HW1

Michael Cho

January 19, 2017

All output and plots are in the appendix towards the end of this pdf. Problems are referenced by question number and part letter i.g. 2.a 2.b.

# Problem 1 -

Please refer to proof attached in the appendix.

# Problem 2 -

## a.

is 29.62201 and is 13.44881

## b.

From the nlm function is 28.13688 and is 12.57428, and from the nls function is 28.13705 and is 12.57445.

# Problem 3 -

## a.

Please refer to appendix to see how the information and covariance matrix is calculated. The standard error for is 0.7418084 and the standard error for is 0.7795476.

## b.

Please refer to appendix to see how the covariance matrix from nls is calculated.The standard error for is 0.7279790 and the standard error for is 0.7630534. As we can see the results from part b do not agree with part a.

## c.

From nlm, the 95% conf interval for is [26.68296,29.59080] and for is [11.04640, 14.10217]. From nlm, the 95% conf interval for is [26.71024, 29.56386] and for is [11.07890, 14.07001]. The nlm confidence intervals is wider than nls confidence intervals as a result of having higher standard errors.

# Problem 4 -

## a.

Please refer to appendix for bootstrap histograms. The bootstrap standard errors are 0.7179607 for and 0.7420604 for .

## b.

The crude 95% confidence intervals is [26.83, 29.64] for and [11.18, 14.09] for .

## c.

The reflected 95% confidence intervals is [27.01, 29.88] for and [11.15, 14.11] for .

## c.

The confidence intervals from part c and b disagree. When comparing the confidence intervals to the histogram, it seems the reflected confidence intervals more accurately capture the skew aross the boostrap samples when comparing to the normal approximation which is not a perfect representation of the sample distributions themselves according to the normal plots.

# Problem 5 -

For X*=27, the 95% confidence interval for Y* is [18.79422, 19.59919] and the 95% prediction interval is [18.10358, 20.28983]. The prediction interval would be more appropriate here because we are trying to get the future response and the prediction interval contains a component that allows for an error component not captured by the model.

# Problem 6 -

We first get the initial guesses for paramater values from running a linear model for = + \* + and then plug this into the nls function. We get an AIC of 1.851093 from Problem 2b and an AIC of 3.05837 for the new model. The original model seems to be better according to AIC.

# Problem 7 -

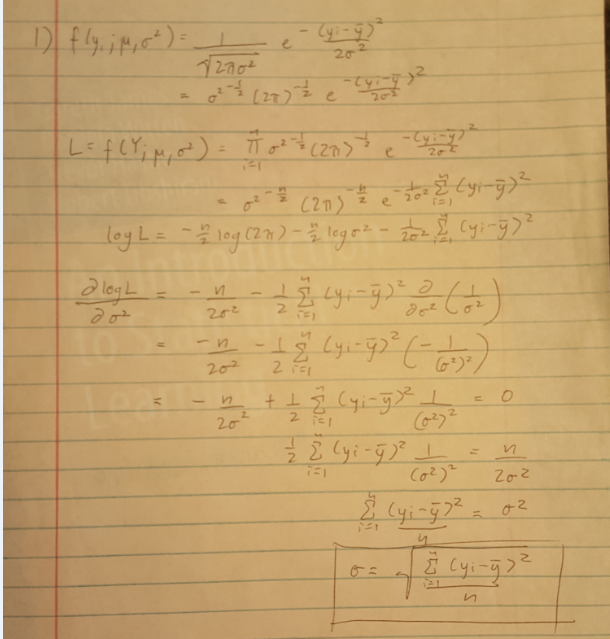
The n fold cross validation only requires 1 replicate since one sample is predicted after fitting all other observations. The n fold cross validation shows that the problem 2b model has 0.29 MSE, 0.99 value and 1.11 MSE, 0.97 value for the square root model.

# Problem 8 -

Please check appendix for residual vs x plots. The original model seems more appropriate as the residuals seem more random around 0 than the square root model which seems to have a more distinct residual pattern. Randomness is required for residuals for the expected value to be 0. The residual plots concur with the conclusions from problem 6 and 7.

