Take Home Final Exam

Bradley Assaly-Nesrallah

Question 1

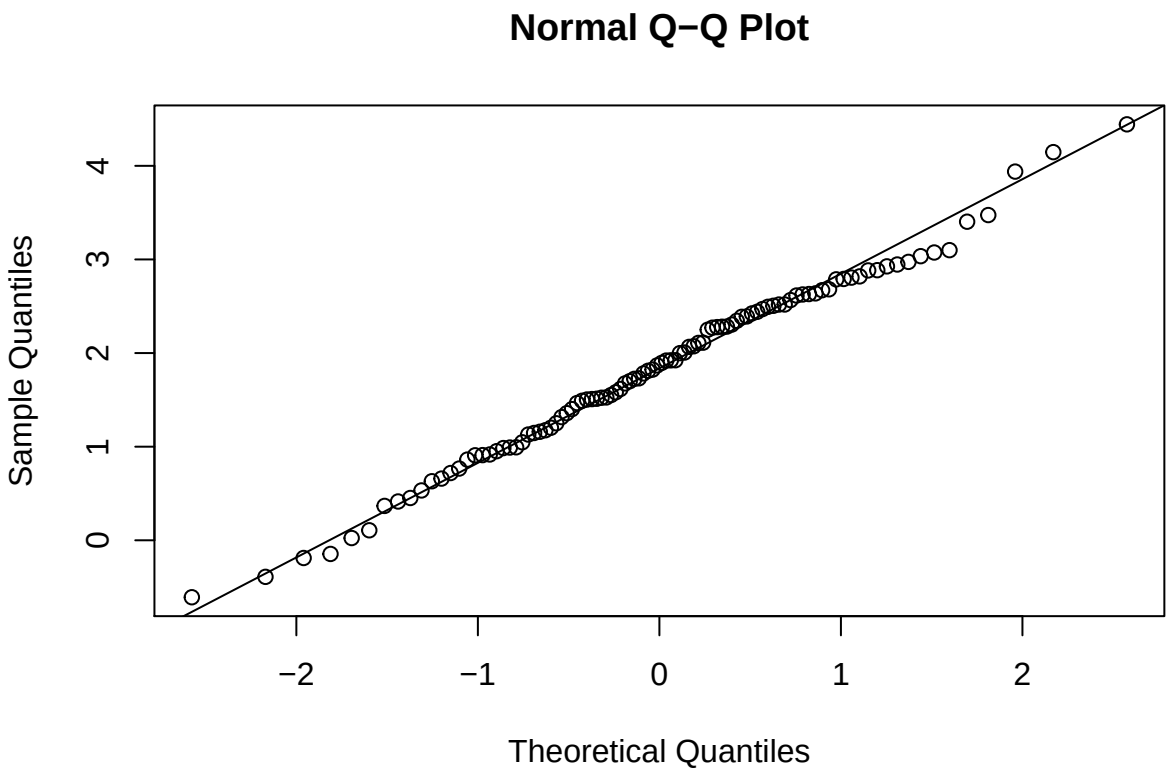
a) We implement the Jarque_Bera statistic in R as follows

```
#we define the function Jarque Bera Statistic with input param x
JB<-function(x){
  #compute length
  n <- length(x)
  #compute means used in the the statistic
  mean1 <- sum(x)/n
  mean2 <- sum((x-mean1)^2)/n
  mean3 <- sum((x-mean1)^3)/n
  mean4 <- sum((x-mean1)^4)/n
  #compute the standardized skewness and kurtosis
  b1 <- (mean3/mean2^(3/2))^2
  b2 <- (mean4/mean2^2)
  #return the Jarque Bera Statistic
  STATISTIC <- n*b1/6+n*(b2-3)^2/24

}
#test the function with rnorm 100 and print the result
x=rnorm(100)
print(JB(x))
```

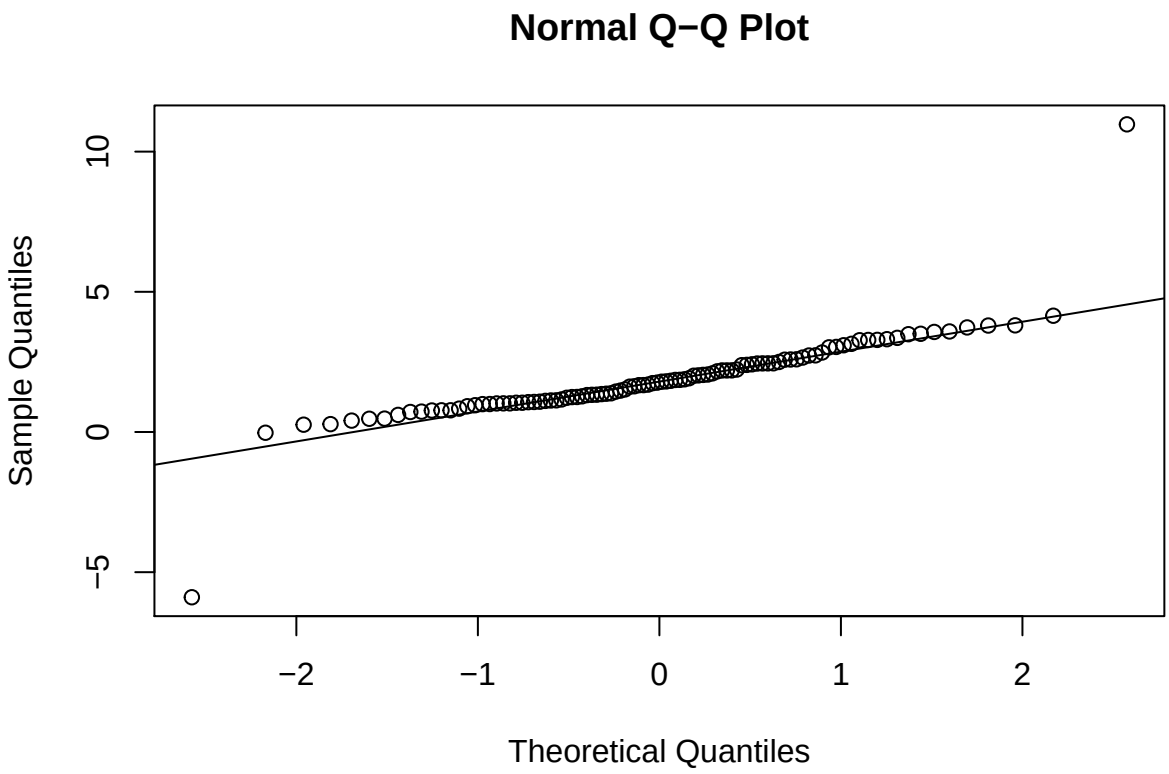
```
## [1] 3.965268
```

b) We simulate data sets x1,x2,x3 with eps values of 0, 0.01, 0.05 resp as follows

