# Normal_GLM.r

*sergazy*

*Wed Nov 21 12:12:00 2018*

```r
setwd("/Users/sergazy/Downloads/Fall 2018 courses and all files/Eric's class/11:06:2018 Normal GLM/")
load("Normal_GLM.Rdata")


#part a)
###
#Fit the model and test whether the quadratic term is significant at
#5% significance. Furthermore, in the quadratic model, make a plot of
#the residuals and discuss whether the data is homoscedastic.
###
x2=x^2
fit=lm(y~x+x2)
summary(fit)
```

```
##
## Call:
## lm(formula = y ~ x + x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9940 -0.6093  0.0508  0.5487  4.1157
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.4348     0.5395   19.34  < 2e-16 ***
## x           -27.1207     2.2597  -12.00 6.46e-16 ***
## x2           27.5193     2.0916   13.16  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.08 on 47 degrees of freedom
## Multiple R-squared:  0.793,  Adjusted R-squared:  0.7842
## F-statistic: 90.01 on 2 and 47 DF,  p-value: < 2.2e-16
```

```r
#from the summary we see that x2 is indeed significant.
#Let us see what F test looks like.
fit_H0=lm(y~x)
summary(fit_H0)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.4054 -1.2862 -0.6689  0.2760  8.5565
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.8034     0.7033   6.829 1.34e-08 ***
## x             1.5636     1.2729   1.228    0.225
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.312 on 48 degrees of freedom
## Multiple R-squared:  0.03047,    Adjusted R-squared:  0.01028
## F-statistic: 1.509 on 1 and 48 DF,  p-value: 0.2253
```

```
RSS=sum(residuals(fit)^2)
RSS
```

```
## [1] 54.77585
```

```
RSS_H0=sum(residuals(fit_H0)^2)
RSS_H0
```

```
## [1] 256.521
```

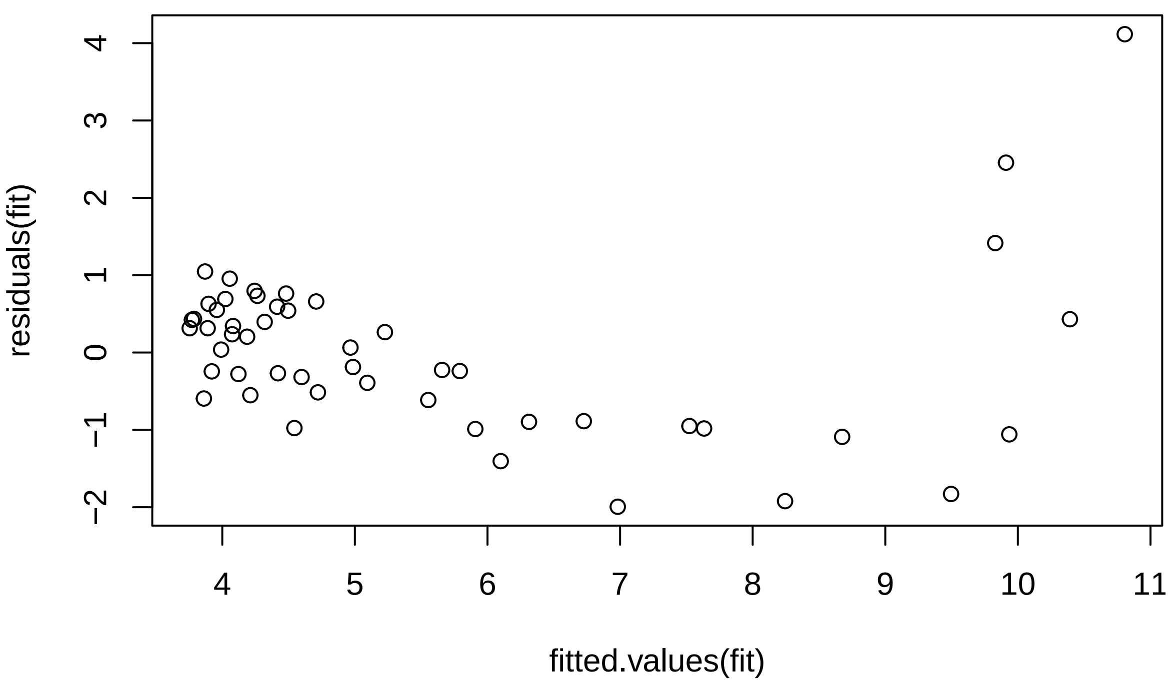
```
F=((RSS_H0-RSS)/RSS)*((length(x)-2)/1)
F
```

```
## [1] 176.789
```

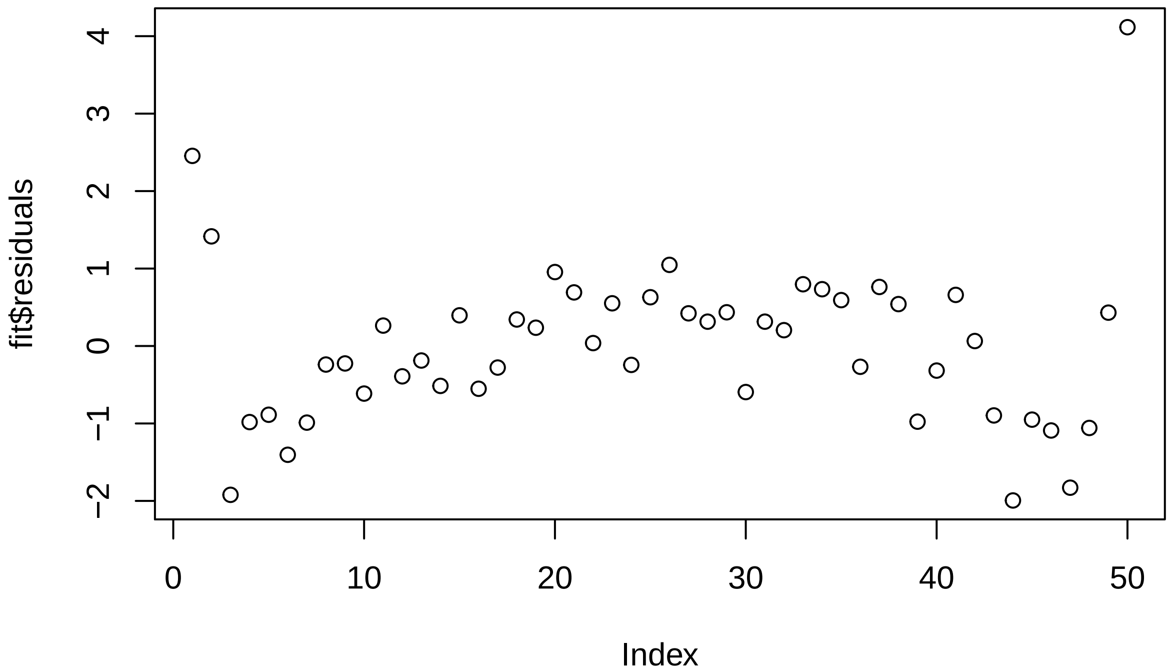
```
pval1=pf(F,1,length(x)-2,lower.tail = FALSE)
pval1
```

```
## [1] 1.037713e-17
```

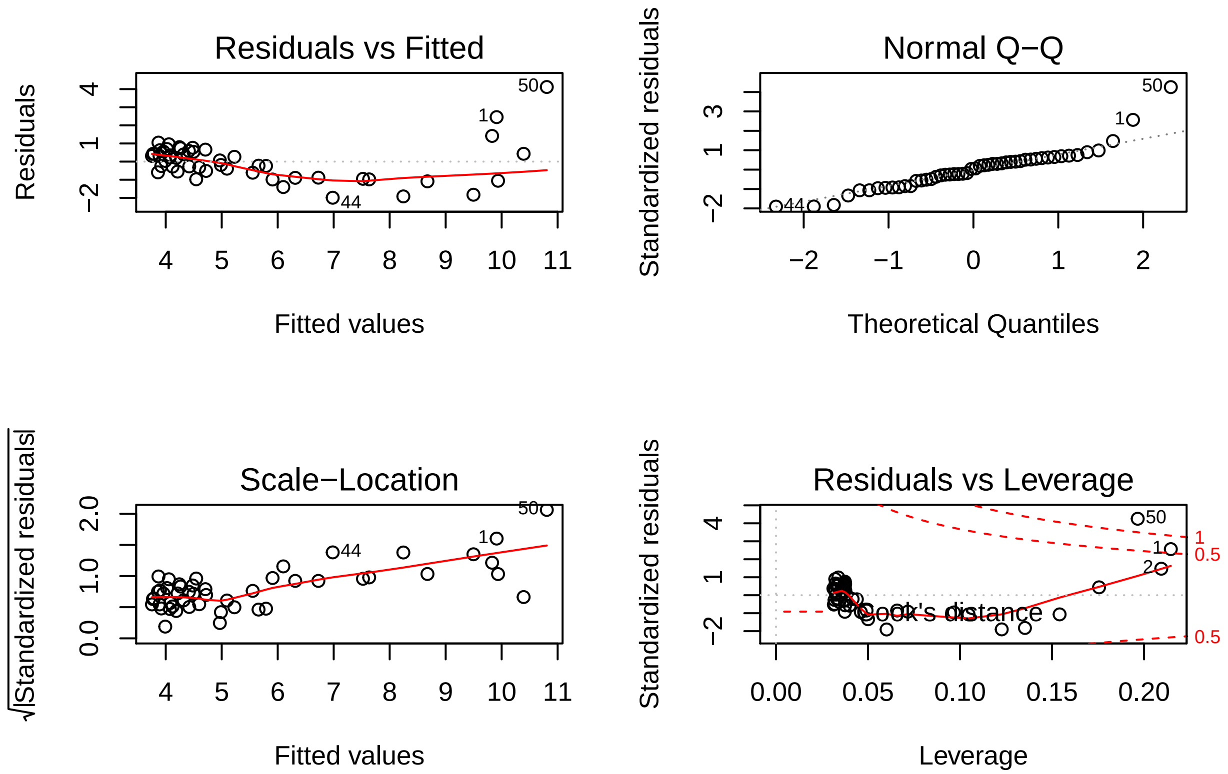
```
#this means we reject the H0. That explains x^2 is indeed significant.


plot(fitted.values(fit),residuals(fit))
```




```
plot(fit$residuals)
```




```
par(mfrow=c(2,2))
plot(fit)
```




```
par(mfrow=c(1,1))