# Appendix

## 1



## 2.a

#install.packages("xlsx")  
library(xlsx)  
data<-read.xlsx("HW1\_data.xls", sheetName = "Prob 2")  
fit<-lm(I(1/Y)~I(1/X), data=data)  
summary(fit)

##   
## Call:  
## lm(formula = I(1/Y) ~ I(1/X), data = data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.056684 -0.004123 0.000694 0.002766 0.063565   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.033759 0.006684 5.051 0.000118 \*\*\*  
## I(1/X) 0.454014 0.020061 22.632 1.41e-13 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.02175 on 16 degrees of freedom  
## Multiple R-squared: 0.9697, Adjusted R-squared: 0.9678   
## F-statistic: 512.2 on 1 and 16 DF, p-value: 1.411e-13

gamma0<-1/coef(fit)[1]  
gamma0

## (Intercept)   
## 29.62201

gamma1<-coef(fit)[2]/coef(fit)[1]  
gamma1

## I(1/X)   
## 13.44881

## 2.b

#nlm  
fn <- function(p) {yhat<-p[1]\*data$X/(p[2]+data$X); sum((data$Y-yhat)^2)}   
out<-nlm(fn,p=c(gamma0,gamma1),hessian=TRUE)  
out$estimate

## [1] 28.13688 12.57428

# nls  
fn2 <- function(x1,p) p[1]\*x1/(p[2]+x1)  
out2<-nls(Y~fn2(X, p),data=data,start=list(p=c(gamma0,gamma1)),trace=TRUE)

## 6.587434 : 29.62201 13.44881  
## 4.303552 : 28.14230 12.59805  
## 4.302271 : 28.13786 12.57534  
## 4.302271 : 28.13708 12.57449  
## 4.302271 : 28.13705 12.57445

coef(summary(out2))[,1]

## p1 p2   
## 28.13705 12.57445

## 3.a

theta<-out$estimate #parameter estimates  
MSE<-out$minimum/(length(data$Y) - length(theta)) #estimate of the error variance  
InfoMat<-out$hessian/2/MSE   
CovTheta<-solve(InfoMat)   
SE<-sqrt(diag(CovTheta))  
InfoMat # information matrix

## [,1] [,2]  
## [1,] 15.37455 -13.73842  
## [2,] -13.73842 13.92197

CovTheta # covariance matrix

## [,1] [,2]  
## [1,] 0.5502797 0.5430248  
## [2,] 0.5430248 0.6076944

SE # standard errors

## [1] 0.7418084 0.7795476

## 3.b

# nls covariance matrix  
vcov(out2)

## p1 p2  
## p1 0.5299535 0.5202828  
## p2 0.5202828 0.5822505

SEnls<-sqrt(diag(vcov(out2)))  
SEnls

## p1 p2   
## 0.7279790 0.7630534

## 3.c

# nlm 95% conf interval  
c(theta[1]-qnorm(0.975)\*SE[1], theta[1]+qnorm(0.975)\*SE[1])

## [1] 26.68296 29.59080

c(theta[2]-qnorm(0.975)\*SE[2], theta[2]+qnorm(0.975)\*SE[2])

## [1] 11.04640 14.10217

# nls 95% conf interval  
confint.default(out2)

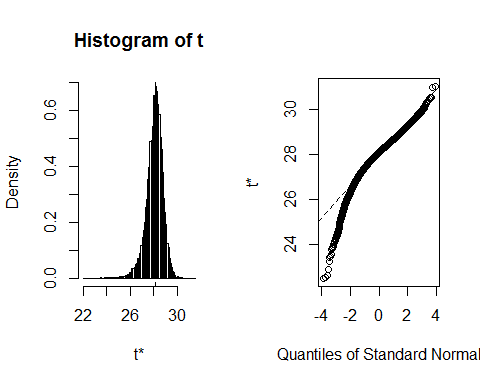
## 2.5 % 97.5 %  
## p1 26.71024 29.56386  
## p2 11.07890 14.07001

## 4.a

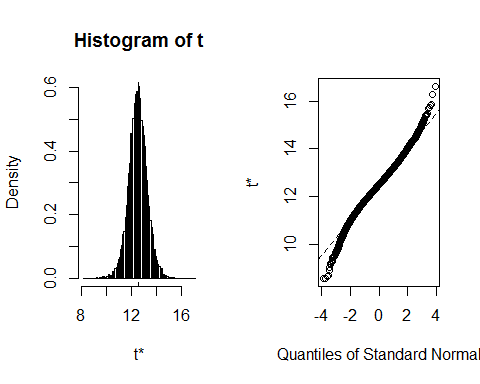
library(boot)  
datafit<-function(Z, i, theta0){  
 Zboot<-Z[i,]  
 # in order it appears in the file for Zboot  
 y<-Zboot[[1]]; x1<-Zboot[[2]]  
 fn3 <- function(p) {yhat<-p[1]\*x1/(p[2]+x1); sum((y-yhat)^2)}   
 out<-nlm(fn3,p=theta0)  
 theta<-out$estimate  
}  
databoot<-boot(data, datafit, R=20000, theta0=c(gamma0, gamma1))  
CovTheta<-cov(databoot$t)  
SE<-sqrt(diag(CovTheta))  
SE