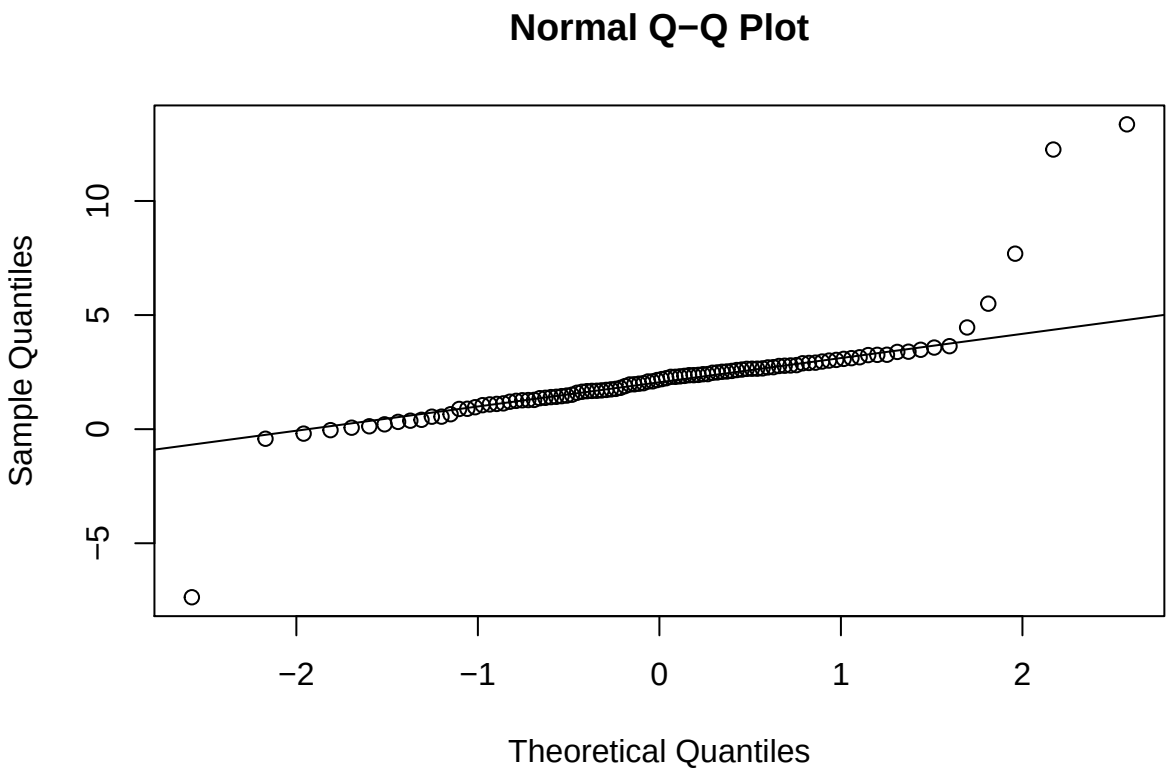
```
#simulate first data with n=100 and epsilon 0
n=100
eps=0
x1=rnorm(n, 2, 1+5*rbinom(n,1,eps))
#plot qqnorm and qqline for x1
qqnorm(x1)
qqline(x1)
```




```
#observe that for the normal qqplot of x1, the plot appears to have
#heavier tails, but is still somewhat normal
#the graph allows visuallization of several outliers in the tail of the plot

#simulate the second data with epsilon 0.01
eps=0.01
x2=rnorm(n, 2, 1+5*rbinom(n,1,eps))
#plot qqnorm and qqline for x2
qqnorm(x2)
qqline(x2)
```




```
#observe that the normal qqplot of x2 appears to be mostly normal except
#the tails past the 2nd theoretical quantile
#have signifigant tails and outliers, however most of the plot is normal
#within the first two quantiles

#simulatte the third data with epsilon 0.05
eps=0.05
x3=rnorm(n, 2, 1+5*rbinom(n,1,eps))
#plot qqnorm and qqline for x3
qqnorm(x3)
qqline(x3)
```



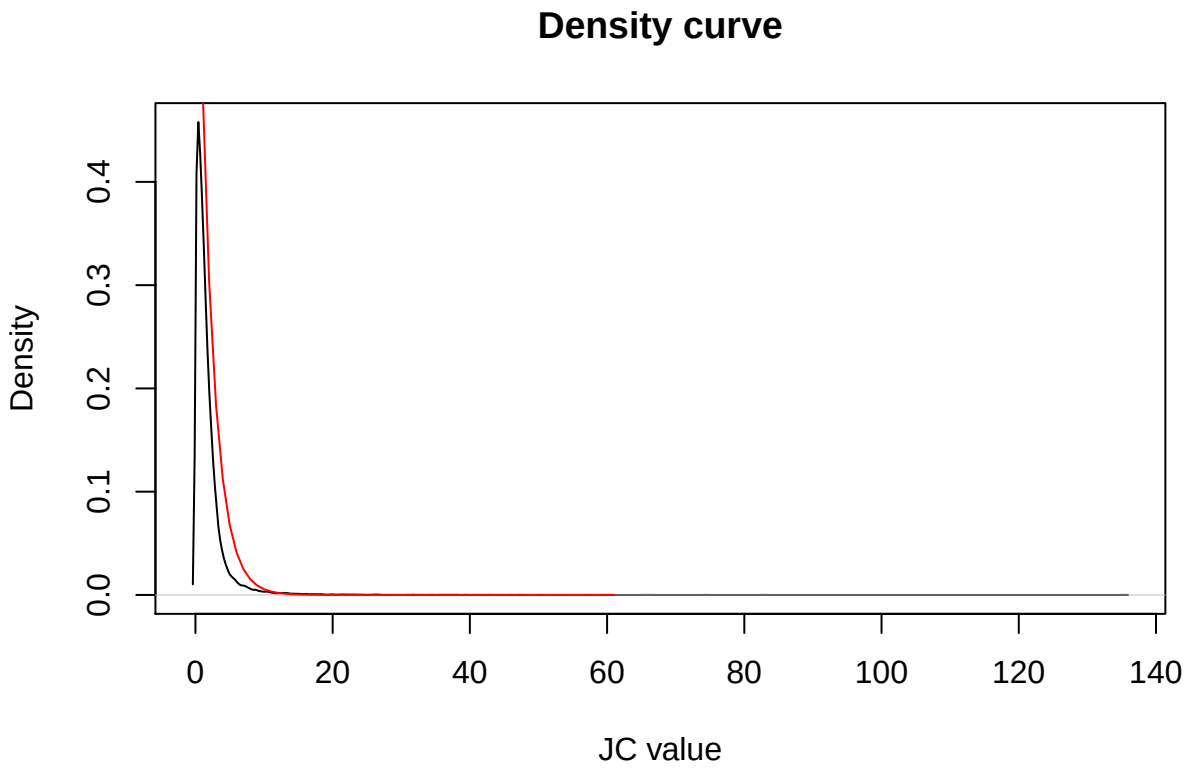

```
#observe that the normal qqplot of x3 is mostly normal except for the tails
#past the 2nd theoretical qantile
#some tails and outliers past the second quantile in the plot
```

c) We us MC simulation to siumulate JBn values under normal assumption as follows

```
#define the monte carlo simulation of JBn
JB.MC<-function(n,K=50000){
  #create numeric storage for the JBn values
  output<-numeric(K)
  #populate with JBn values using JB function from part a
  for (j in 1:K){
    x=rnorm(n)
    output[j]=JB(x)
  }
  output
}
```

d) We generate output data from JB.MC with n =100 and plot histogram as follows

```
#generate the monte carlo values of JBn with the function from c and plot its
#density histogram
x=JB.MC(100)
plot(density(x),xlab="JC value",
   main="Density curve" )
#add chi squared density curve of 2 df in red
lines(dchisq(0:60,df=2), col=2)
```




```
#observe that they have similar density except the chi sqaured curve isnt as
#steep and the tail is lighter
# the left side is similar denisty however the chi sqaured is asymptotic
#approaching infinity at 0 compared to
#the density of the JB plot which starts at zero
```

e) We Use JB from AB with values from x1,x2,x3 and find p-values though X^2

```r
#we use the JB function with each of x1,x2,x3 to compute their JBn values
r1 = JB(x1)
r2 = JB(x2)
r3 = JB(x3)
r1
```

```
## [1] 0.1335143
```

```r
r2
```

```
## [1] 1173.633
```

```r
r3
```

```
## [1] 882.3258
```

```r
#we compute the p-values using the chisq with 2 df
p1=pchisq(r1, df=2)
p2=pchisq(r2, df=2)
p3=pchisq(r3, df=2)
p1
```

```
## [1] 0.06457768
```

```r
p2
```