#by looking the plots we can see that data is clearly heterscedastic.

#part b).
###
```

```r
#Calculate the log-likelihood of the data using the function lik, for
#the parameters corresponding to the model you fitted in (a)
#(this is possible because model (a) is a submodel of the normal GLM: explain!).
#Think about how to define the matrices X1 and X2. Also calculate the gradient
#and discuss your findings (the gradient has a special feature that you can explain!).
###


#we estimate for the variance sigma^2
sighatsq = RSS/(length(x)-2)
sighatsq
```

```
## [1] 1.141164
```

```r
#mu_i=eta1_i/eta2_i=beta_10+beta_11x_i+beta_12x_i (this beta will come from part a)
beta1 = fit$coefficients/sighatsq
beta1
```

```
## (Intercept)           x          x2
##    9.143971  -23.765857   24.115123
```

```r
beta2=c(1/sighatsq,0,0)
names(beta2)=names(beta1)
beta2
```

```
## (Intercept)           x          x2
##   0.8762986   0.0000000   0.0000000
```

```r
X1 = model.matrix(fit)
X2 = model.matrix(fit)

#This file contains functions relating to the log-likelihood
#of the normal exponential family, with a linear model for each
#of the two natural parameters.

lik=function(beta1,beta2,y,X1,X2){
  #y consists of n observations
  #X1 is an n-by-p1 matrix, with p1 covariates for each subject
  #X2 is an n-by-p2 matrix, with p2 covariates for each subject
  #beta1 is a vector of length p1 OR a m-by-p1 matrix
  #beta2 is a vector of length p2 OR a m-by-p2 matrix
  if (is.matrix(beta1)){
    beta1=t(beta1)
    beta2=t(beta2)
  }
  L=t(y)%*%X1%*%beta1 - t(y^2)%*%X2%*%beta2/2
  if (is.matrix(beta1)){
    L=L - 0.5*colSums((X1%*%beta1)^2/(X2%*%beta2))+0.5*colSums(log(X2%*%beta2))
    L=t(L)
    colnames(L)="Log-likelihood"
  } else {
    L=L - 0.5*sum((X1%*%beta1)^2/(X2%*%beta2))+0.5*sum(log(X2%*%beta2))
    L=L[1,1]
  }
  return(L-0.5*log(2*pi)*dim(X1)[1]) #returns a vector with m values of the likelihood
}
```

```r
#our log likelyhood is
lik(beta1,beta2,y,X1,X2)
```

```
## [1] -73.24814
```

```r
#our gradient is
Dlik=function(beta1,beta2,y,X1,X2){
  #y consists of n observations
  #X1 is an n-by-p1 matrix, with p1 covariates for each subject
  #X2 is an n-by-p2 matrix, with p2 covariates for each subject
