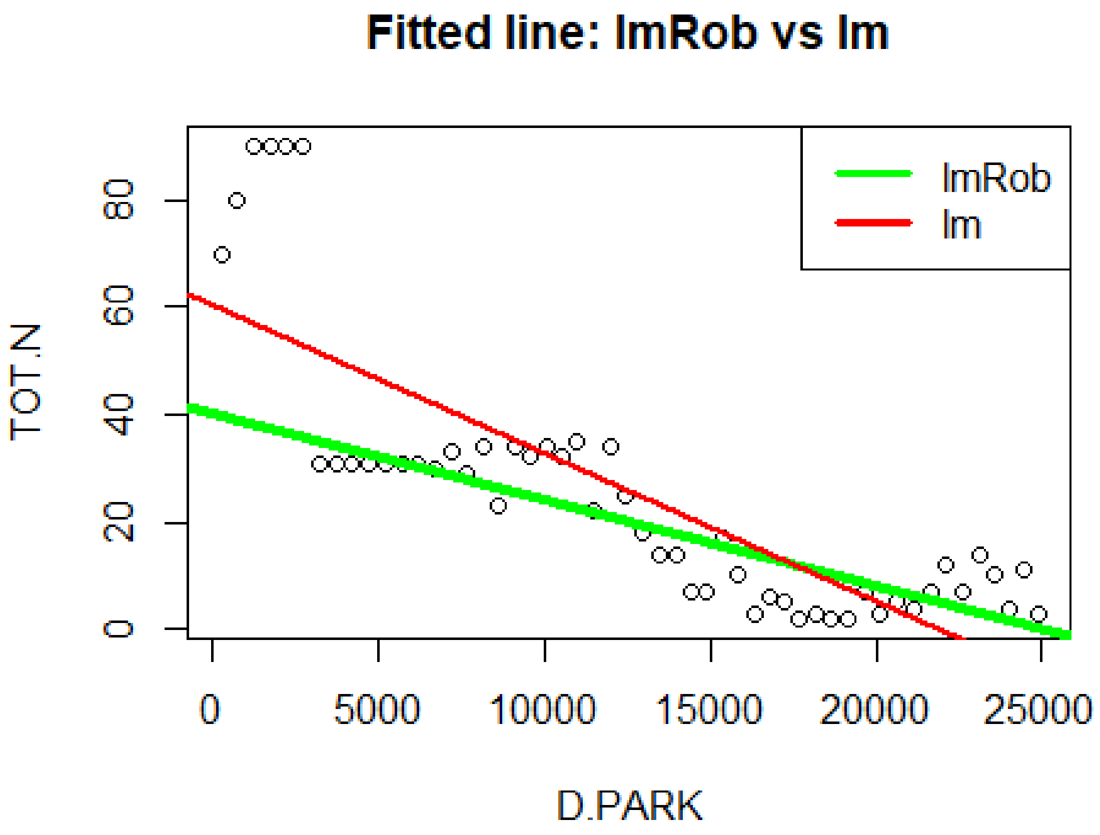
```
## Call:
## lmRob(formula = TOT.N ~ D.PARK, data = data1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.0479  -4.0277   0.3711   7.8236  54.0197
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 40.3696058  2.5716771  15.698  < 2e-16 ***
## D.PARK      -0.0016113  0.0001657  -9.725 4.07e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.602 on 50 degrees of freedom
## Multiple R-Squared: 0.521
##
## Test for Bias:
##             statistic p-value
## M-estimate      2.915 0.23287
## LS-estimate     6.565 0.03753
```

Contrast these with what is implemented by the lm function.

lm is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance.

In light of the fitted lines of lmRob and lm: lmRob can take care of outliers, as seen in the left-hand side of the figure whereas lm is extremely sensitive to the outliers, eventually performed poorly. In other words, lm function (the fitted line is represented by red) is vulnerable to outliners, whereas lmRob can ignore or conduct some treatments on outliers. To conclude, lmRob outperforms lm in various aspects, including handling outliners.

```
# Linear Modeling
model.lm <- lm(TOT.N~D.PARK,data=data1)
summary(model.lm)

##
## Call:
## lm(formula = TOT.N ~ D.PARK, data = data1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.698  -9.762  -2.865   7.263  37.251
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 60.2416287  4.0604309  14.836  < 2e-16 ***
## D.PARK      -0.0027505  0.0002779  -9.897 2.28e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.54 on 50 degrees of freedom
## Multiple R-squared:  0.662,  Adjusted R-squared:  0.6553
## F-statistic: 97.94 on 1 and 50 DF,  p-value: 2.277e-13

# Plotting lmRob & lm fit
plot(data1$D.PARK,data1$TOT.N, main = "Fitted line: lmRob vs lm", xlab="D.PARK", ylab="TOT.N")
abline(model.lmRob, col = "green", lwd=4)
abline(model.lm, col = "red", lwd=2)
# Add a legend
legend("topright", c("lmRob", "lm"), col=c("green", "red"), lwd=3)
```



Explain how to proceed with data analysis when you have good reason to believe that a linear model is reasonable for your data but that there may well be data recording errors.

The erroneous records or entries should be deleted (or corrected, if possible). After removing the inaccurate data, the model will be refitted by the remaining dataset. Furthermore, an insightful analysis would be conducted why the data recording errors are occurred.

By contrast, a careful consideration should be taken as the alternate view might be plausible as well (not in the given scenario of the assignment but other data science problems): Sometimes, we throw out impeccable data when we ought to throw out questionable models.

Question 2

Explain how the Akaike information criterion (AIC) is computed for a linear model

In Linear Model, the formula to calculate AIC is given below: AIC = -2(log-likelihood) + 2K Where K is the number of model parameters (the number of variables in the model plus the intercept). Log-likelihood is a measure of model fit. The higher the number, the better the fit. This is usually obtained from statistical output. Generic function calculating Akaike's 'An Information Criterion' for one or several fitted model objects for which a log-likelihood value can be obtained, according to the formula -2*log-likelihood* + *k*npar, where npar represents the number of parameters in the fitted model, and k = 2 for the usual AIC.

The value of AIC computed for the Linear Model in the previous question is 429.94.

How AIC is commonly utilized for model (or variable) selection purposes.

Given a collection of models for the data, AIC estimates the quality of each model, relative to each of the other models. AIC=2[-l(β^)+p] P is the penalty term. Estimation of the estimators, beta_hat L = likelihood

When comparing multiple models, the model containing a minimum value of AIC is chosen. The value of AIC computed for the Linear Model, and Multiple Linear Model in the previous question are 429.94 and 426.79, respectively. Therefore, in this case, multiple linear regression might be a better choice.

Note: Further scrutiny is needed, which is out of the scope of this assignment. In addition, regardless of the AIC value, for the dataset considered in Question 1, Robust Linear Regression is better since it can deal with outliers in the dataset.

```
# Calculating AIC for Linear Model from previous question
AIC_lm = AIC(model.lm)

# Lets quickly develop a multiple linear regression model to compare it with lm model at hand.
model.mlm <- lm(TOT.N~D.PARK + MONT, data=data1)
AIC_mlm = AIC(model.mlm) # Calculating AIC
```

Question 3

Consider the negative binomial distribution.

3(a) Write down its density function.

In a sequence of independent Bernoulli(p) trials, the random variable X denotes the trial at which the r th success occurs, where r is a fixed integer. Then, The density function is as below when X has a negative binomial(r, p) distribution.:

$$P(X = x|r,p) = \binom{x-1}{r-1} p^r (1-p)^{x-r}, x = r, r+1, \ldots (1)$$

3(b) Plot the density curves from a negative binomial distribution for a range of values of it's two parameters.

We need to consider the two parameters of a negative binomial distribution, which means we will set a range of values for the parameter 1: size (number of failures), and the parameter 2: p (success probability in each experiment).

```r
# 3(b)
# Param 1: size, number of failure
# negative_binom <- rnbinom(n, size, prob, mu)
# n = number of observation
# Density Curve for size = 1 in rnbinom
nb <- rnbinom(200, size = 1, prob = .5)
hist1 <- hist(nb, breaks = 20, plot = FALSE)
xfit<-seq(min(nb),max(nb))
yfit<-dnorm(xfit,mean=mean(nb),sd=sd(nb))
yfit <- yfit*diff(hist1$mids[1:2])*length(nb)

# Desity Curve for size=5,10,20,50
size_i = c(1, 5, 10, 20, 50)
neg_binom = rnbinom(200, size = size_i[2],prob = .5)
hist_neg5 <- hist(neg_binom, breaks = 20, plot = FALSE)
xfit5<-seq(min(neg_binom),max(neg_binom))
yfit5<-dnorm(xfit5,mean=mean(neg_binom),sd=sd(neg_binom))
yfit5 <- yfit5*diff(hist_neg5$mids[1:2])*length(neg_binom)
neg_binom = rnbinom(200, size = size_i[3],prob = .5)
hist_neg10 <- hist(neg_binom, breaks = 20, plot = FALSE)
xfit10<-seq(min(neg_binom),max(neg_binom))
yfit10<-dnorm(xfit10,mean=mean(neg_binom),sd=sd(neg_binom))
yfit10 <- yfit10*diff(hist_neg10$mids[1:2])*length(neg_binom)
neg_binom = rnbinom(200, size = size_i[4],prob = .5)
hist_neg20 <- hist(neg_binom, breaks = 20, plot = FALSE)
xfit20<-seq(min(neg_binom),max(neg_binom))
yfit20<-dnorm(xfit20,mean=mean(neg_binom),sd=sd(neg_binom))
yfit20 <- yfit20*diff(hist_neg20$mids[1:2])*length(neg_binom)
neg_binom = rnbinom(200, size = size_i[5],prob = .5)
hist_neg50 <- hist(neg_binom, breaks = 20, plot = FALSE)
xfit50<-seq(min(neg_binom),max(neg_binom))
yfit50<-dnorm(xfit50,mean=mean(neg_binom),sd=sd(neg_binom))
yfit50 <- yfit50*diff(hist_neg50$mids[1:2])*length(neg_binom)

# Plotting all the curves
plot(xfit, yfit, main="Density Curves of Negative Binomial: Size is varied", xlab = "x", ylab="Density", type="l",
col="black", lwd=3, xlim=c(min(xfit, xfit5, xfit10, xfit20, xfit50), max(xfit, xfit5, xfit10, xfit20, xfit50)), ylim=c(
min(yfit, yfit5, yfit10, yfit20, yfit50), max(yfit, yfit5, yfit10, yfit20, yfit50)))
# Add a legend
legend("topright", c("Size = 1", "Size = 5", "Size = 10", "Size = 20","Size = 50"), col=c("black", "blue", "green",
"red", "blue"), lwd=3)
lines(xfit5, yfit5, col="blue", lwd=3)
lines(xfit10, yfit10, col="green", lwd=3)
lines(xfit20, yfit20, col="red", lwd=4)
lines(xfit50, yfit50, col="blue", lwd=5)
```