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

```r
p3
```

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

```r
#now we compute the p values using the output data set from part d
P = ecdf(x)
p1.2=P(r1)
p2.2=P(r2)
p3.2=P(r3)
p1.2
```

```
## [1] 0.06828
```

```r
p2.2
```

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

```r
p3.2
```

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

```
#observing the results of the p values, it appears that both the chi
#squared p values and the output data
#roughly match the graphical observation of the values obtained from
#the JB function
#The differences between them are clearly observed as by the
#differences in the left tail of both functions
#thus for x1 the p differ due to the left tail of both distributions
#and the chi being shifted right
#also observe the p-vals for x2 and x3 are similar since both are far
#to the right and are thus rare
```

Question 2

a) We write theta estimator function and use it on huron mean as follows

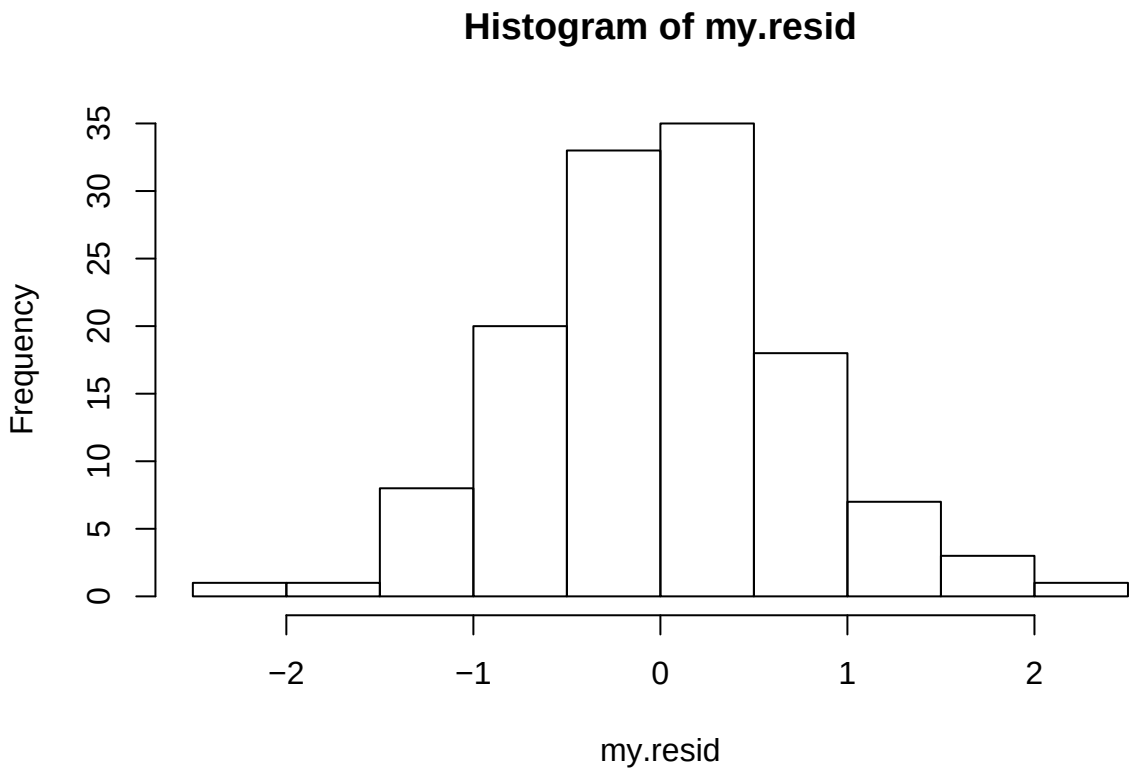
```
source(file = "huron.R")
#we implement the theta hat estimator function for parameter x
theta.est<-function(x){
  n=length(x)
  x.prev = c(0,x[1:n-1])
  #computes previous X value and calculates output of theta hat
  output = (sum(x.prev*x))/(sum(x^2))
}

#tests the value for the huron-mean(huron) from the huron data set
my.est=theta.est(huron-mean(huron))
my.est
```

```
## [1] 0.8314656
```

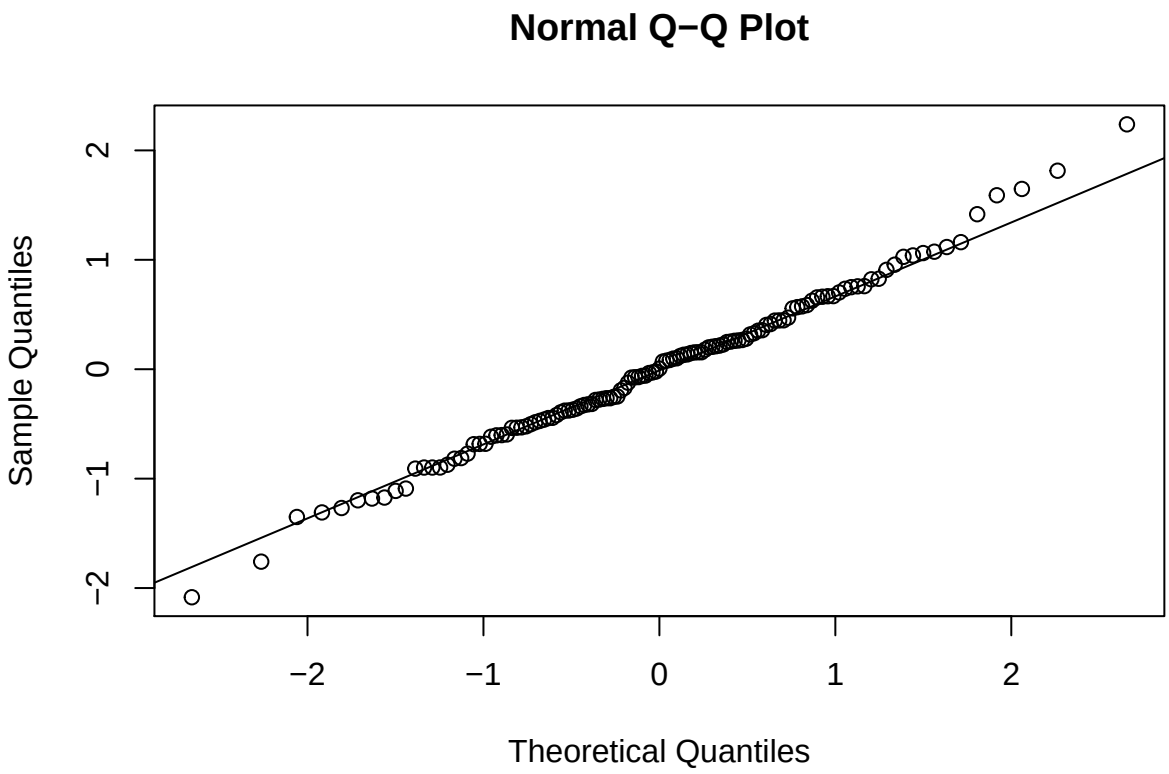
b) We write a residual estimator function my.resid and check distribution with a histogram

```
# we estimate the residuals for the huron mean
n=length(huron-mean(huron))
x=huron-mean(huron)
#compute xprev
x.prev = c(0,x[1:n-1])
#we compute the mean of all the epsilon values
eps.mean = sum(x-my.est*x.prev)/n
#now populate the array of epsilon
eps.arr= x-my.est*x.prev
#save the sample mean centered residuals
my.resid = eps.arr-eps.mean
```

```r
#check distribution with histogram and qqplot with qqline
hist(my.resid)
```




```r
qqnorm(my.resid)
qqline(my.resid)
```




```
#observe that both the histogram and the qqplot are mostly normally
#distributed
```