  #beta1 is a vector of length p1 OR a m-by-p1 matrix
  #beta2 is a vector of length p2 OR a m-by-p2 matrix

  if (is.matrix(beta1)){
    m=dim(beta1)[1]
    L=t(t(rep(1,m)))%*%c(t(y)%*%X1, - t(y^2)%*%X2/2)
    L=L - cbind(t((X1%*%t(beta1))/(X2%*%t(beta2)))%*%X1,
                -0.5*t((X1%*%t(beta1))^2/(X2%*%t(beta2))^2)%*%X2
                -0.5*t(1/(X2%*%t(beta2)))%*%X2)
  } else {
    L=c(t(y)%*%X1, - t(y^2)%*%X2/2)
    L=L - c(t((X1%*%beta1)/(X2%*%beta2))%*%X1,
            -0.5*t((X1%*%beta1)^2/(X2%*%beta2)^2)%*%X2 -0.5*t(1/(X2%*%beta2))%*%X2)
  }
  if (is.matrix(beta1)) {
    colnames(L)=c(colnames(beta1),colnames(beta2))
    rownames(L)=rownames(beta1)} else {
      names(L)=c(names(beta1),names(beta2))
    }
  return(L) #returns m-by-(p1+p2) matrix with the m gradient vectors
}

Dlik(beta1,beta2,y,X1,X2)
```

```
##   (Intercept)             x            x2   (Intercept)             x
##  1.705303e-13  8.526513e-14  5.684342e-14  1.141164e+00 -1.510741e+00
##            x2
## -9.045551e+00
```

```r
#Dlik function shows that it achieves its maximum only for beta1 but not beta2.
#That is why, we need to improve our model to get maximum for both beta1 and beta2.




#part c
###
#Implement the Newton-Raphson optimization to find the MLE of (beta1,beta2).
#Be careful: not all values for\beta2 are allowed, since \eta2 has to be positive!
#If you do not succeed, use the R-function optim to find the MLE. Note that in
#optim, you can also give the gradient function! Plot the data together with the
#estimated expectation according to the normal GLM model and according to the
#standard linear model. Also plot the residuals (using GLM) with plus and minus
#the estimated standard error at each value ofx. You could also make this last
#plot for the standard linear model.
###
```

```r
D2lik=function(beta1,beta2,y,X1,X2){
  #y consists of n observations
  #X1 is an n-by-p1 matrix, with p1 covariates for each subject
  #X2 is an n-by-p2 matrix, with p2 covariates for each subject
  #beta1 is a vector of length p1 (only one parameter!)
  #beta2 is a vector of length p2 (only one parameter!)

  p1=length(beta1)
  p2=length(beta2)
  eta1=X1%*%beta1
  eta2=X2%*%beta2