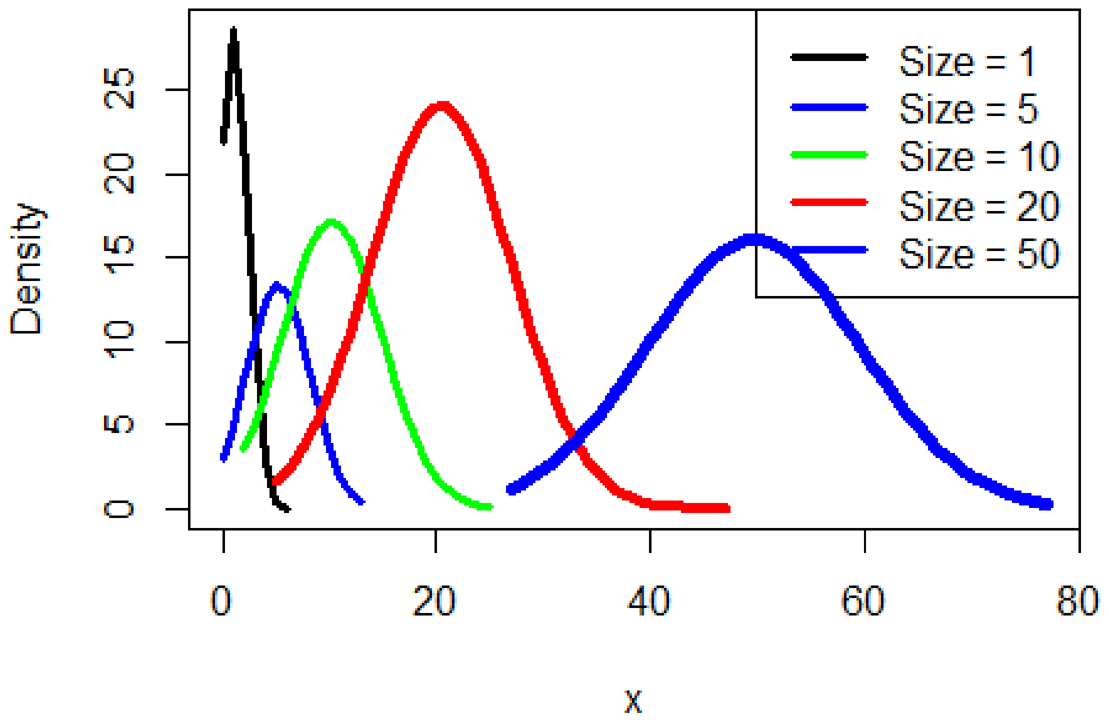


Figure: Density Curves when the size is variable

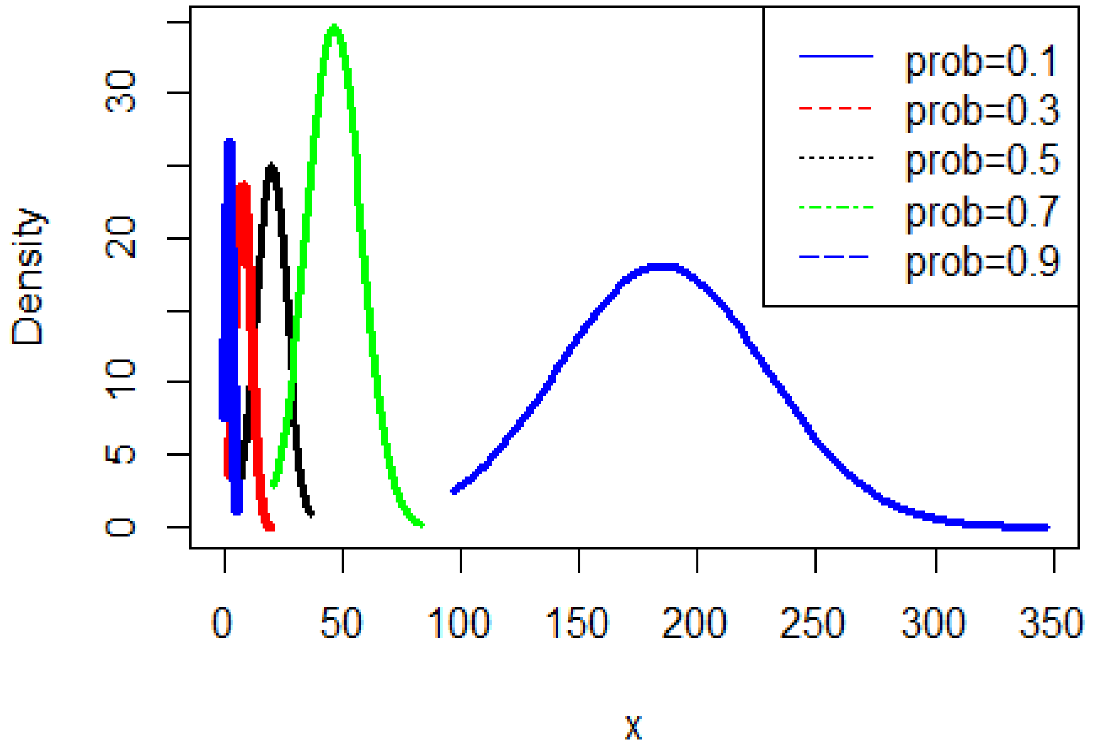


Figure: Density Curves when the probability is variable

```
# 3(b)
# Changing parameter 2: p, success probability
# negative_binom <- rnbinom(n, size, prob, mu)
# n = number of observation

# Density Curve for prob=0.5 in rnbinom
nb <- rnbinom(200, size = 20, prob = .5)
hist1 <- hist(nb, breaks = 20, plot = FALSE)
xfit<-seq(min(nb),max(nb))
yfit<-dnorm(xfit,mean=mean(nb),sd=sd(nb))
yfit <- yfit*diff(hist1$mids[1:2])*length(nb)
# Density Curve for prob=0.1, 0.3, 0.7, 0.9 in rnbinom
prob_i = c(.1, .3, .5, .7, .9)
neg_binom = rnbinom(200, 20,prob = prob_i[1])
hist_neg5 <- hist(neg_binom, breaks = 20, plot = FALSE)
xfit5<-seq(min(neg_binom),max(neg_binom))
yfit5<-dnorm(xfit5,mean=mean(neg_binom),sd=sd(neg_binom))
yfit5 <- yfit5*diff(hist_neg5$mids[1:2])*length(neg_binom)
neg_binom = rnbinom(200, 20,prob = prob_i[2])
hist_neg10 <- hist(neg_binom, breaks = 20, plot = FALSE)
xfit10<-seq(min(neg_binom),max(neg_binom))
yfit10<-dnorm(xfit10,mean=mean(neg_binom),sd=sd(neg_binom))
yfit10 <- yfit10*diff(hist_neg10$mids[1:2])*length(neg_binom)
neg_binom = rnbinom(200, 20,prob = prob_i[4])
hist_neg20 <- hist(neg_binom, breaks = 20, plot = FALSE)
xfit20<-seq(min(neg_binom),max(neg_binom))
yfit20<-dnorm(xfit20,mean=mean(neg_binom),sd=sd(neg_binom))
yfit20 <- yfit20*diff(hist_neg20$mids[1:2])*length(neg_binom)
neg_binom = rnbinom(200, 20,prob = prob_i[5])
hist_neg50 <- hist(neg_binom, breaks = 20, plot = FALSE)
xfit50<-seq(min(neg_binom),max(neg_binom))
yfit50<-dnorm(xfit50,mean=mean(neg_binom),sd=sd(neg_binom))
yfit50 <- yfit50*diff(hist_neg50$mids[1:2])*length(neg_binom)
# Plotting all the curves
plot(xfit, yfit, main="Density Curves of Negative Binomial: varying probability", xlab = "x", ylab="Density", typ
e="l", col="black", lwd=3, xlim=c(min(xfit, xfit5, xfit10, xfit20, xfit50), max(xfit, xfit5, xfit10, xfit20, xfit50)), y
lim=c(min(yfit, yfit5, yfit10, yfit20, yfit50), max(yfit, yfit5, yfit10, yfit20, yfit50)))
# Add a legend
legend("topright", c("prob=0.1", "prob=0.3", "prob=0.5", "prob=0.7","prob=0.9"), lty=1:5, col=c("blue",  "red", "
black", "green","blue"))
lines(xfit5, yfit5, col="blue", lwd=3)
lines(xfit10, yfit10, col="green", lwd=3)
lines(xfit20, yfit20, col="red", lwd=4)
lines(xfit50, yfit50, col="blue", lwd=5)
```

(c) What distribution do you arrive at if the dispersion parameter is taken to be 1?

Poisson Distribution will be achieved when the dispersion parameter is taken to be 1.

Geometric distribution is also possible to attain when other values of the parameter k is taken into consideration.

3(d) How is the dispersion parameter estimated in the function glm.nb?

The maximum likelihood estimation is utilized to calculate the dispersion parameter in glm.nb

When the sample mean is equal to the sample variance, the method of moments estimator becomes infinity. By contrast, the maximum likelihood estimator ceases to exist when the sample variance is less than the sample mean. It is also important to note that the dispersion parameter in binomial, Poisson and negative binomial with known shape parameter is 1 by definition of the exponential family, not an assumption.

$Var = \mu + \mu^2/k$ where k is the dispersion parameter.

```
# 3(d)
library(MASS)    # to import glm.nb package

## Warning: package 'MASS' was built under R version 3.6.2

library(foreign)  # to read Stata files

## Warning: package 'foreign' was built under R version 3.6.2

dat <- read.dta("../Data/nb_data.dta")
mod.nb <- glm.nb(daysabs ~ math + prog, data = dat)
summary(mod.nb)

##
## Call:
## glm.nb(formula = daysabs ~ math + prog, data = dat, init.theta = 1.022168288,
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1294  -1.0013  -0.3686   0.2578   2.4153
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.487288   0.225281  15.480  < 2e-16 ***
## math        -0.006737   0.002478  -2.719  0.00655 **
## prog        -0.677939   0.097091  -6.983  2.9e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(1.0222) family taken to be 1)
##
##     Null deviance: 424.53  on 313  degrees of freedom
## Residual deviance: 358.35  on 311  degrees of freedom
## AIC: 1741.5
## ## Number of Fisher Scoring iterations: 1
##               Theta:  1.022
##           Std. Err.:  0.105
##  2 x log-likelihood:  -1733.516

# checking other values!
# summary(mod.nb, dispersion = 1)
```

Question 4

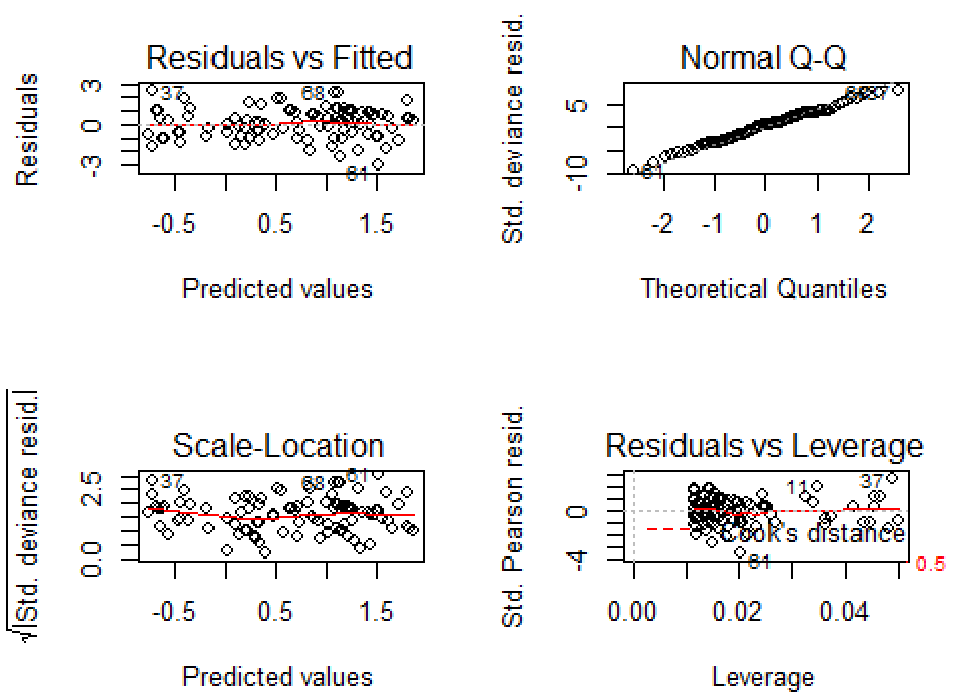
Residual checking for GLMs is not always as straightforward as for linear models, and the problems are particularly acute in the case of binary responses. This question explores this issue.

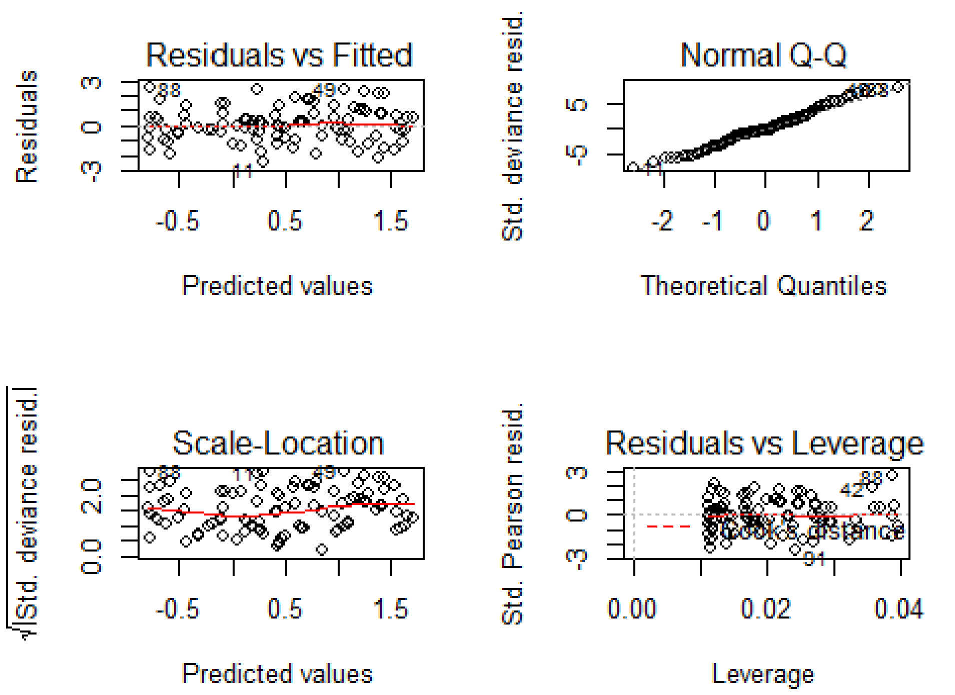
(a) The following code fits a GLM to data simulated from a simple binomial model and examines the default residual plots. Run the code several times to get a feel for the range of results that are possible even when the model is correct (as it is in this case).

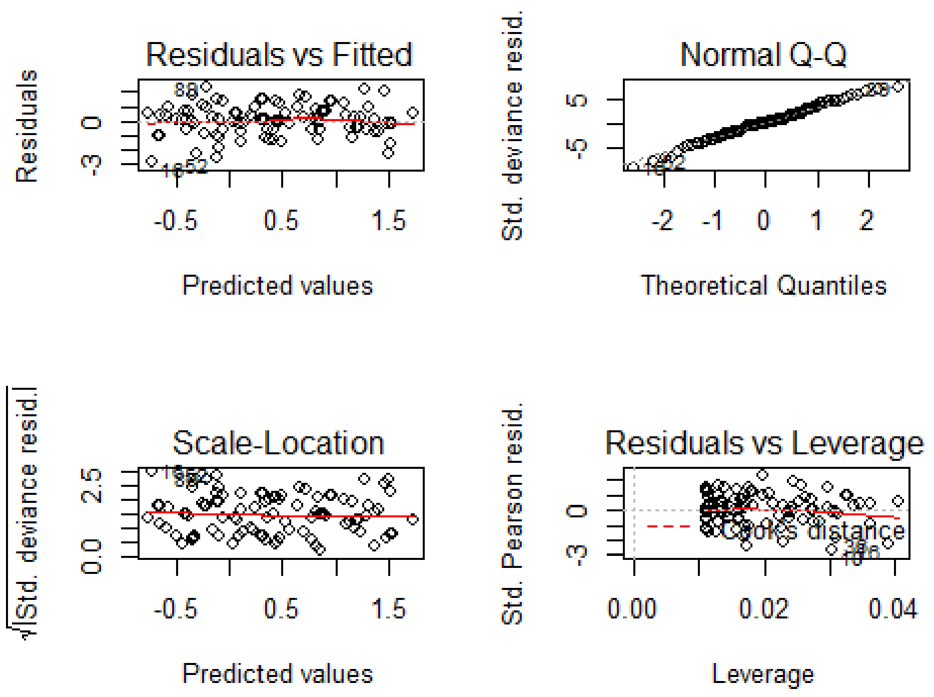
Examine the Residual Plot: The residual plots are not exhibiting clear pattern. To further comprehend the residual plots, an investigation of Pearson Residuals has been conducted. There is no obvious pattern, but there are some. It is worthy to consider that different runs will produce different results, eventually, different residual plots every time we run the code. It is because the input data is generated randomly every time we run the code.