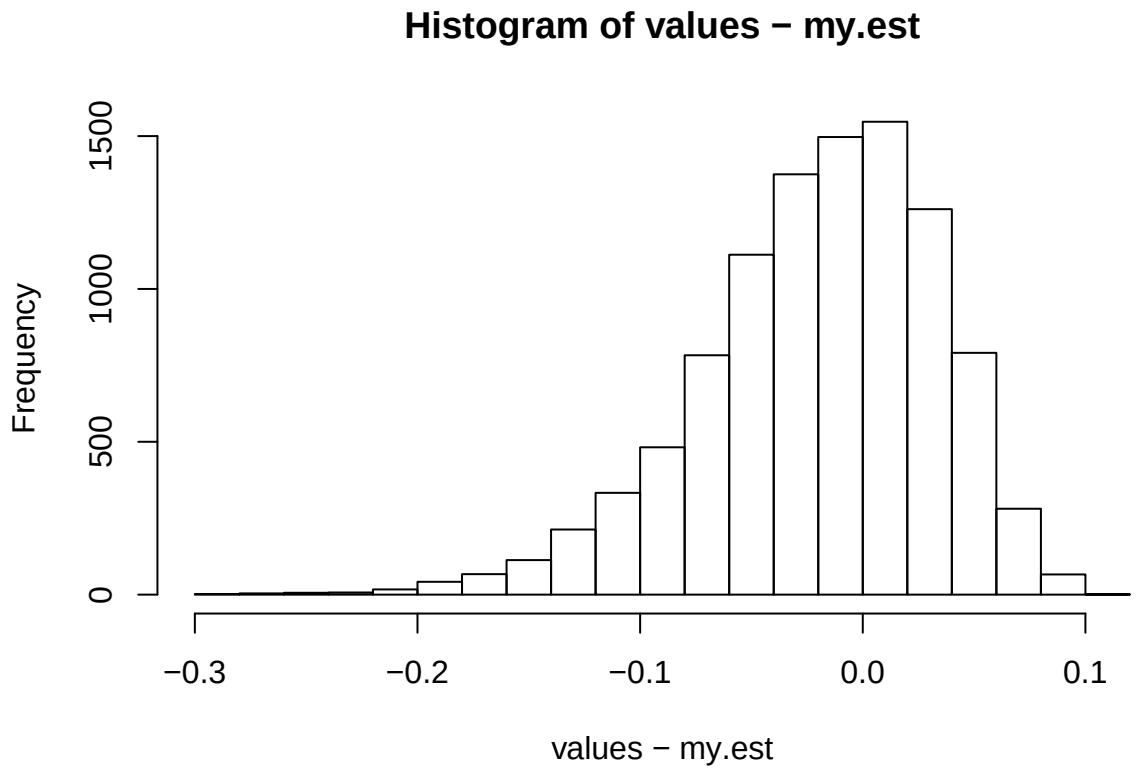
c) We write a function one.boot to compute a bootsrap estimation of theta as follows

```
#define bootstrap function to bootstrap an estimate of theta
one.boot<-function(eps=my.resid,theta=my.est){
  n = length(eps)
  #resample the epsilon value
  eps.star=sample(eps, n, replace=TRUE)
  xstar = numeric(n)
  #create numeric array for xstar and make first element 0
  xstar[1]=0
  #populate the xstar value using the epsilon star array
  for (j in 2:n){
    xstar[j]= theta*xstar[j-1]+eps.star[j]
  }
  #use the theta estimator from a to return the lkeast estimator for
  #theta hat star
  theta.est(xstar)
}
```

```
#use replicate to obtain 10000 bootraps and return vector array
output=replicate(n=10000, one.boot(), simplify=FALSE)
```

d) We find the histogram of theta hat star - theta hat and theta and find 95% CI

```
# convert the output to a vector
values<-as.vector(unlist(output))
#we produce a histogram of theta hat star - theta hat
hist(values-my.est)
```




```
#now we compute a 95% CI
a=mean(values)
a
```

```
## [1] 0.8114079
```

```
s=sd(values)
n=length(values)
error <- qt(0.975,df=n-1)*s/sqrt(n)
left <- a-error
```

```
right <- a+error
print("The left and right boundaries for a 95% CI are")
```

```
## [1] "The left and right boundaries for a 95% CI are"
```

```
left
```

```
## [1] 0.8103659
```

```
right
```

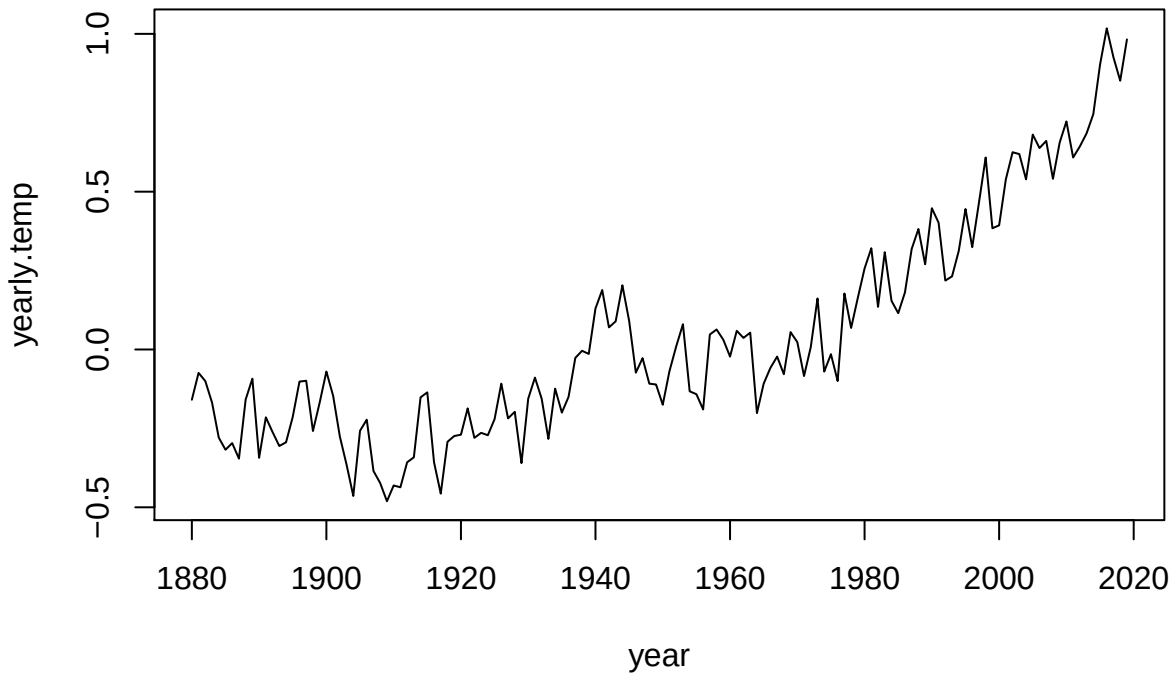
```
## [1] 0.8124499
```

```
#observe the histogram has a single peak around 0 with a left skewed
#distribution
```

Question 3

a) We import the dataset into R as a data frame and keep only the first 12 month cols

```
#import data as dataframe
mydata <-read.csv('GLB.Ts_dSST.csv')
#make new dataframe for the desired year, jan-dec cols
df = mydata[c(2:13)]
year<-1880:2019
#use apply function to get yearly mean temperature
yearly.temp = apply(df,c(1), mean,main="Yearly temperature plot" )
#plot yearly temperature
plot(year,yearly.temp, type="l")
```