  L=matrix(0,p1+p2,p1+p2)
  for (m in 1:(p1+p2)){
    for(n in 1:(p1+p2)){
      if (m<=p1 & n<=p1){
        L[m,n]=-sum(X1[,m]*X1[,n]/eta2)
      }
      if (m<=p1 & n>p1){
        L[m,n]=sum(eta1*X1[,m]*X2[,n-p1]/eta2^2)
      }
      if (m>p1 & n<=p1){
        L[m,n]=sum(eta1*X2[,m-p1]*X1[,n]/eta2^2)
      }
      if (m>p1 & n>p1){
        L[m,n]=-sum(X2[,m-p1]*X2[,n-p1]*(eta1^2/eta2^3 + 0.5/eta2^2))
      }
    }
  }
  colnames(L)=c(names(beta1),names(beta2))
  rownames(L)=c(names(beta1),names(beta2))

  return(L) #returns (p1+p2)-by-(p1+p2) second derivative matrix
}


D1 = Dlik(beta1,beta2,y,X1,X2)
D1
```

```
##   (Intercept)             x            x2   (Intercept)             x
##  1.705303e-13  8.526513e-14  5.684342e-14  1.141164e+00 -1.510741e+00
##            x2
## -9.045551e+00
```

```r
D2 = D2lik(beta1,beta2,y,X1,X2)
D2
```

```
##             (Intercept)         x         x2 (Intercept)          x
## (Intercept)   -57.05818 -27.91341 -17.419155    317.7195  161.31614
## x             -27.91341 -17.41916 -12.444590    161.3161  111.49554
## x2            -17.41916 -12.44459  -9.710897    111.4955   87.85731
## (Intercept)   317.71955 161.31614 111.495541  -2041.1532 -1093.15425
## x             161.31614 111.49554  87.857309  -1093.1543  -835.75650
## x2            111.49554  87.85731  74.316641   -835.7565  -712.57303
##                      x2
```

```
## (Intercept)  111.49554
## x             87.85731
## x2            74.31664
## (Intercept) -835.75650
## x           -712.57303
## x2          -637.84238
```

```r
err=1
h_m = -(solve(D2))%*%(D1)
h_m
```

```
##                    [,1]
## (Intercept) -10.997560
## x            56.019710
## x2          -56.641650
## (Intercept)  -1.310334
## x             7.232384
## x2           -7.182624
```

```r
#Now calculate l(beta+hm) and check if it is bigger than l(beta). If not,
#then divide hm by 2 (so replace hm by h_m/2) and check again if l(beta + h_m)
#is bigger than l(beta). Repeat this until you find the next value of the
#parameter with larger value for l, and hen repeat the whole procedure
#until you met a certain criterion. This could be a very small gradient or
#a very small increase of the function l.