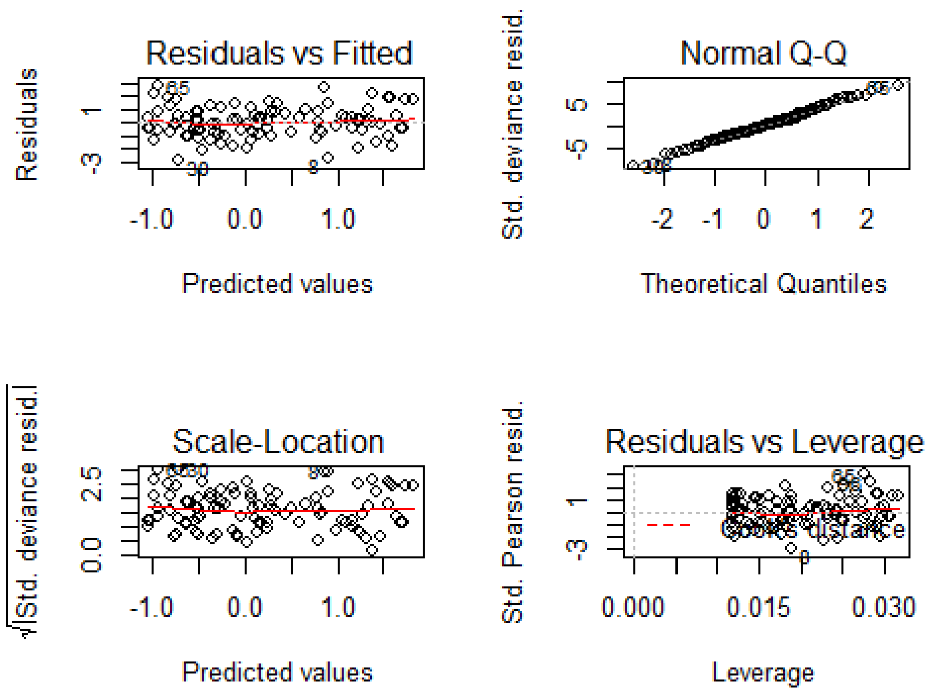
Since glm is fitted in this problem, the investigation of the normal Q-Q plot seems redundant.

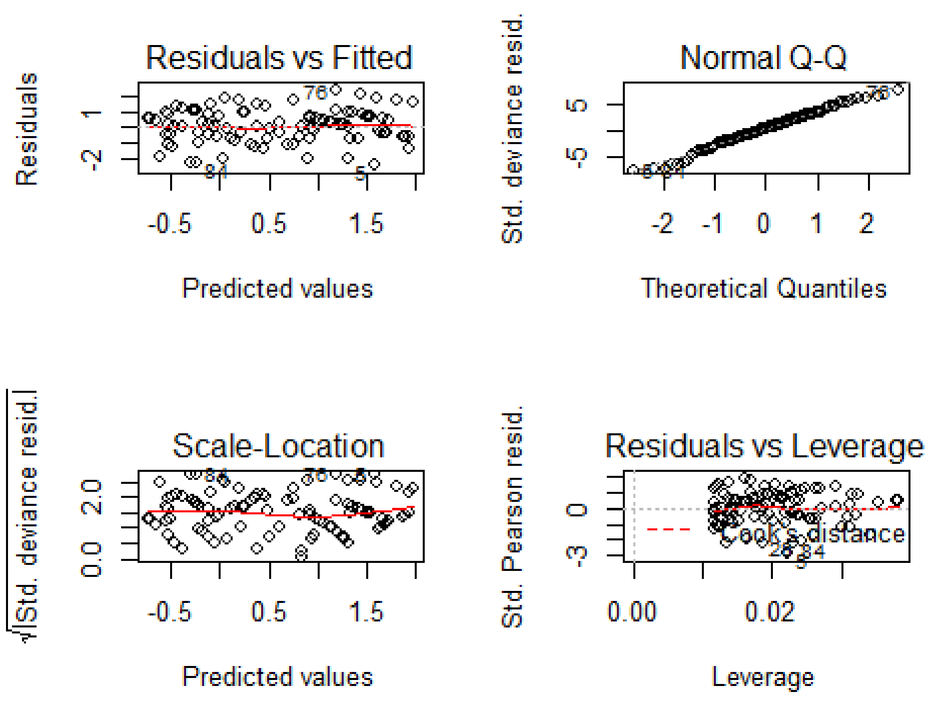
```
# 4(a)
for(i in 1:10) {
  n<-100;m<-10
  x<-runif(n)
  lp<-3*x-1
  mu<-binomial()$linkinv(lp)
  y<-rbinom(1:n,m,mu)
  par(mfrow=c(2,2))
  model.glm_4a <- glm(y/m~x,family=binomial,weights=rep(m,n))
  plot(model.glm_4a)
  summary(model.glm_4a)
}
```

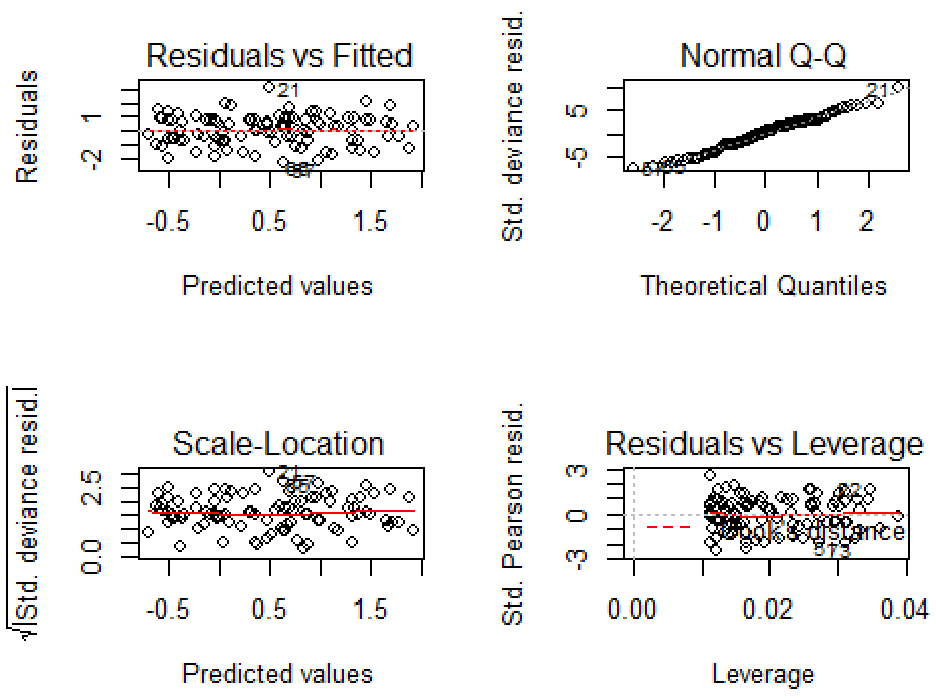

Residuals vs Fitted
Residuals
Predicted values
37
68
61
Normal Q-Q
Std. deviance resid.
Theoretical Quantiles
61
Scale-Location
√|Std. deviance resid.|
Predicted values
37
68
61
Residuals vs Leverage
Std. Pearson resid.
Leverage
Cook's distance
11
37
61
0.5

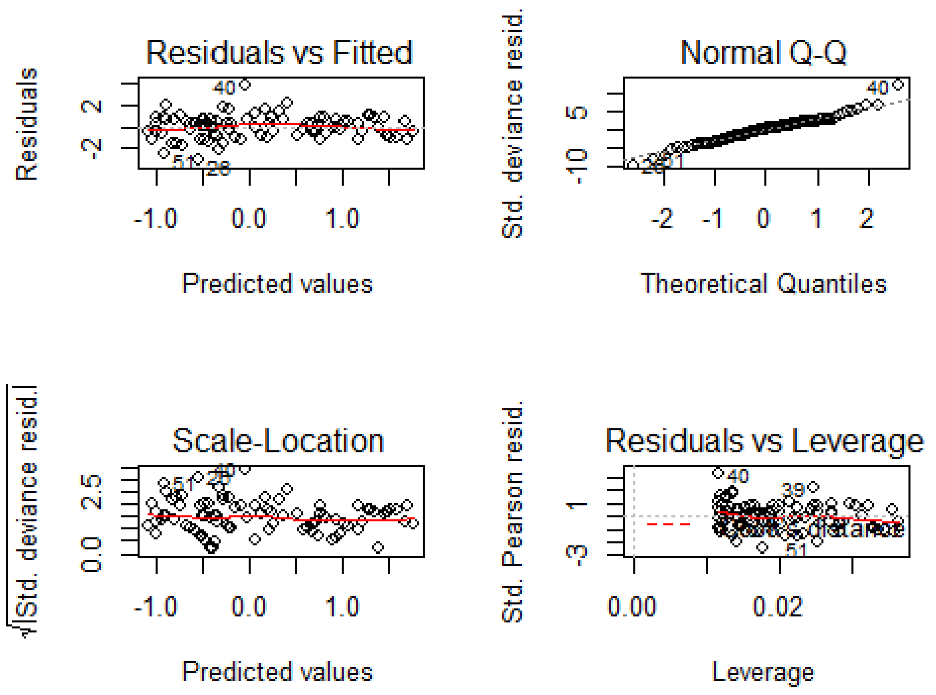

Residuals vs Fitted
Residuals
Predicted values
88
49
11
Normal Q-Q
Std. deviance resid.
Theoretical Quantiles
11
Scale-Location
√|Std. deviance resid.|
Predicted values
88
11
49
Residuals vs Leverage
Std. Pearson resid.
Leverage
Cook's distance
42
88
91

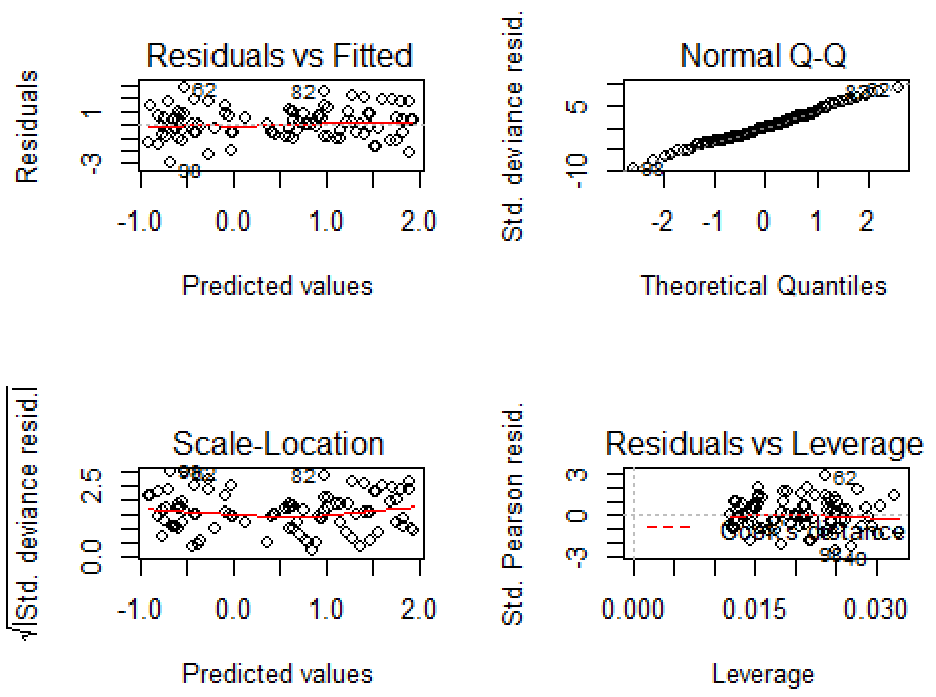

Residuals vs Fitted
Residuals
Predicted values
Normal Q-Q
Std. deviance resid.
Theoretical Quantiles
Scale-Location
√|Std. deviance resid.|
Predicted values
Residuals vs Leverage
Std. Pearson resid.
Cook's distance
Leverage


Residuals vs Fitted
Residuals
Predicted values
Normal Q-Q
Std. deviance resid.
Theoretical Quantiles
Scale-Location
√|Std. deviance resid.|
Predicted values
Residuals vs Leverage
Std. Pearson resid.
Cook's distance
Leverage


Residuals vs Fitted
Residuals
Predicted values
Normal Q-Q
Std. deviance resid.
Theoretical Quantiles
Scale-Location
√|Std. deviance resid.|
Predicted values
Residuals vs Leverage
Std. Pearson resid.
Cook's distance
Leverage