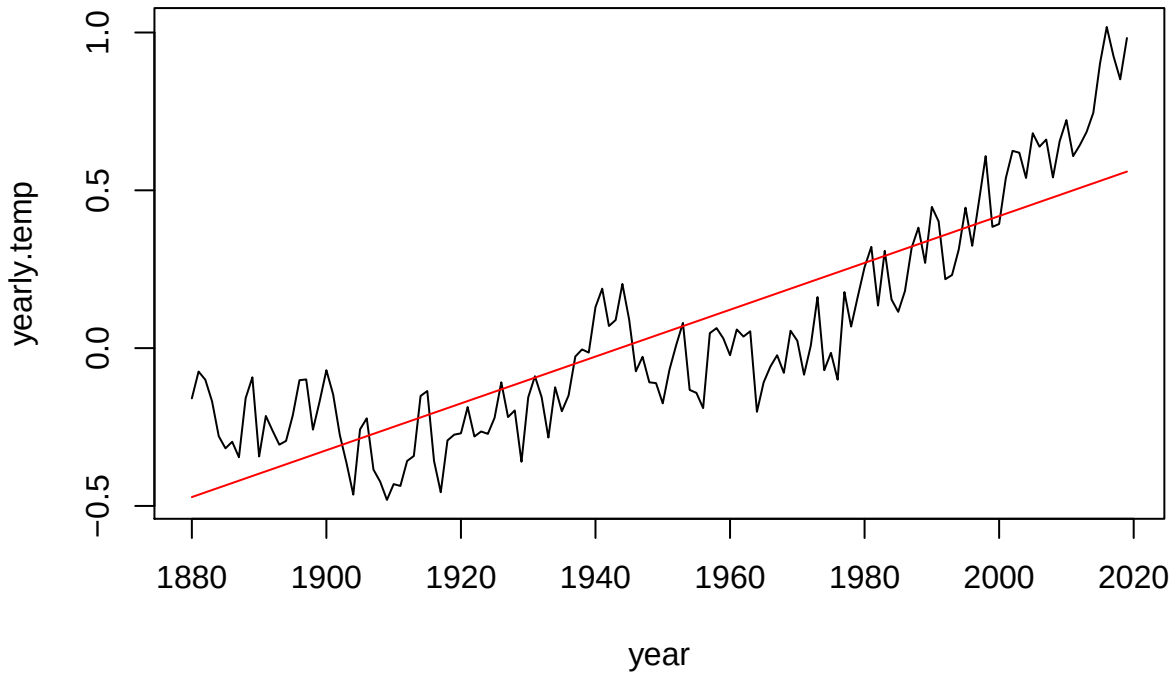



```
#observe that the temperature appears to be increasing ober time
#and the slope appears to be getting steeper
# as the years increase
```

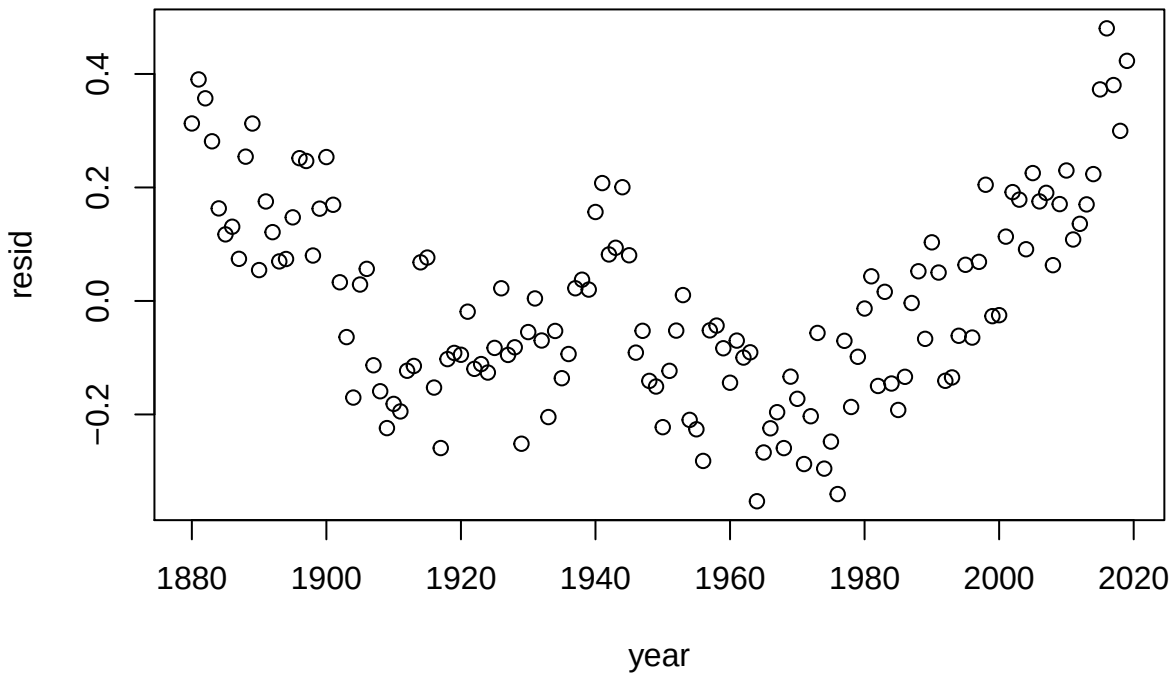
b We construct an objective function based on sum of square as follows

```
#objective function based on sum of squares
sumofsqr<-function(a,b){
  sum((yearly.temp-a-b*year)^2)
}
#wrapper function
g<-function(p){
  sumofsqr(p[1],p[2])
}
#estimate a, b param using nlminb function
ls.est = nlminb(c(-10,0.1),g)
#convert params to vector and obtain best a and b params
params=as.vector(unlist(ls.est[1]))
a=params[1]
b=params[2]
#compute residuals
```

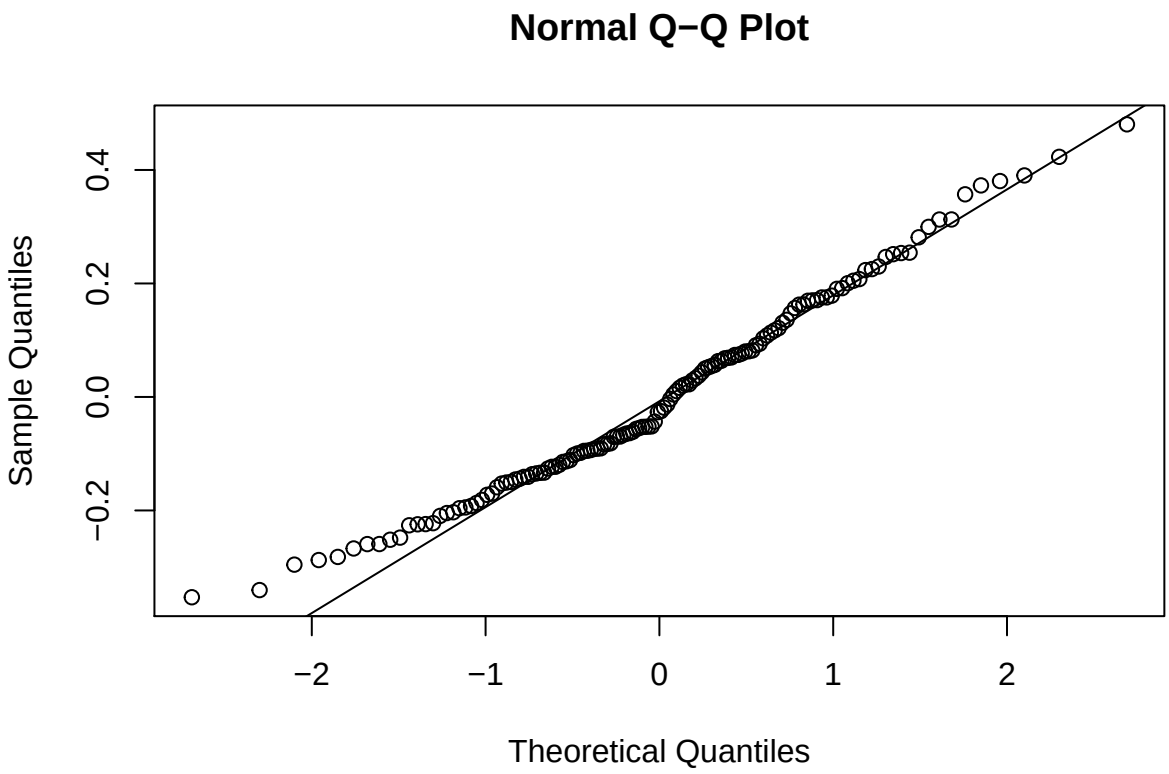
```
resid=yearly.temp-a-b*year
#plot the best temperatures
plot(year,yearly.temp, type="l")
#use params for fitted line and plot in red
fitted1=(a+b*year)
lines(year,fitted1,col=2)
```




```
#the fitted line is a decent basic approximation but overall
#very poor fitting to the curve
#plot residuals
plot(year,resid)
```




```
#the residuals appppear to form a W, with more negative values
#in the middle years and positive in the outer years
#use qqnorm and qqline to see if resid is normally distributed
qqnorm(resid)
qqline(resid)
```