# We will do Newton-Raphson optimization.
while(err > 10^-15){
  D1 = Dlik((beta1),(beta2),y,X1,X2)
  D2 = D2lik(beta1,beta2,y,X2,X2)
  h_m = -(solve(D2))%*%(D1)
  while(sum(X2%*%(beta2+h_m[4:6])>0)<length(x)){
    h_m = (h_m)/2
  }
  while(lik(beta1+h_m[1:3],beta2+h_m[4:6],y,X1,X2) - lik(beta1,beta2,y,X1,X2)<0){
    h_m = (h_m)/2
  }
  err = abs(lik(beta1+h_m[1:3],beta2+h_m[4:6],y,X1,X2) - lik(beta1,beta2,y,X1,X2))
  beta1 = beta1+h_m[1:3]
  beta2 = beta2+h_m[4:6]
  print(err)
}
```

```
## [1] 15.59637
## [1] 10.63484
## [1] 6.27827
## [1] 2.279274
## [1] 0.3134914
## [1] 0.00670344
## [1] 3.489444e-06
## [1] 5.115908e-13
## [1] 9.30811e-13
## [1] 2.842171e-14
## [1] 0
```

```r
print(beta1)
```

```
## (Intercept)           x          x2
##    11.45196    43.89866   -42.74408
```

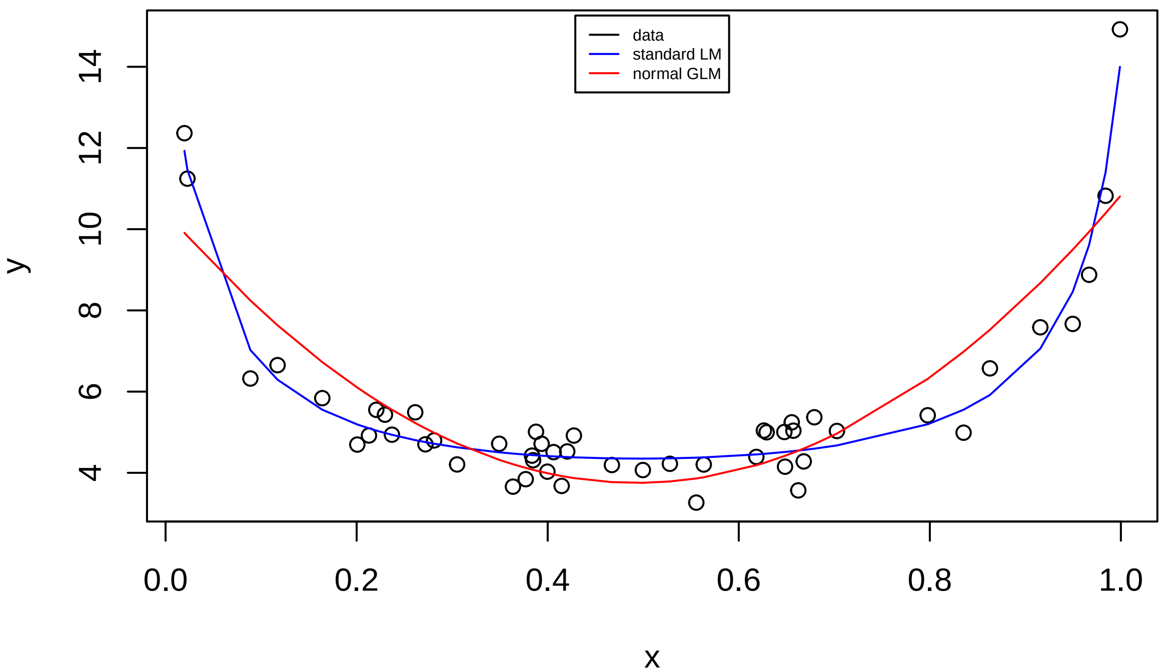
```r
print(beta2)
```

```
## (Intercept)           x          x2
##     0.68391    17.95136   -17.74880
```

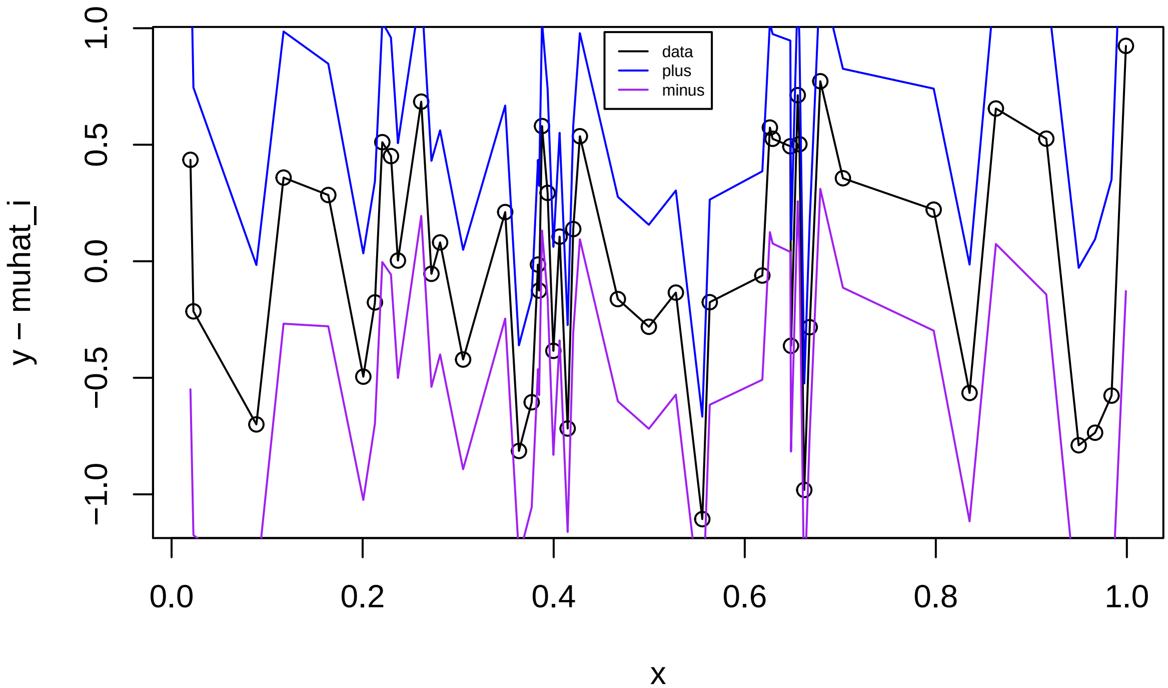
```r
print(err)
```

```
## [1] 0
```

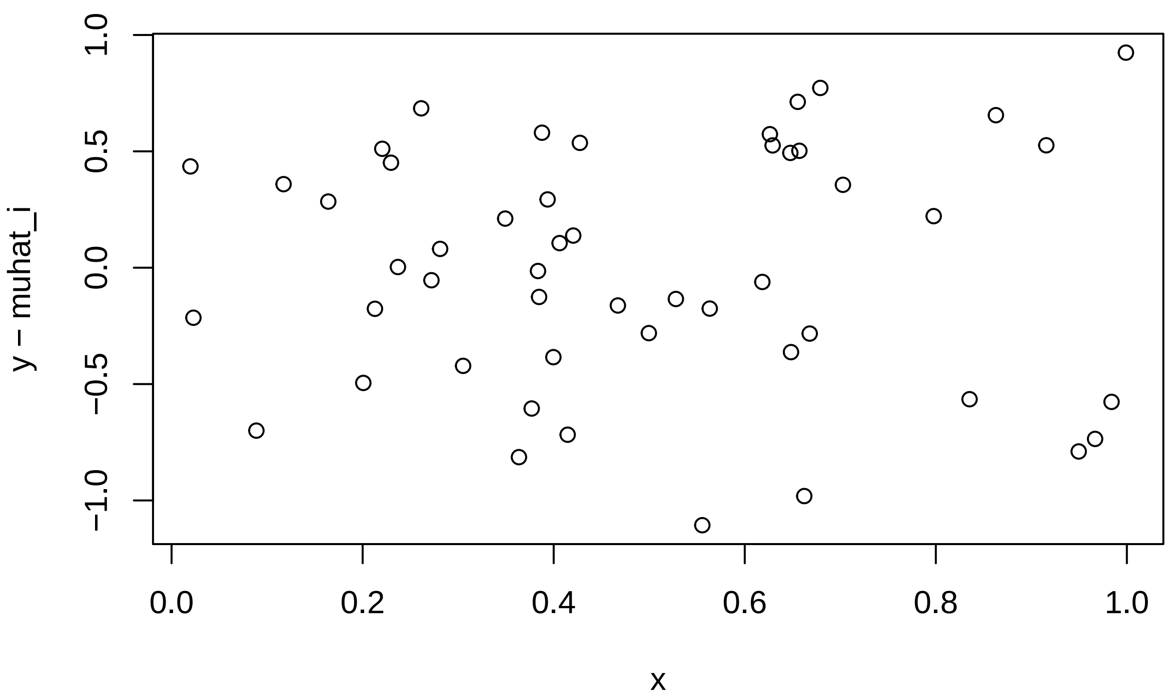
```r
#Now we are goint to plot. But first we need these values.