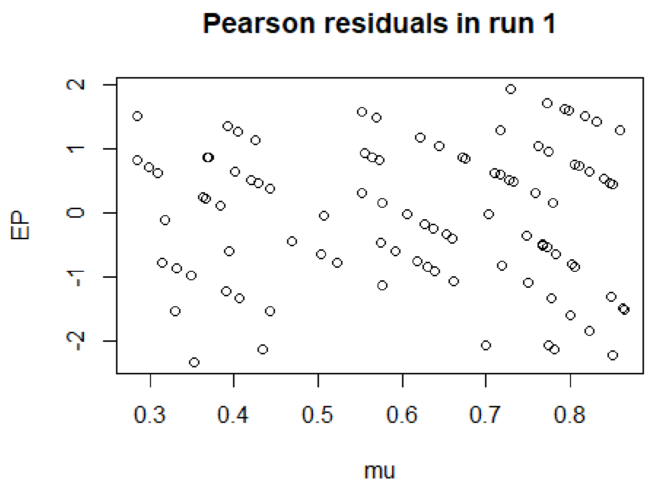

Residuals vs Fitted
Residuals
Predicted values
Normal Q-Q
Std. deviance resid.
Theoretical Quantiles
Scale-Location
√|Std. deviance resid.|
Predicted values
Residuals vs Leverage
Std. Pearson resid.
Cook's distance
Leverage

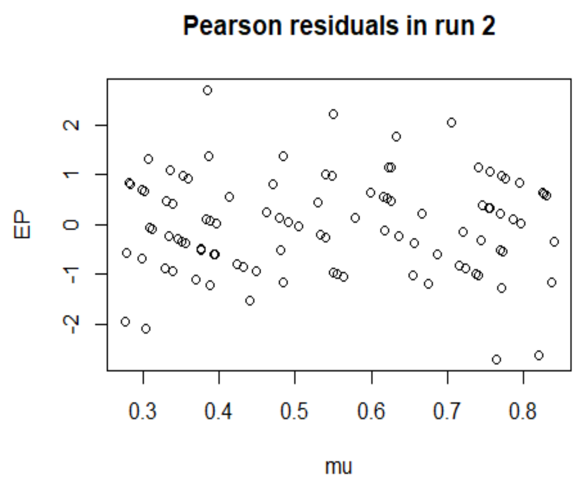


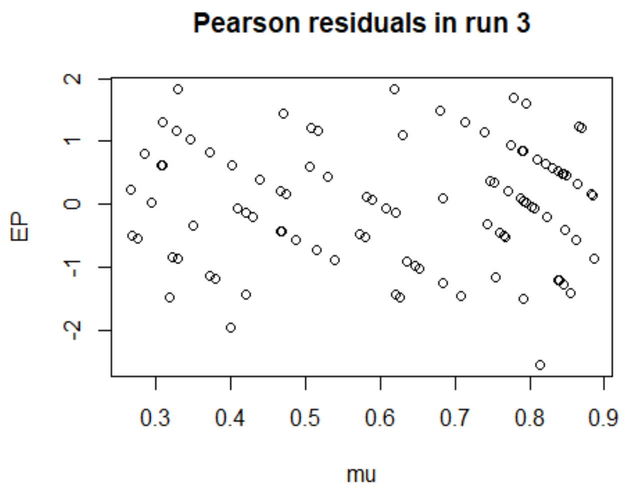


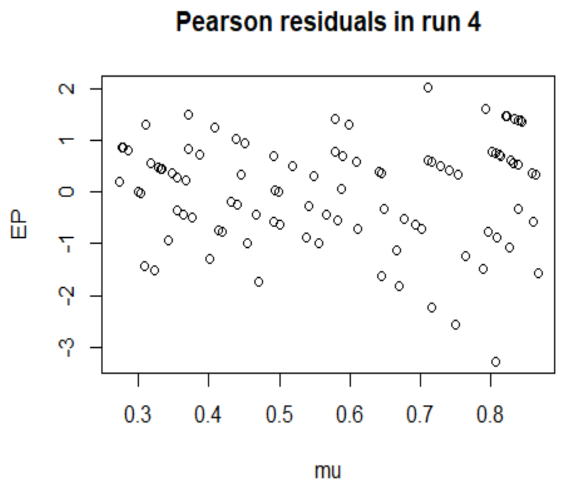
```r
# Additional analyses on Pearson Residual plot
for(i in 1:10) {
  n<-100;m<-10
  x<-runif(n)
  lp<-3*x-1
  mu<-binomial()$linkinv(lp)
  y<-rbinom(1:n,m,mu)
```

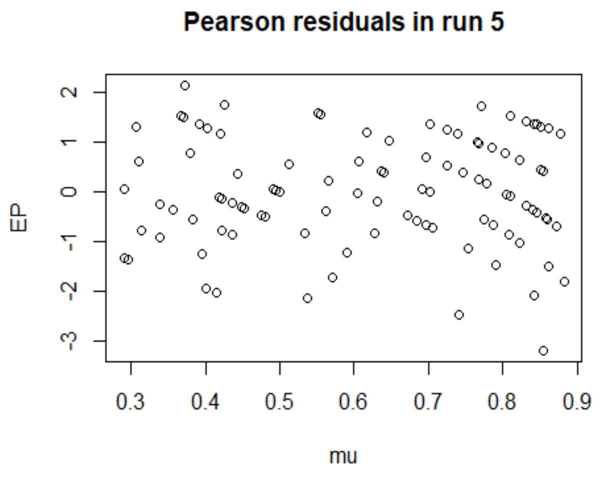
```
 model.glm_4a <- glm(y/m~x,family=binomial,weights=rep(m,n))
 EP=resid(model.glm_4a,type="pearson") # residual plots can be used in order to identify non-linearity.
 mu=predict(model.glm_4a,type="response")
 plot(x=mu,y=EP,main=paste("Pearson residuals in run", i))
}
```

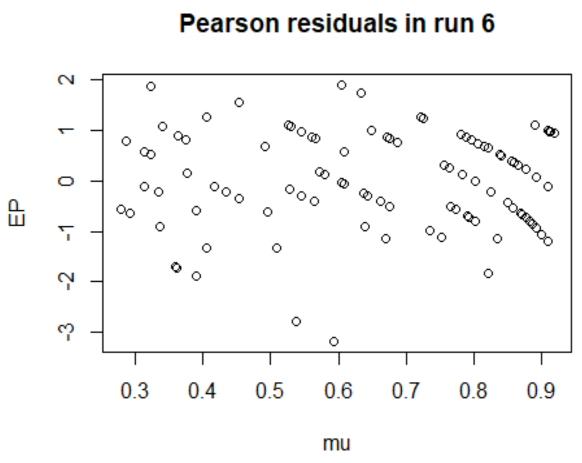


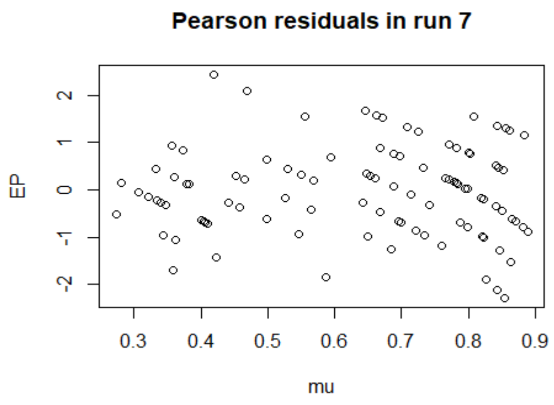


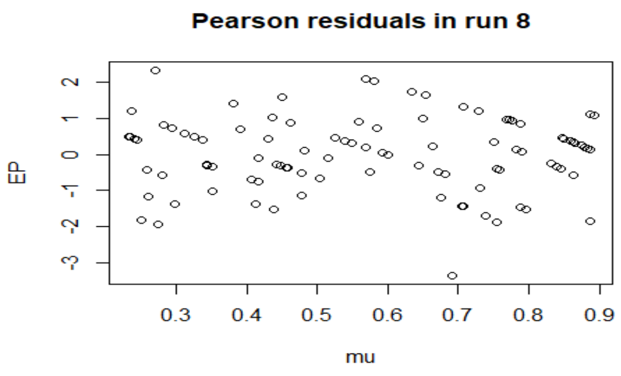


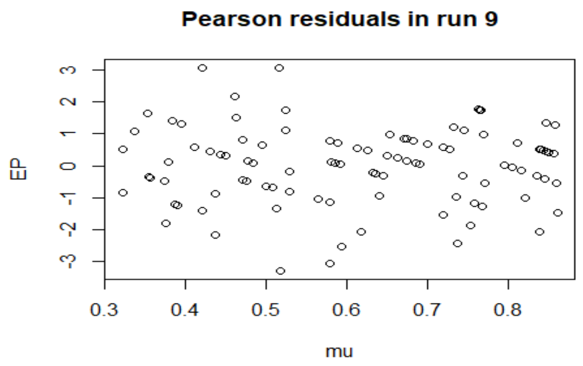


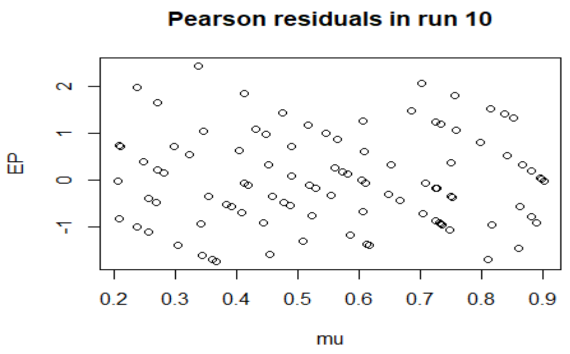








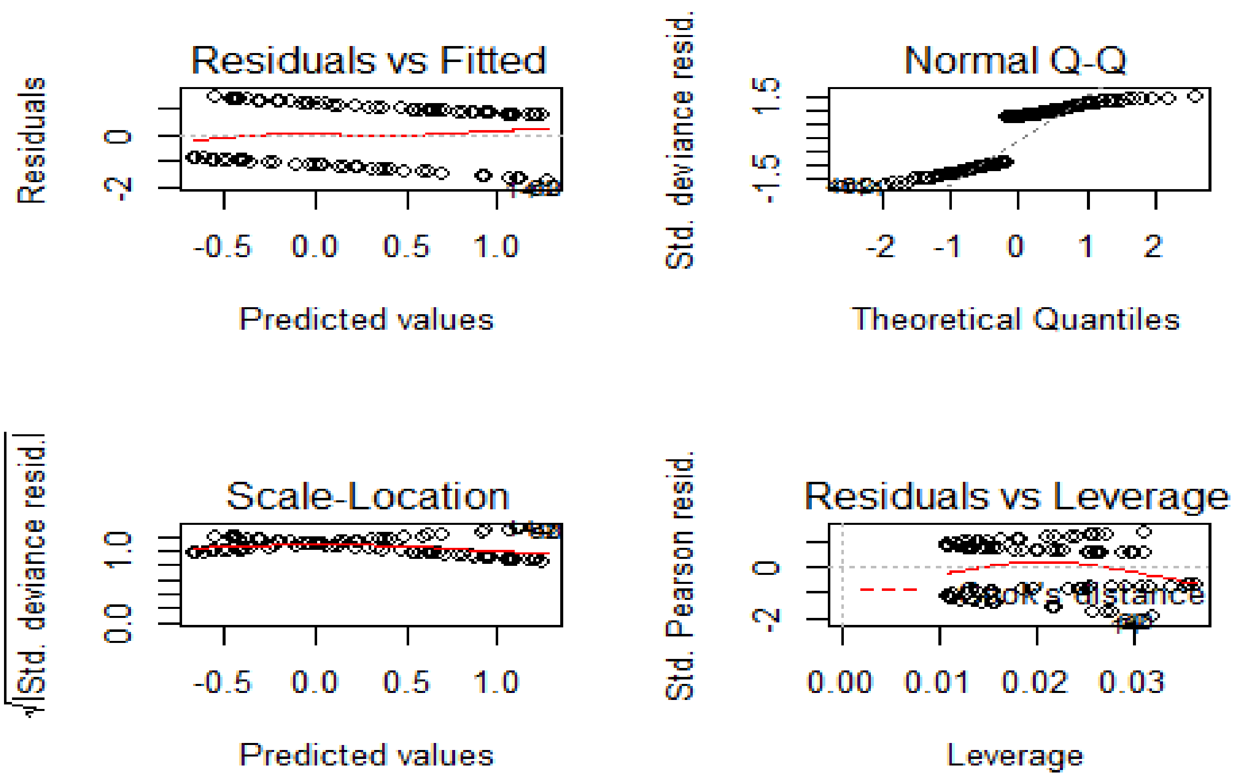




4(b) Explore how the plots change as m (the number of binomial trials) is reduced to 1.