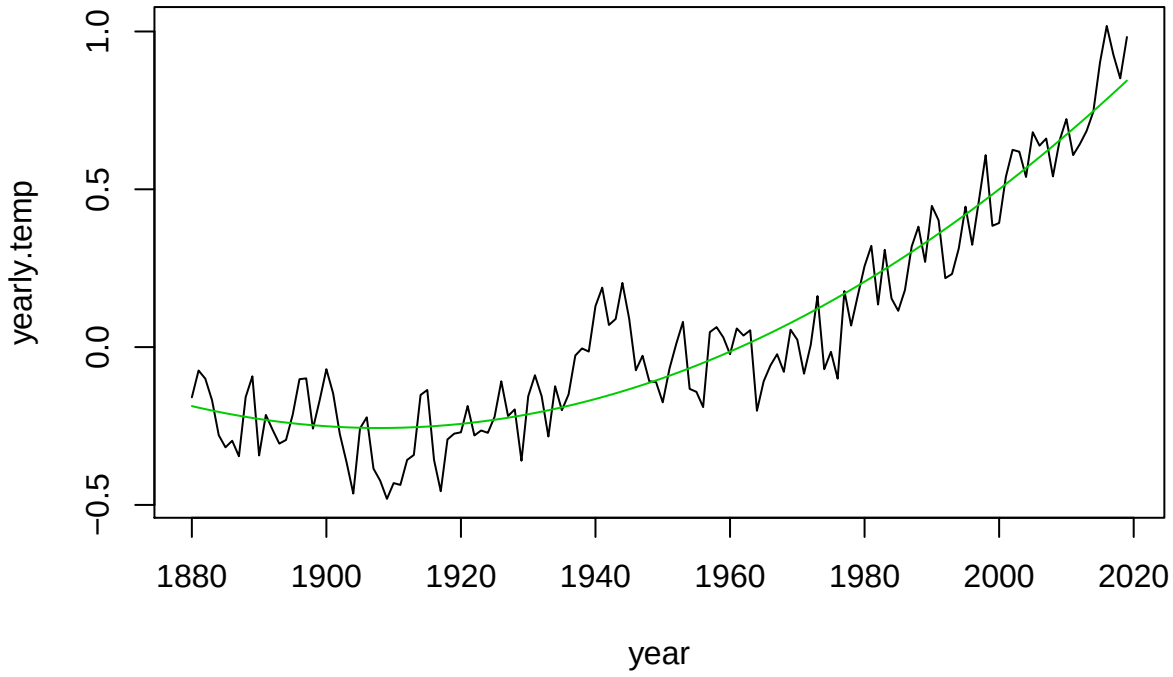



```
#the normal qqplot appears to be fairly normally distributed
#with light tails on either end
```

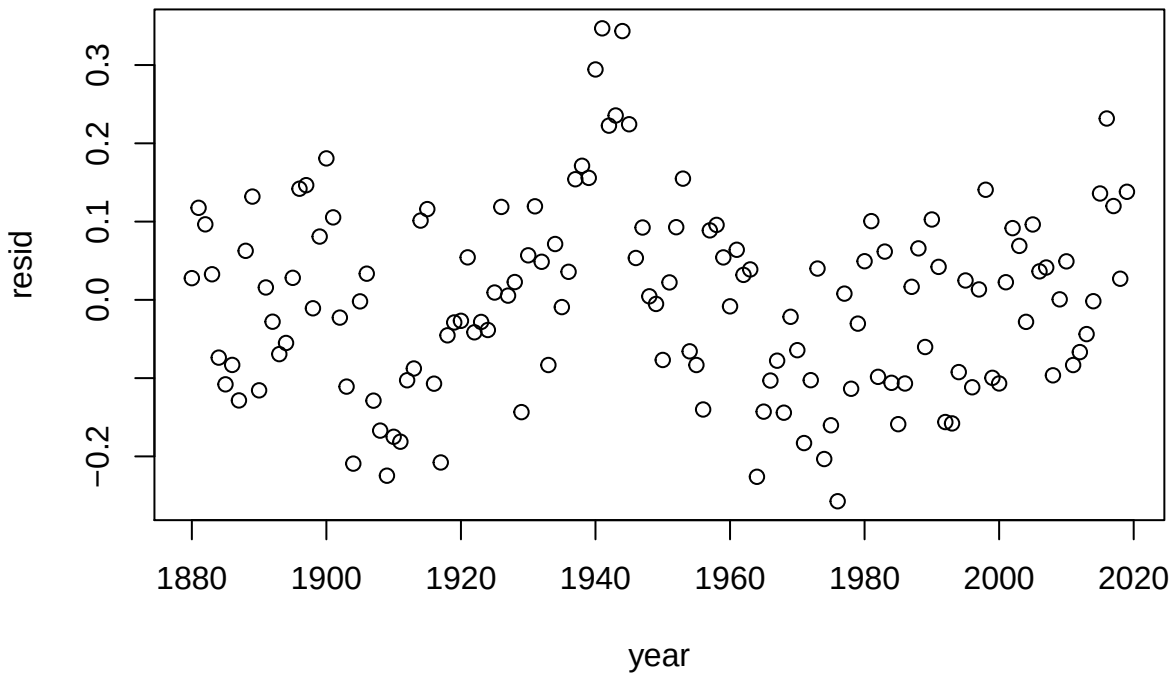
c We do the same as above but change the objective function as defined in c as follows

```
#define the objective function based on the new formula
objfunct2<-function(a,b,c){
  sum((yearly.temp-a-b*year-c*year^2)^2)
}
#wrapper function
g<-function(p){
  objfunct2(p[1],p[2],p[3])
}
#use nlimb to obtain estimations for a,b,c params
ls.est = nlminb(c(300,-1,0.1),g, scale=c(1,300,1000))
params=as.vector(unlist(ls.est[1]))
#convert values to vector and store each param value
a=params[1]
b=params[2]
c=params[3]
#compute residuals
```