eta1fit=X1%*%beta1
eta2fit=X2%*%beta2

sigmahat_i=1/eta2fit
muhat_i=X1%*%beta1*sigmahat_i

#Plot the data together with the estimated expectation according to the normal GLM model and
#according to the standard linear model.
plot(x,y); lines(x,muhat_i,col="blue");
lines(x,fitted.values(fit),col="red")
legend("top", inset = 0.01, legend=c("data","standard LM","normal GLM"),col=c("black","blue","red"),lty=
```




```r
#Plot the residuals (using GLM) with plus and minus the estimated standard error at each value of x.
plot(x,y-muhat_i);
lines(x,y-muhat_i,col="black" );
lines(x,y-muhat_i+sqrt(sigmahat_i),col="blue" );
lines(x,y-muhat_i-sqrt(sigmahat_i),col="purple");
legend("top", inset = 0.01, legend=c("data","plus","minus"),col=c("black","blue","purple"),lty=1:1,cex=(
```




```r
plot(x,y-muhat_i);
```




```r
#part d
#Test whether the quadratic term in eta1 is significant, and test whether the quadratic term in eta2 is
#significant, both at the 5% level. Furthermore, test whether the normal GLM is a better model for this
#data than the standard normal model.

fisher_info = D2lik(beta1,beta2,y,X1,X2)
fisher_info
```

```
##               (Intercept)          x         x2 (Intercept)           x
```

```
## (Intercept)  -15.570040 -8.009300 -5.883572   111.01437   60.50685
## x             -8.009300 -5.883572 -4.881191    60.50685   49.78855
## x2            -5.883572 -4.881191 -4.303586    49.78855   44.72373
## (Intercept)  111.014368 60.506850 49.788551  -962.66335 -556.72988
## x             60.506850 49.788551 44.723725  -556.72988 -498.36122
## x2            49.788551 44.723725 41.647848  -498.36122 -470.79876
##                      x2
## (Intercept)   49.78855
## x             44.72373
## x2            41.64785
## (Intercept) -498.36122
## x           -470.79876
## x2          -452.76765
```

```
diagonal = diag(solve(-fisher_info))
diagonal
```

```
##  (Intercept)            x           x2  (Intercept)            x
##  10.54007872 198.26697935 179.22093630   0.07047266  14.49932800
##           x2
##  14.01701907
```

```
z1 = beta1[3]/sqrt(diagonal[3]/50)
z1
```

```
##        x2
## -22.57702
```

```
pnorm(z1,mean=0,sd=1,lower.tail = TRUE)
```

```
##            x2
## 3.645092e-113
```

```
#this tells we reject null hypothesis.That is quadratic term is significant in eta1.
#
z2 = beta2[3]/sqrt(diagonal[3]/50)
z2
```

```
##        x2
## -9.374746
```

```
pnorm(z2,mean=0,sd=1,lower.tail = TRUE)
```

```
##           x2
## 3.467128e-21
```

```
#this tells we reject null hypothesis.That is quadratic term is significant in eta2.

SS_glm = sum((y-muhat_i)^2)
SS_lm = sum(fit$residuals^2)
T = SS_lm - SS_glm
pchisq(T,2,lower.tail = FALSE)
```

```
## [1] 9.093541e-10
```

```
# from here we see that we reject the null hypothesis.
#That means GLM is better model comparing to standard normal model.