Exploration: The value reduction of m has a drastic effect on the model outputs. There appears to be a trend in the mean of the residuals plotted against fitted values when $m = 1$, which may cause concern. As interpreted from the figure below, for the lesser number of trials, the model will not work appropriately, which can be considered reasonable. In other words, the model will not be able to learn from the data.

```
# 4(b) Part 1: m = 1
n<-100; m<-1
x<-runif(n)
lp<-3*x-1
mu<-binomial()$linkinv(lp)
y<-rbinom(1:n,m,mu)
par(mfrow=c(2,2))
plot(glm(y/m~x,family=binomial,weights=rep(m,n)))
```

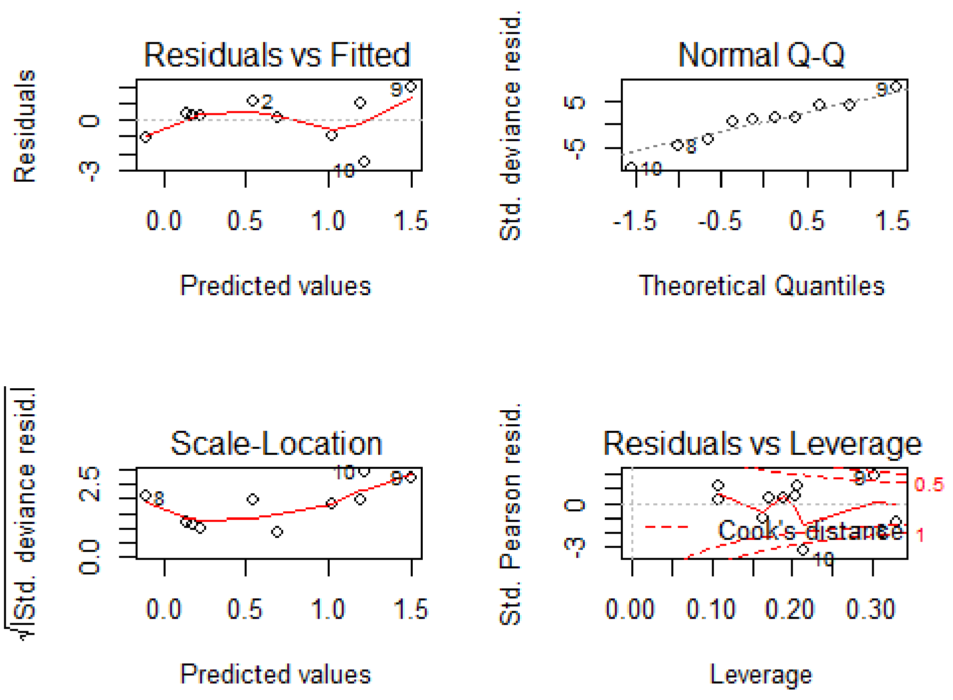


4(b) part2: Also examine the effect of sample size n. Describe your conclusions.

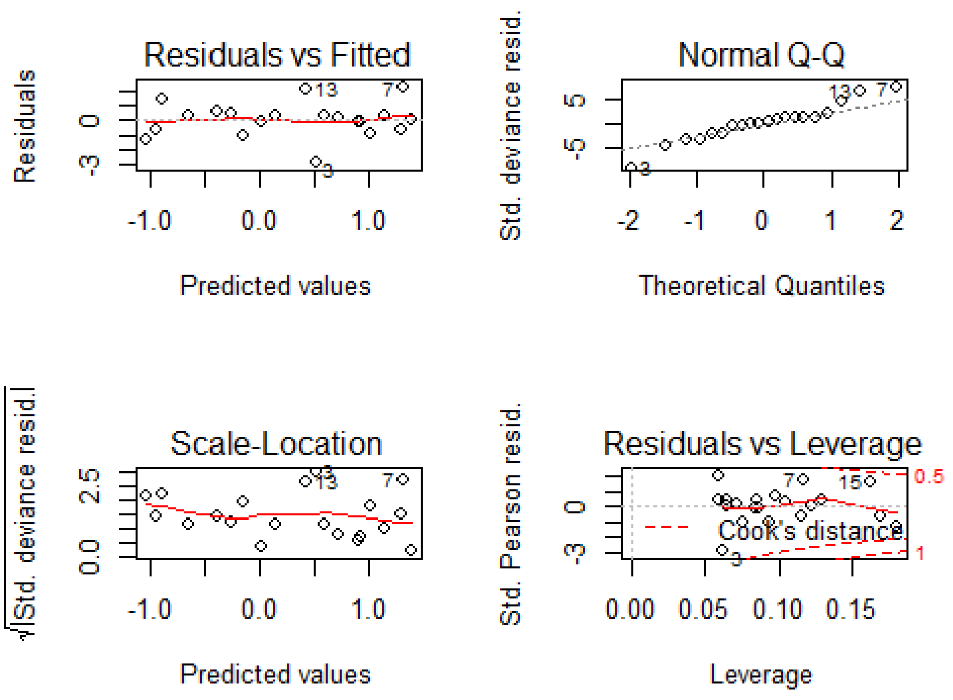
Description: There are still some patterns in the reduced vs. fitted plot, especially for a smaller sample size, which calls for a further investigation. If the sample size is increased, better models are obtained in a sense that residual plots exhibits less patterns. When n = 1 or 5, there are clear trend in the residual plots. With an increased number of sample size, the residual plots get better.

Conclusion: Rerunning the program generates different results every time. But, overall changes are not noteworthy until we change the parameters as performed within the scope of the question 4(b). Considering different sample size, when the sample size is within 30 to 80, the residual plots look more random compared to the other values of sample size n. The residual deviances and the AIC values are reasonable within this range. These might suggest to select a value of sample size in between 30 to 80.

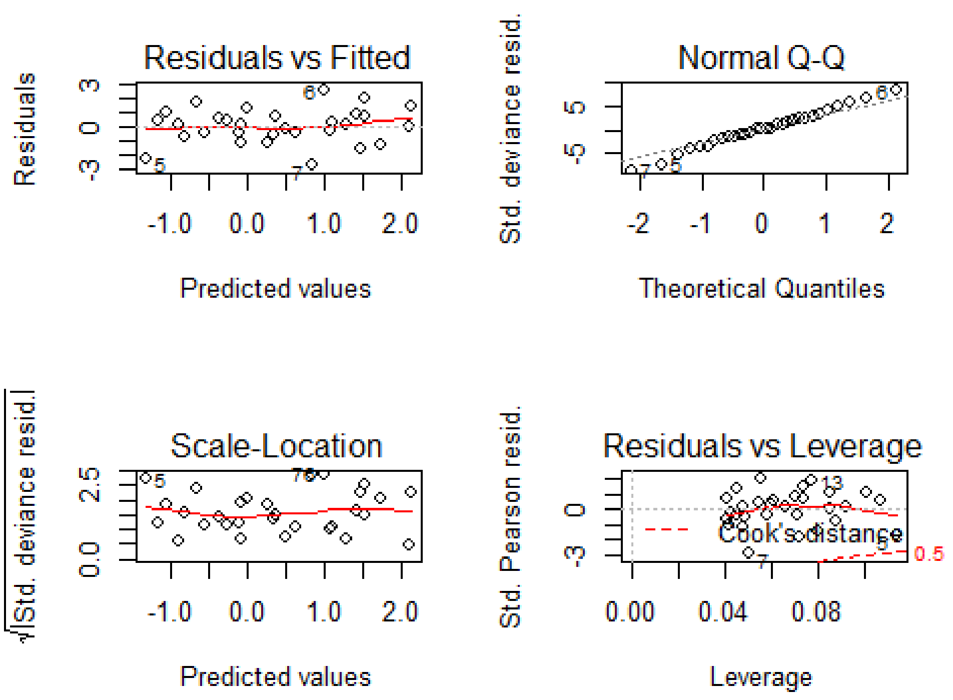
```r
# 4(b) part 2: The effect of sample size, n
m<-10
for(n in c(10,20,30, 40, 50, 60, 70, 80, 100, 200, 500)) {
  x<-runif(n) #,40,50,60,70,80,90,100
  lp<-3*x-1
  mu<-binomial()$linkinv(lp)
  # rbinom(n, size, prob)
  y<-rbinom(1:n,m,mu)
  par(mfrow=c(2,2))
  model.glm_4b = glm(y/m~x,family=binomial,weights=rep(m,n))
  plot(model.glm_4b)
  #residual_pr = resid(model.glm_4b,type="pearson")
  residual_deviance = as.numeric(unlist(model.glm_4b["deviance"]))
  aic = as.numeric(unlist(model.glm_4b["aic"]))
  print(c(residual_deviance, aic))
}
```