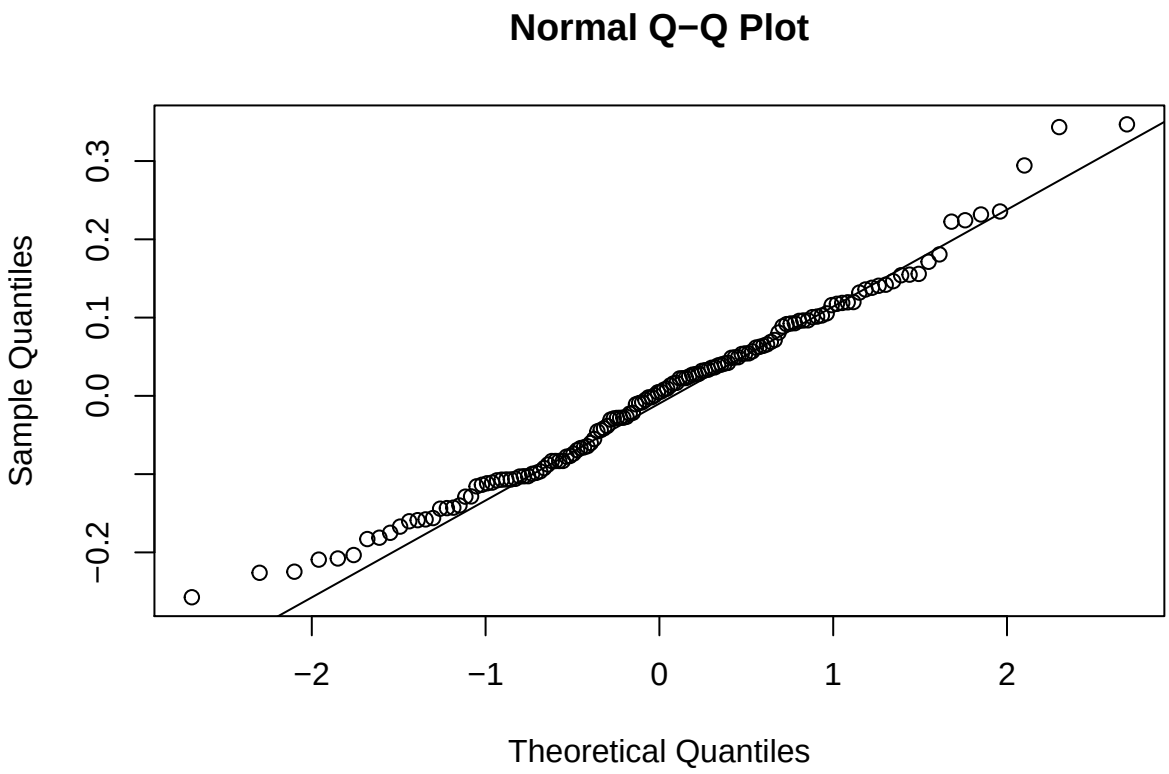
```
resid=yearly.temp-a-b*year-c*year^2
#plot the fitted curve in green
plot(year,yearly.temp, type="l")
fitted2=(a+b*year+c*year^2)
lines(year,fitted2,col=3)
```




```
#observe the fitted curve matches the distribution well
#and shows the increasing trend of temp increase better
#plot the residuals
plot(year,resid)
```




```
# the residual plot forms a butterfly with a symmetric
#form where most negative are at middle value
#and positive around zero
#qqplot of the values with qqline
qqnorm(resid)
qqline(resid)
```

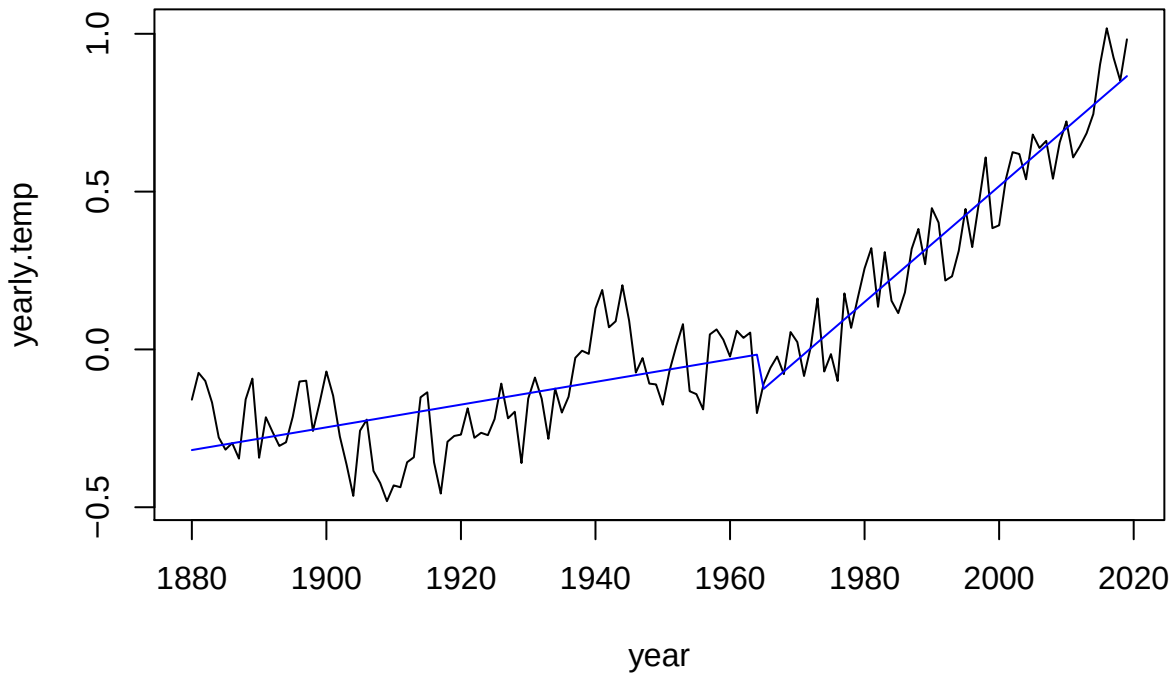



```
#the normal qqplot appears to be fairly normally distributed
#with light tails on either end
```

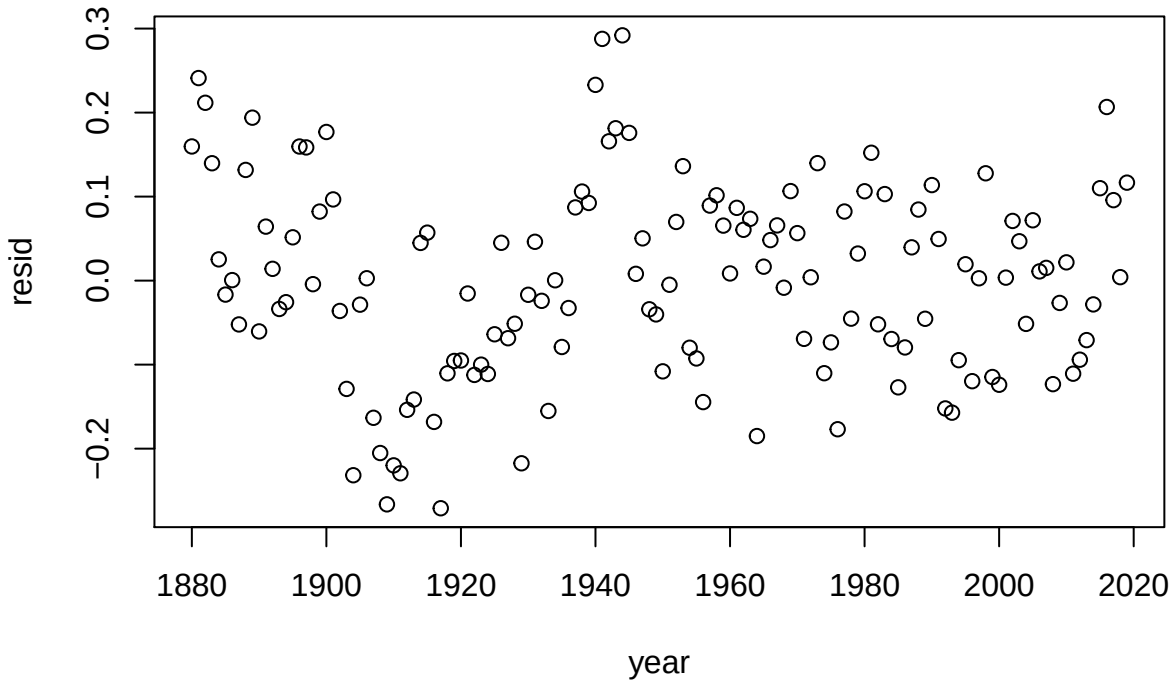
c We do the same as above but change the objective function as defined in c as follows

```
#create dummy vector
dummy= c(rep(0,85), rep(1,55))
#define objective function defined from the question
objfunct3<-function(a,b,c,d){
  sum((yearly.temp-a-b*year-c*dummy-d*dummy*year)^2)
}
#wrapper function
g<-function(p){
  objfunct3(p[1],p[2],p[3],p[4])
}
#estimate the a,b,c,d values
ls.est = nlminb(c(-1,1,-1,1),g)
params=as.vector(unlist(ls.est[1]))
#convert values to vector and store each param value
a=params[1]
b=params[2]
```