```

```r
#part e
###
#We wish to see whether the normal GLM model can be used effectively for prediction.
#For this, we repeat the following experiment many times: simulate new yi's using
#the model fitted in (c). Then fit the GLM model, and predict the value of y at
#x = 0.75. Also determine the standard deviation at x = 0.75. Do the same prediction,
#but now using the standard linear model. Compare both predictions with the (known!)
#true value, given by the model you simulate from. Determine which prediction is
#best. Also,think of a way to assess the estimation of the standard deviation at
#x = 0.75. If you feel up to it, you could repeat this for different vaues of x!
###

#first we calculate what is true mean and sd for the object at x=0.75.
eta1_true=c(1.0,0.75,0.75^2)%*%beta1
eta2_true=c(1.0,0.75,0.75^2)%*%beta2
mean_true=eta1_true/eta2_true
sigma_true=1/eta2_true


#this for the simulation
eta_one = X1 %*% beta1
eta_two = X2 %*% beta2
mu_i=eta_one/eta_two
sigma_i=1/eta_two
y_sim=rnorm(50,mean=mu_i,sd=sigma_i)


#we see what is log-likelihood, gradient and information for new simulated y's
D1_e = Dlik((beta1),(beta2),y_sim,X1,X2)
D2_e = D2lik(beta1,beta2,y_sim,X1,X2)
#L = lik(beta1,beta2,y_sim,X1,X2)

err=1
h_m = -(solve(D2_e))%*% (D1_e)
N=500
y_sims_glm = vector("numeric",N)
sigma_glm = vector("numeric",N)
y_sims_lm = vector("numeric",N)
sigma_lm = vector("numeric",N)
error1=0
error2=0
error3=0
error4=0
for(i in 1:N){
  y_sim=rnorm(50,mean=mu_i,sd=sigma_i)
  D1_e = Dlik((beta1),(beta2),y_sim,X1,X2)
  D2_e = D2lik(beta1,beta2,y_sim,X1,X2)
  #L = lik(beta1,beta2,y_sim,X1,X2)
  err=1
  h_m = -(solve(D2_e))%*% (D2_e)
  while(err > 10^-20){
    D1_e = Dlik((beta1),(beta2),y_sim,X1,X2)
    D2_e= D2lik(beta1,beta2,y_sim,X2,X2)
```

```r
    h_m = -(solve(D2_e))%*%(D1_e)
    while(sum(X2%*%(beta2+h_m[4:6])>0)<length(x)){
      h_m = (h_m)/2
    }
    while(lik(beta1+h_m[1:3],beta2+h_m[4:6],y_sim,X1,X2) -
          lik(beta1,beta2,y_sim,X1,X2)<0){
      h_m = (h_m)/2
    }
    err = abs(lik(beta1+h_m[1:3],beta2+h_m[4:6],y_sim,X1,X2) -
                lik(beta1,beta2,y_sim,X1,X2))
    beta1 = beta1+h_m[1:3]
    beta2 = beta2+h_m[4:6]
  }
y_sims_glm[i]=(c(1.0,0.75,0.75^2)%*%beta1)/(c(1.0,0.75,0.75^2)%*%beta2)
y_sims_lm[i]=c(1.0,0.75,0.75^2)%*%lm(y_sim~x+x2)$coefficients
sigma_glm[i]=(c(1.0,0.75,0.75^2))%*%(1/beta2)
sigma_lm[i]=1/sighatsq
error1=error1+sum(y_sims_glm[i]-mean_true)
error2=error2+sum(y_sims_lm[i]-mean_true)
error3=error3+sum(sigma_glm[i]-sigma_true)
error4=error4+sum(sigma_lm[i]-sigma_true)

}
average_error1=error1/N #mean for glm
average_error1
```

```
## [1] -0.001254882
```

```r
mean_true_vector=rep(mean_true,N)
mean(y_sims_glm-mean_true_vector) #this is same as average_error1
```

```
## [1] -0.001254882
```

```r
average_error2=error2/N #mean for lm
average_error2
```

```
## [1] 0.6904762
```

```r
mean(y_sims_lm-mean_true_vector)  #this is same as average_error2
```

```
## [1] 0.6904762
```

```r
average_error3=error3/N #SD for glm
average_error3
```

```
## [1] 0.1518339
```

```r
average_error4=error4/N #SD for lm
average_error4
```

```
## [1] 0.6361294
```

```r
#this tells us actually glm model is better than standard normal model,
#which makes sense because we started with glm.
```