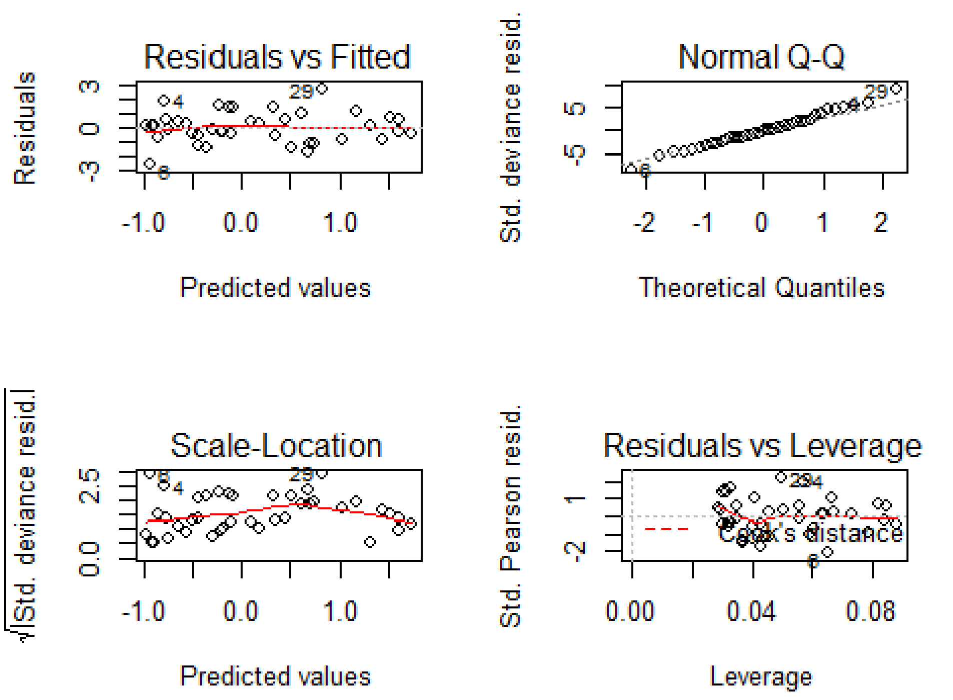
```
## [1] 15.43559 42.84150
```



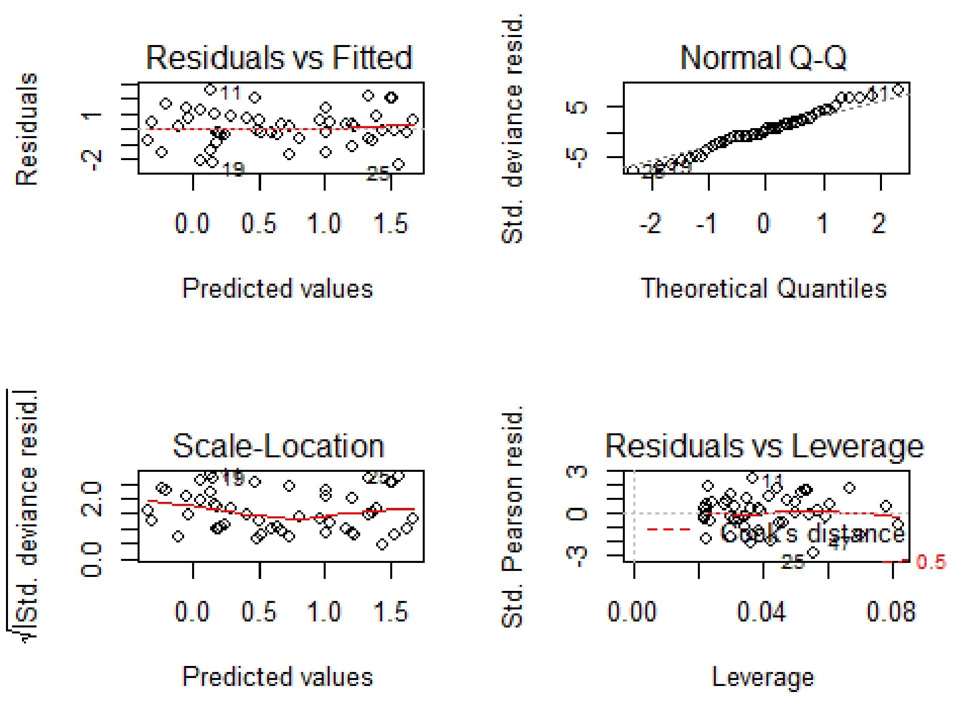
```
## [1] 24.08504 76.82513
```



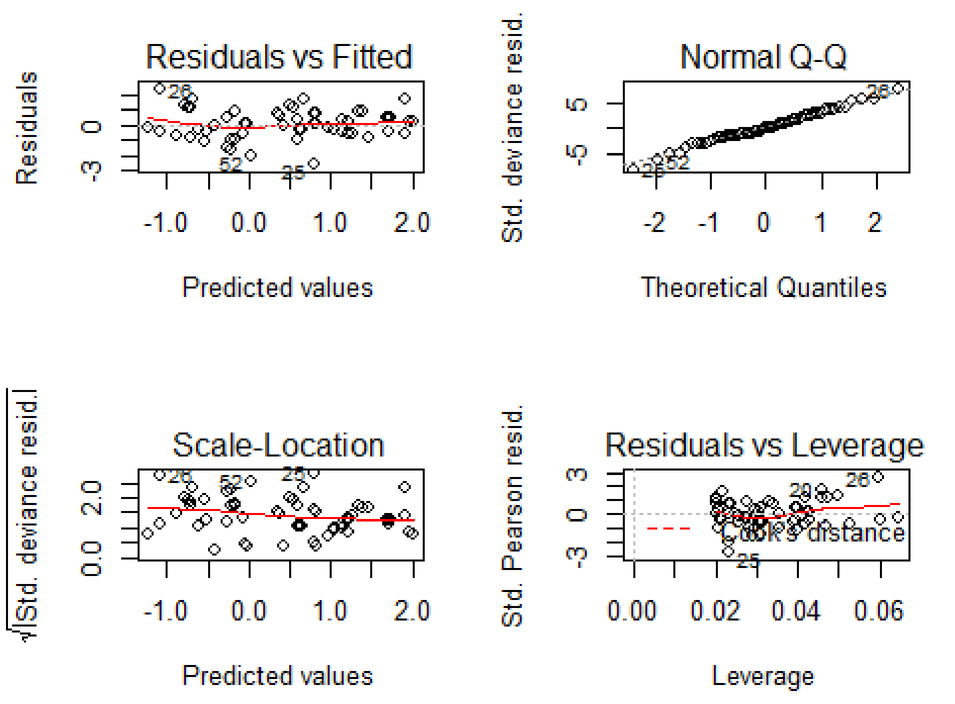
```
## [1]  39.84115 111.07398
```



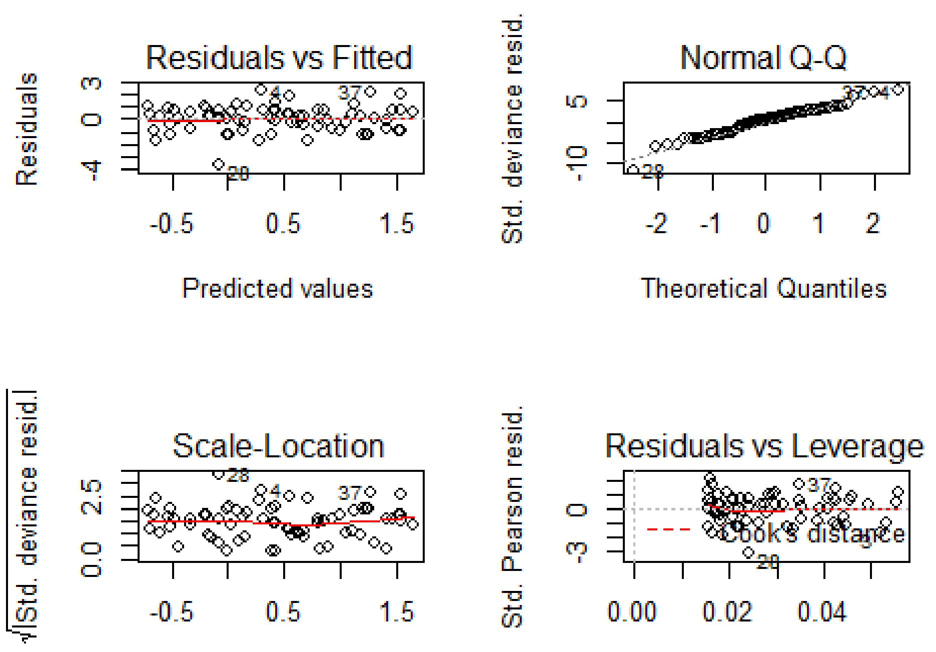
```
## [1]  45.7309 148.0140
```



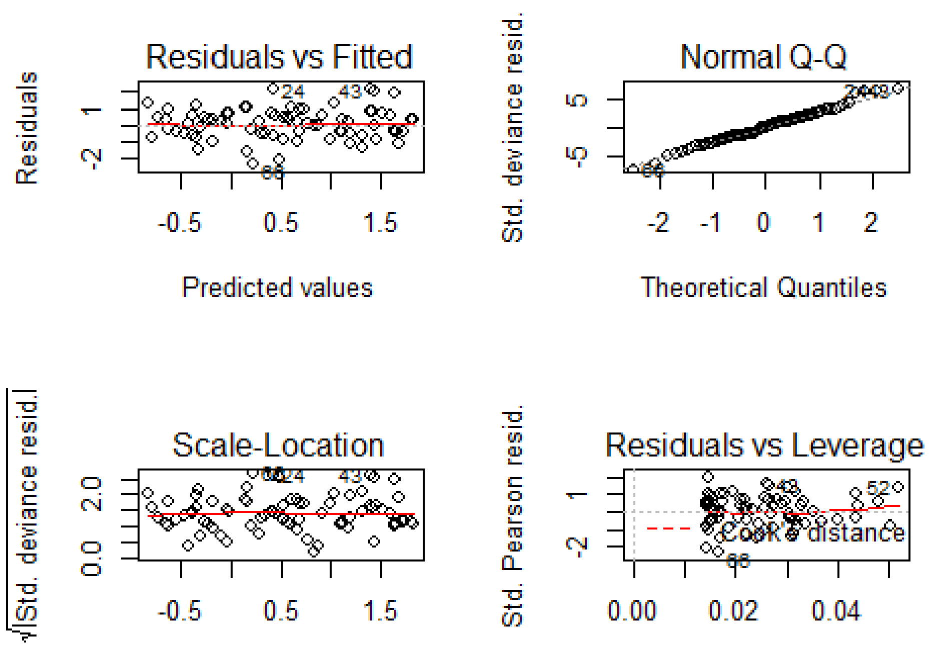
```
## [1]  65.70017 189.84366
```



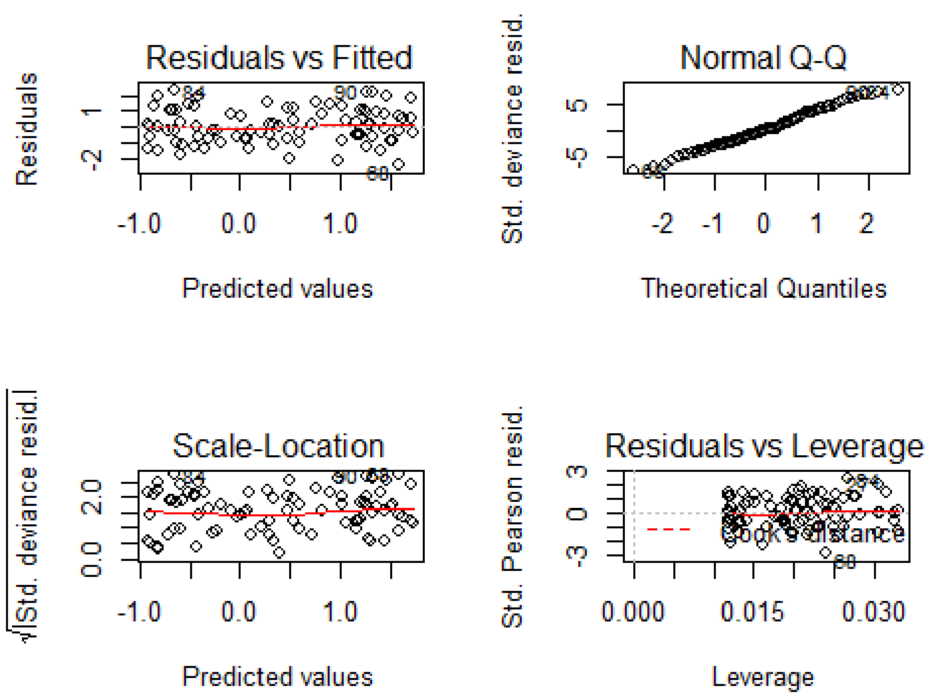
```
## [1]  51.6181 202.9626
```



```
## [1]  74.00232 251.80355
```



```
## [1]  63.06989 264.10139
```

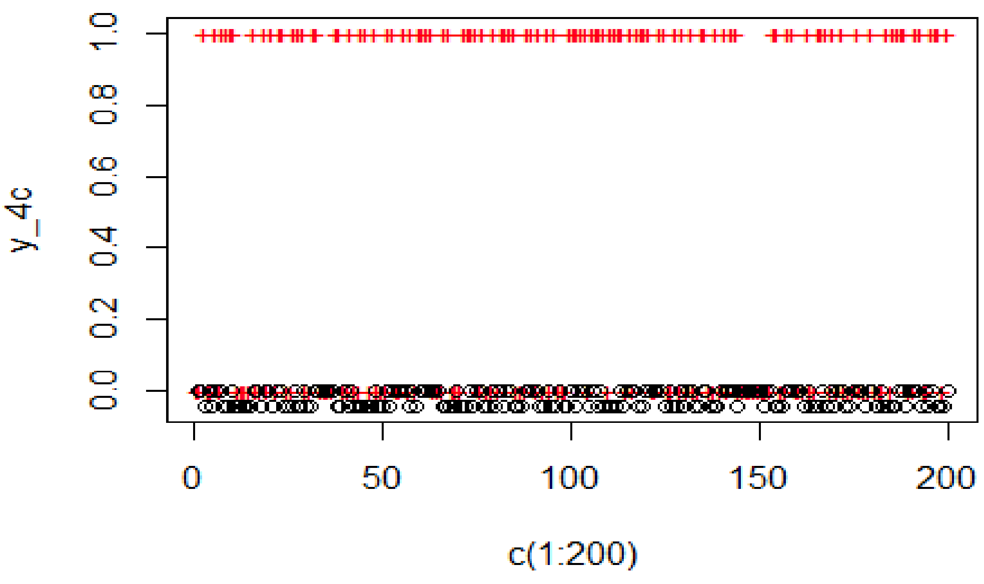


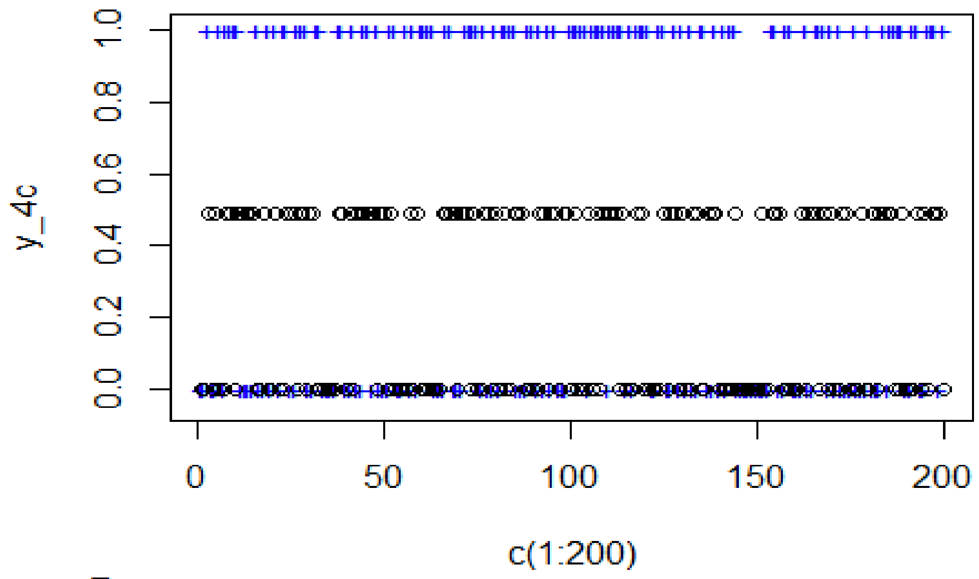
```
## [1] 109.9713 355.9535
```

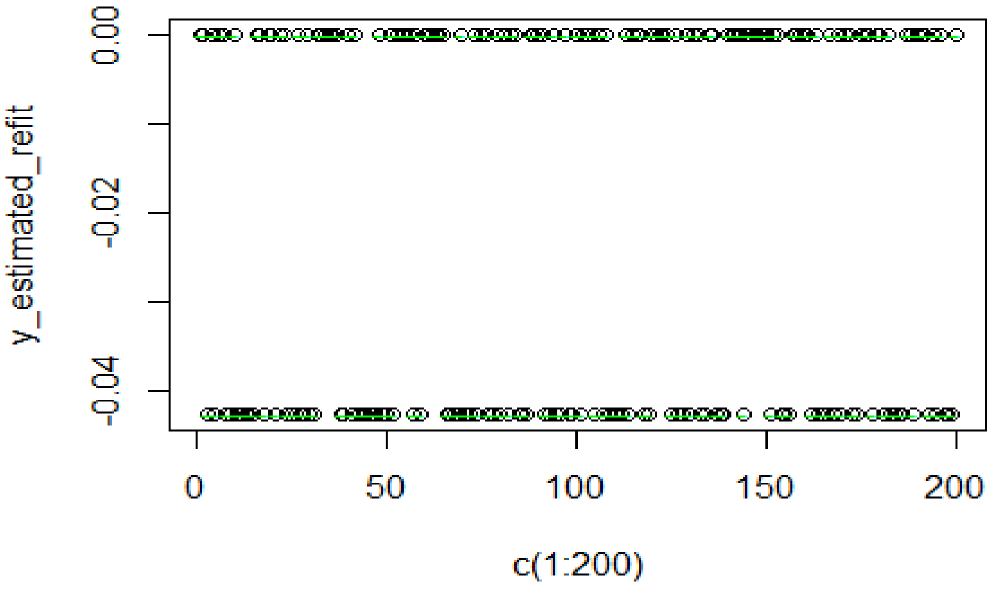
4(c) Write code to take a glm object fitted to binary data, simulate binary data given the model fitted values, refit the model to the resulting data, and extract residuals from the resulting fits.

```
# 4(c)
# Developing GLM object fitted to binary data
# Generate some binary data
x_4c <- rbinom(200, 1, 0.5)  # rbinom(n, size, prob)
# n: number of observations; size: number of trials
y_4c <- rbinom(200, 1, 0.5)
# plot(x_4c, y_4c,  col="blue", xlab="Explanatory var (Binomial)", ylab="Response (Binary: 0,1)")

# model fitting
model_glm_4c <- glm(y_4c ~ x_4c , family=binomial) #(link="logit")
summary_glm_4c = summary(model_glm_4c)

# simulate binary data given the model fitted values,
# find the estimated y
y_estimated = predict.glm(model_glm_4c)
plot(c(1:200), y_4c, pch="+", col="red", ylim=(c(min(y_estimated, 0),max(y_estimated, 1))))
points(c(1:200), y_estimated)

model_fit_4c_glmFit <- glm.fit(x_4c, y_4c)
y_estimated_glmFit = predict.glm(model_fit_4c_glmFit)
plot(c(1:200), y_4c, pch="+", col="blue", ylim=(c(min(y_estimated, 0),max(y_estimated, 1))))
points(c(1:200), y_estimated_glmFit)
```

```r
# refit the model to the resulting data, and
model_glm_4c_refit <- glm(y_estimated ~ x_4c ) # family=binomial
y_estimated_refit = predict.glm(model_glm_4c_refit)
plot(c(1:200), y_estimated_refit)
points(c(1:200), y_estimated, pch="-", col="green") # , legend="y_true"
```


y_4c
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
c(1:200)