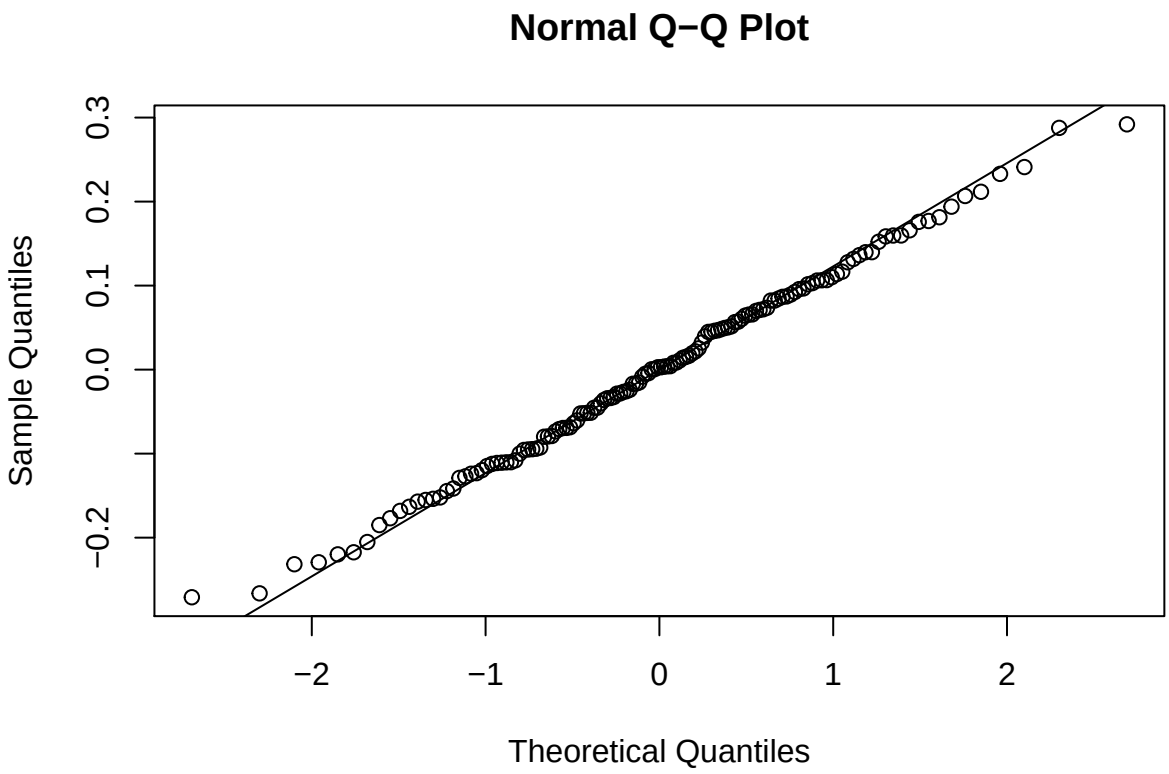
```
c=params[3]
d=params[4]
#compute the residuals
resid=yearly.temp-a-b*year-c*dummy-d*dummy*year
#plot the fitted curve in blue
plot(year,yearly.temp, type="l")
fitted3=(a+b*year+c*dummy+d*dummy*year)
lines(year,fitted3,col=4)
```




```
#observe the fitted curve fits the model very well matching
#the two distinct slopes of the curve
#plot the residuals
plot(year,resid)
```



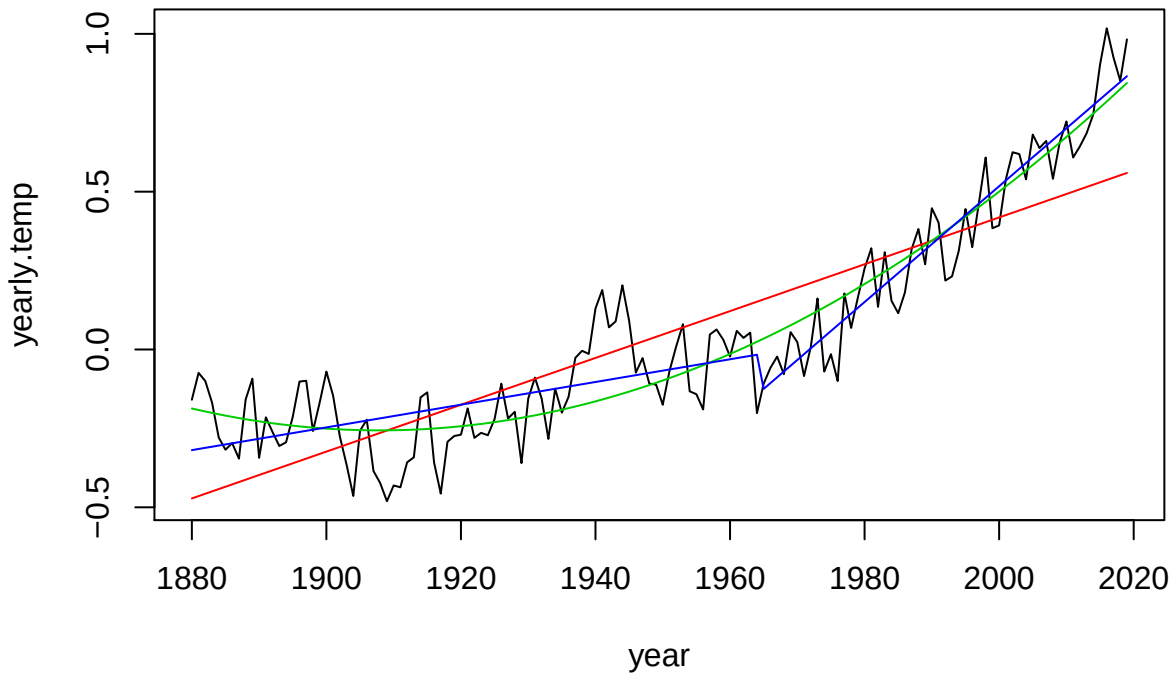

```
#similar butterfly slightly less symettric distribution
#with more negative values around moderate
#values and positive at extremes
#plot qqnorm and qqline
qqnorm(resid)
qqline(resid)
```




```
#normally distributed curve with slight tails at the ends
```

d) We compare the similarities and differences between the three models graphically as follows

```
#we plot the three curves on the same line in red,green,blue
plot(year,yearly.temp, type="l")
lines(year,fitted1,col=2)
lines(year,fitted2,col=3)
lines(year,fitted3,col=4)
```




```
#comparing the three models thhe best fitted are the second and third model
#observe the green and blue line match the increasing slope of curve
#the third model appears to slightely match the curve better with
# the two different slopes so overall that is the best fitted bodel
#overall conclusion is the rate of increase in temperature is increasing
#over time causing an increasing rate of tempurate gain over time
```