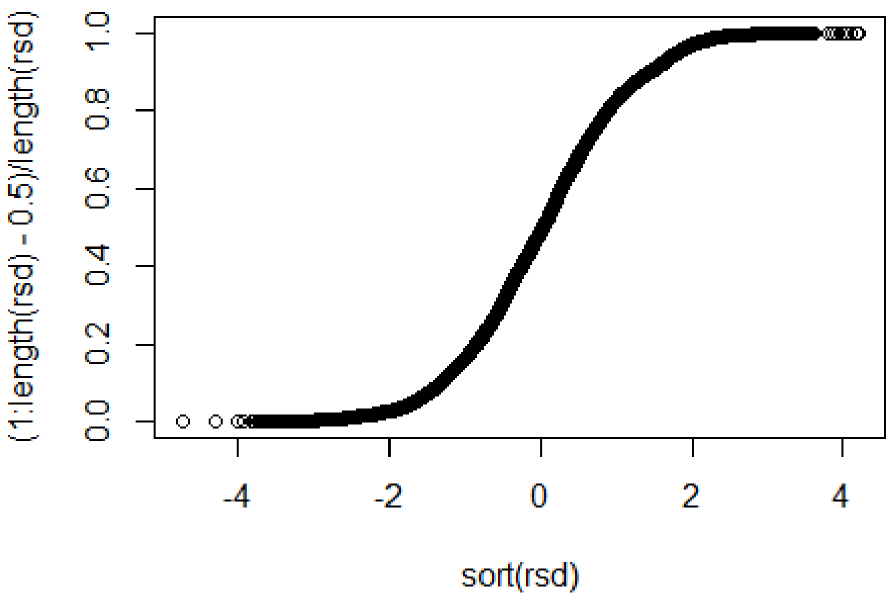

y_4c
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
c(1:200)


y_estimated_refit
0.00
-0.02
-0.04
0
50
100
150
200
c(1:200)

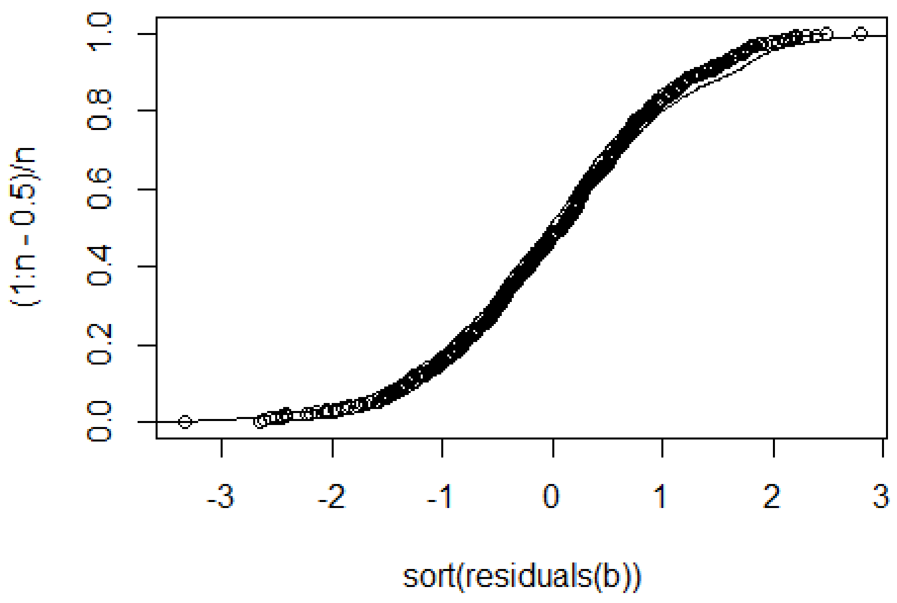
```r
# model_refit_glm_4c <- glm(y_estimated ~ x_4c , family=binomial)

# extract residuals from the resulting fits.
rsd = model_fit_4c_glmFit[["residuals"]]

# 4(c)
b <- glm(y/m~x,family=binomial,weights=rep(m,n))
reps <- 200;
mu <- fitted(b)
rsd <- matrix(0,reps,n) # Empty array for simulated resids
runs <- rep(0,reps) # array for simulated run counts
for (i in 1:reps) { # simulation loop
  ys <- rbinom(1:n,m,mu) # simulate from fitted model
  ## refit model to simulated data
  br <- glm(ys/m~x,family=binomial,weights=rep(m,n))
  rs <- residuals(br) # simulated resids (meet assumptions)
  rsd[i,] <- sort(rs) # store sorted residuals
  fv.sort <- sort(fitted(br),index.return=TRUE)
  rs <- rs[fv.sort$ix] # order resids by sorted fit values
  rs <- rs > 0 # check runs of +ve, -ve resids
  runs[i] <- sum(rs[1:(n-1)]!=rs[2:n])
}
```

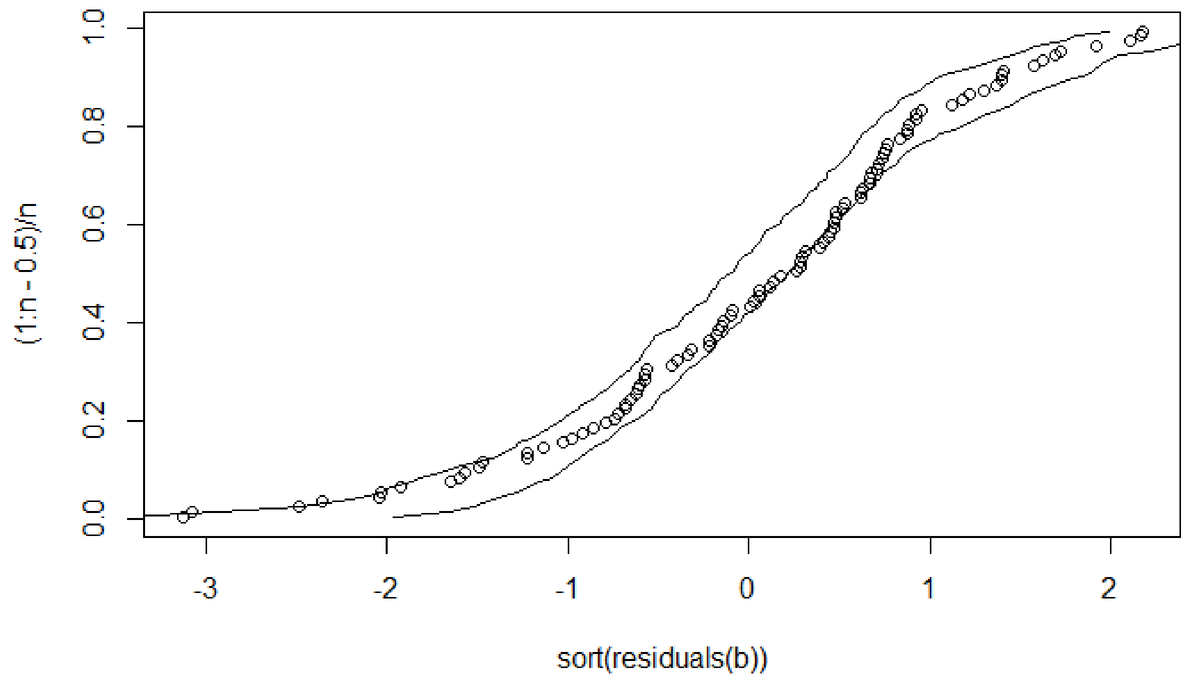
4(d)

```r
# 4(d)

plot(sort(rsd), (1:length(rsd)-.5)/length(rsd))
```



```
# plot original ordered residuals, and simulation envelope
for (i in 1:n) rsd[,i] <- sort(rsd[,i])
par(mfrow=c(1,1))
plot(sort(residuals(b)),(1:n-.5)/n) # original
## plot 95% envelope ....
lines(rsd[5,],(1:n-.5)/n);
lines(rsd[reps-5,],(1:n-.5)/n)
```





4(e)

There are 51 cases where individual runs are greater than the observed runs of 248.

```
# 4(e)
# compare original runs to distribution under independence
rs <- residuals(b)
fv.sort <- sort(fitted(b),index.return=TRUE)
rs <- rs[fv.sort$ix]
rs <- rs > 0
obs.runs <- sum(rs[1:(n-1)]!=rs[2:n])
sum(runs>obs.runs)

## [1] 51
```

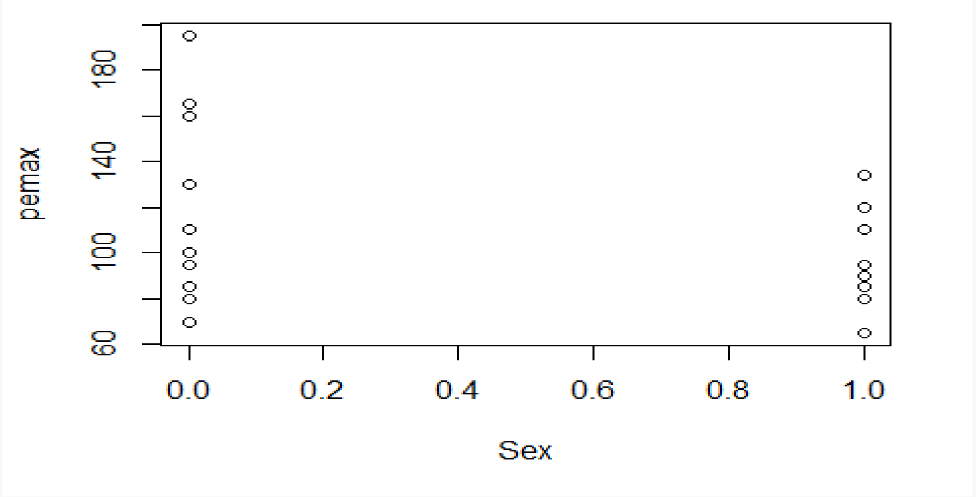
Question 5

We are interested in a study concerning lung function in patients with cycstic fibrosis (Altman (1991, p.338)). Data are in the ISwR package and can be loaded into the workspace with data(cystfibr).

5(a) Fit a model relating Maximum expiratory pressure (pemax) to the explanatory variables contained in the dataset.

To start building models, we will develop linear model, then, generalized linear model, and after that reduced generalized linear model suggested by the step function.

```
# 5(a)
# Load the dataset
data5 = read.csv("../Data/cystfibr.csv")
head(data5)

##   age sex height weight bmp fev1  rv frc tlc pemax
## 1   7   0    109   13.1  68   32 258 183 137    95
## 2   7   1    112   12.9  65   19 449 245 134    85
## 3   8   0    124   14.1  64   22 441 268 147   100

plot(data5$sex, data5$pemax, xlab="Sex", ylab="pemax")
```



```
# fit a linear model
model.lm_5a <- lm( pemax ~ ., data=data5) # all explanatory variables are considered by . notation
```

```
# fit a generalized linear model
model.glm_5a <- glm( pemax ~ ., data=data5) #
```

5(b) Interpret results for the sex variable.

According to the summary table, the variable SEX has the highest p-value. A p-value this high indicates that there is insufficient evidence in sample to conclude that a non-zero correlation exists between SEX and the response variable considered. Therefore, this SEX variable can be dropped & refit the model. The STEP function will later justify this decision.

```
# 5(b) Generating the Summary table to check the SEX variable
summary(model.glm_5a)

##
## Call:
## glm(formula = pemax ~ ., data = data5)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -37.338  -11.532    1.081   13.386   33.405
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 176.0582   225.8912   0.779    0.448
## age          -2.5420     4.8017  -0.529    0.604
## sex          -3.7368    15.4598  -0.242    0.812
## height       -0.4463     0.9034  -0.494    0.628
## weight        2.9928     2.0080   1.490    0.157
## bmp          -1.7449     1.1552  -1.510    0.152
## fev1          1.0807     1.0809   1.000    0.333
## rv            0.1970     0.1962   1.004    0.331
## frc          -0.3084     0.4924  -0.626    0.540
## tlc           0.1886     0.4997   0.377    0.711
##
## (Dispersion parameter for gaussian family taken to be 648.75)
##
##     Null deviance: 26832.6  on 24  degrees of freedom
## Residual deviance:  9731.2  on 15  degrees of freedom
## AIC: 242.05
##
## Number of Fisher Scoring iterations: 2
```

5(c) Try using the step function and interpret results.

A backward STEP function is tried as follow. The minimum AIC (233.6) has been obtained by pemax ~ weight + bmp + fev1 + rv . As a consequence, a generalized linear model is developed using pemax as response variable and weight, bmp, fev1, and rv as explanatory variable, rather than exploiting all explanatory variables.

```
# 5(c)
# Step function
step(model.glm_5a)
```

```
## Start:  AIC=242.05
## pemax ~ age + sex + height + weight + bmp + fev1 + rv + frc +
##     tlc
##
##          Df Deviance    AIC
## - sex     1   9769.2 240.15
## - tlc     1   9823.7 240.29
## - height  1   9889.6 240.46
## - age     1   9913.1 240.51
## - frc     1   9985.8 240.70
## - fev1    1  10379.7 241.66
## - rv      1  10385.0 241.68
## <none>        9731.2 242.05
## - weight  1  11172.5 243.50
## - bmp     1  11211.4 243.59
##
## Step:  AIC=240.15
## pemax ~ age + height + weight + bmp + fev1 + rv + frc + tlc
##
##          Df Deviance    AIC
## - tlc     1   9885.1 238.44
## - height  1   9900.4 238.48
## - age     1   9914.7 238.52
## - frc     1   9990.7 238.71
## - rv      1  10405.3 239.73
## <none>        9769.2 240.15
## - weight  1  11215.4 241.60
## - bmp     1  11243.9 241.66
## - fev1    1  11539.6 242.31
##
## Step:  AIC=238.44
## pemax ~ age + height + weight + bmp + fev1 + rv + frc
##
##          Df Deviance    AIC
## - frc     1  10018.3 236.78
## - height  1  10100.9 236.98
## - age     1  10137.3 237.07
## - rv      1  10428.6 237.78
## <none>        9885.1 238.44
## - fev1    1  11612.5 240.47
## - weight  1  12017.6 241.33
## - bmp     1  12239.4 241.78
##
## Step:  AIC=236.78
## pemax ~ age + height + weight + bmp + fev1 + rv
##
##          Df Deviance    AIC
## - age     1    10164 235.14
## - height  1    10176 235.17
## - rv      1    10586 236.16
## <none>         10018 236.78
## - weight  1    12046 239.39
## - bmp     1    12342 240.00
## - fev1    1    12870 241.04
```

```
##
## Step:  AIC=235.14
## pemax ~ height + weight + bmp + fev1 + rv
##
##          Df Deviance    AIC
## - height  1    10355 233.60
## - rv      1    10993 235.10
## <none>         10164 235.14
## - weight  1    12767 238.84
## - bmp     1    12907 239.11
## - fev1    1    13374 240.00
##
## Step:  AIC=233.6
## pemax ~ weight + bmp + fev1 + rv
##
##          Df Deviance    AIC
## <none>         10355 233.60
## - rv      1    11538 234.31
## - bmp     1    13427 238.10
## - fev1    1    14072 239.27
## - weight  1    21285 249.62

##
## Call:  glm(formula = pemax ~ weight + bmp + fev1 + rv, data = data5)
##
## Coefficients:
## (Intercept)       weight          bmp         fev1           rv
##     63.9467       1.7489      -1.3772       1.5477       0.1257
##
## Degrees of Freedom: 24 Total (i.e. Null);  20 Residual
## Null Deviance:       26830
## Residual Deviance: 10350     AIC: 233.6
```

5(d) Perform a complete examination of diagnostics.

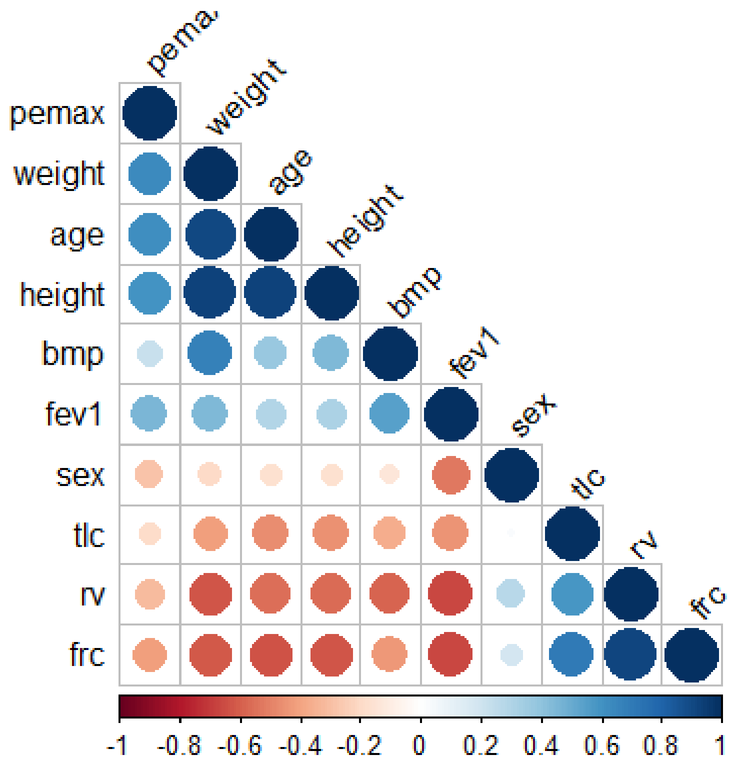
A comprehensive examinations of diagnostics are conducted below to check the validity of the models:

Comparing two models (model.glm_5a using all explanatory var vs model.glm_5c using explanatory var suggested by STEP). We may drop the SEX variables based on the p-value criterion during the model evaluation phase. This is also justified when it is observed that final combination recommended by STEP function exclude SEX variable as well. Considering this particular dataset, the 2nd model would work better, as observed by conducting ANOVA analysis and comparing residual degree of freedom and other parameters.

```
# model.glm_5a = glm(formula = pemax ~ ., data = data5)
model.full = model.glm_5a
model.reduced = glm(formula = pemax ~ weight + bmp + fev1 + rv, data = data5) # suggested by step function
model.glm_5c = model.reduced
# Anova
anova_test = anova(model.glm_5c, model.glm_5a)
names(anova_test)

## [1] "Resid. Df"  "Resid. Dev" "Df"         "Deviance"
```

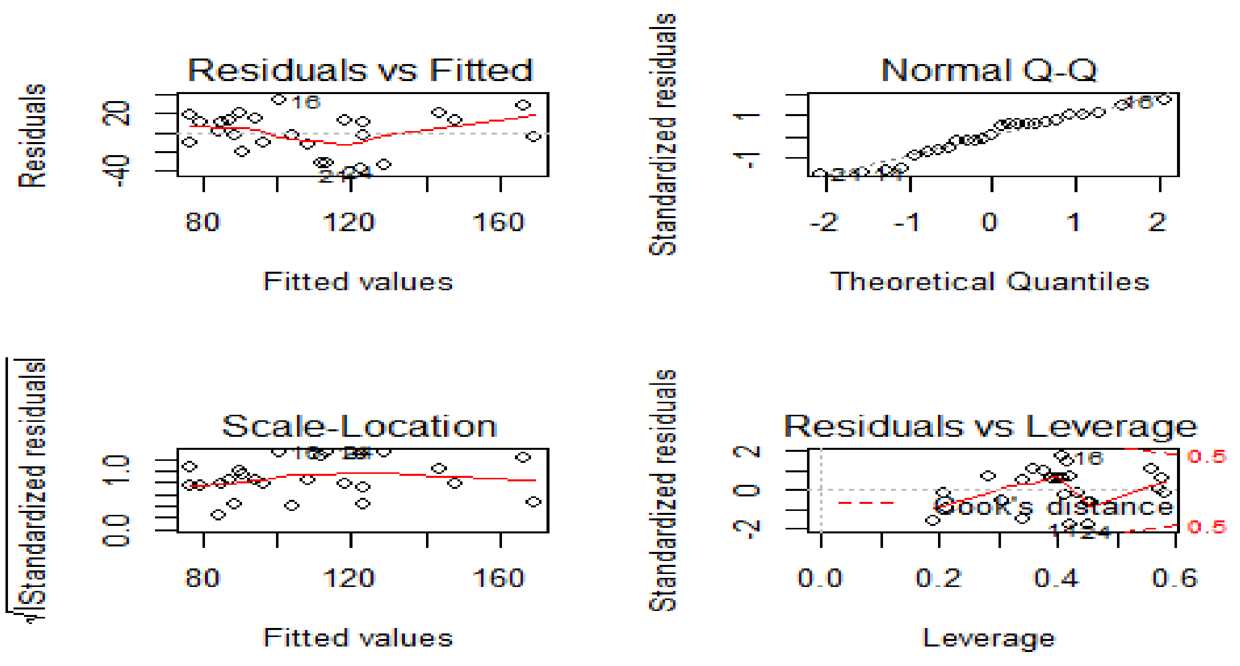
5d_1: Relationship between response and explanatory variables: It is obvious, from the correlation matrix below, that pemax has strong correlation with weight, age, height, fev1 and frc explanatory variables.

```
library(corrplot)

## Warning: package 'corrplot' was built under R version 3.6.2

## corrplot 0.84 loaded

# df_data5 = data.frame(data5)
correlation_mat = cor(data5)
corrplot(correlation_mat, type = "lower", order = "hclust",
         tl.col = "black", tl.srt = 45)
```

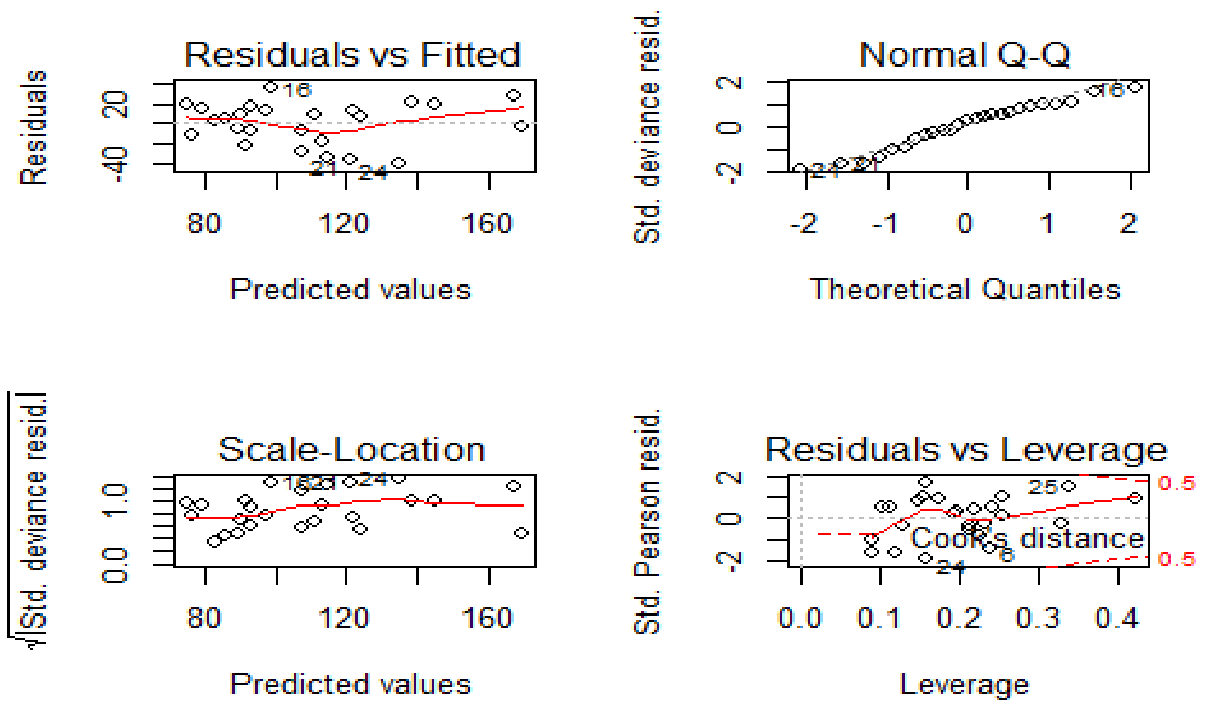


5d_2: Assessing model assumptions: considering the Linear Model developed in question 5(a) A few assumptions were taken into consideration when fitting a linear regression model e.g. the residuals are normally distributed. However, the Normal Q-Q plot obtained for the linear model of question 5(a) does not strongly suggest normality. Indeed, when we consider generalized linear model, either full model or reduced model, Normal Q-Q plot is not relevant anymore.

```
# Model checking plots for lm on cystfibr considering all variables
par(mfrow=c(2,2))
plot(model.lm_5a) #Model checking plots for the linear model fitted to the cystfibr data"
```



```
# Model checking plots for glm on cystfibr after step function
par(mfrow=c(2,2))
plot(model.glm_5c) #Model checking plots for the reduced linear model fitted to the cystfibr data"
```



5d_3: Adequacy of existing explanatory variables considering reduced model (5c): Out of four explanatory variables, three still have a substantially lower p-value (0.000175, 0.02, 0.01 for weight, bmp and fev1 respectively), indicating their importance in the model.

```
summary(model.glm_5c)

##
## Call:
## glm(formula = pemax ~ weight + bmp + fev1 + rv, data = data5)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -39.77  -11.74    4.33   15.66   35.07
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 63.94669   53.27673   1.200 0.244057
## weight       1.74891    0.38063   4.595 0.000175 ***
## bmp         -1.37724    0.56534  -2.436 0.024322 *
## fev1         1.54770    0.57761   2.679 0.014410 *
## rv           0.12572    0.08315   1.512 0.146178
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 517.73)
##
##     Null deviance: 26833  on 24  degrees of freedom
## Residual deviance: 10355  on 20  degrees of freedom
## AIC: 233.6
##
## Number of Fisher Scoring iterations: 2
```

5(e) What can you reasonably conclude from your analysis?

A linear model would not be a good fit for the $cystfibr$ data. Moreover, when explanatory variables have little correlation with the response variable, dropping them all together can be advisable; as a consequence, a better performance from the model would be attainable. In addition, the model run will be computationally efficient, mainly when the dataset contains a lot of variables.

To illustrate, the full generalized model developed in question 5(a) would be compared with the reduced model developed in question 5(c). The AIC values of question 5(a) and 5(c) are 242.05 and 233.6 respectively, which recommend to select the reduced model 5(c) over the full model 5(a) as well. The p-values of the explanatory variables in the reduced model are at a satisfactory level. Ultimately, some of the explanatory variables are not important.