## [1] 0.7184703 0.7437743

# bootstrap histogram for gamma0  
plot(databoot,index=1)



# boostrap histogram for gamma1  
plot(databoot,index=2)



## 4.b

boot.ci(databoot,conf=.95,index=1,type="norm")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
## Based on 20000 bootstrap replicates  
##   
## CALL :   
## boot.ci(boot.out = databoot, conf = 0.95, type = "norm", index = 1)  
##   
## Intervals :   
## Level Normal   
## 95% (26.82, 29.63 )   
## Calculations and Intervals on Original Scale

boot.ci(databoot,conf=.95,index=2,type="norm")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
## Based on 20000 bootstrap replicates  
##   
## CALL :   
## boot.ci(boot.out = databoot, conf = 0.95, type = "norm", index = 2)  
##   
## Intervals :   
## Level Normal   
## 95% (11.18, 14.09 )   
## Calculations and Intervals on Original Scale

## 4.c

boot.ci(databoot,conf=.95,index=1,type="basic")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
## Based on 20000 bootstrap replicates  
##   
## CALL :   
## boot.ci(boot.out = databoot, conf = 0.95, type = "basic", index = 1)  
##   
## Intervals :   
## Level Basic   
## 95% (27.01, 29.87 )   
## Calculations and Intervals on Original Scale

boot.ci(databoot,conf=.95,index=2,type="basic")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
## Based on 20000 bootstrap replicates  
##   
## CALL :   
## boot.ci(boot.out = databoot, conf = 0.95, type = "basic", index = 2)  
##   
## Intervals :   
## Level Basic   
## 95% (11.14, 14.12 )   
## Calculations and Intervals on Original Scale

## 5

datafit2<-function(Z, i, theta0, x\_pred){  
 Zboot<-Z[i,]  
 # in order it appears in the file for Zboot  
 y<-Zboot[[1]]; x1<-Zboot[[2]]  
 fn3 <- function(p) {yhat<-p[1]\*x1/(p[2]+x1); sum((y-yhat)^2)}   
 out<-nlm(fn3,p=theta0)  
 theta<-out$estimate  
 y\_pred<- theta[1]\*x\_pred/(theta[2]+x\_pred)} #predicted response  
databoot2<-boot(data, datafit2, R=20000, theta0=c(gamma0, gamma1), x\_pred=27)  
  
MSE<-out$minimum/(length(data$Y) - length(theta))   
VarYhat<-var(databoot2$t); VarYhat

## [,1]  
## [1,] 0.04360575

SEg<-sqrt(VarYhat); SEg

## [,1]  
## [1,] 0.2088199

SEy<-sqrt(VarYhat + MSE); SEy

## [,1]  
## [1,] 0.5590149

# conf int  
c(databoot2$t0-qnorm(.975)\*SEg, databoot2$t0+qnorm(.975)\*SEg)

## [1] 18.78743 19.60598

# pred int  
c(databoot2$t0-qnorm(.975)\*SEy, databoot2$t0+qnorm(.975)\*SEy)

## [1] 18.10106 20.29235

## 6

# AIC for problem 2b  
AIC2b<- -2\*as.numeric(logLik(out2))/18+2\*4/18  
AIC2b

## [1] 1.851093

# get initial guesses from linear model of new model  
fit6<-lm(I(Y)~I(sqrt(X)), data=data)  
summary(fit6)

##   
## Call:  
## lm(formula = I(Y) ~ I(sqrt(X)), data = data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.7997 -0.5451 0.2085 0.4413 1.6771   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.4566 0.5154 -0.886 0.389   
## I(sqrt(X)) 3.7720 0.1401 26.918 9.41e-15 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.9483 on 16 degrees of freedom  
## Multiple R-squared: 0.9784, Adjusted R-squared: 0.977   
## F-statistic: 724.6 on 1 and 16 DF, p-value: 9.414e-15

coef(fit6)

## (Intercept) I(sqrt(X))   
## -0.4565841 3.7720135

fn4 <- function(x1,p) p[1]+p[2]\*sqrt(x1)  
out4<-nls(Y~fn4(X,p),data=data, start=list(p=c(coef(fit6)[1],coef(fit6)[2])),trace=TRUE)

## 14.38837 : -0.4565841 3.7720135

AIC6<- -2\*as.numeric(logLik(out4))/18+2\*4/18  
AIC6

## [1] 3.05837

## 7

CVInd <- function(n,K) { #n is sample size; K is number of parts; returns K-length list of indices for each part  
 m<-floor(n/K) #approximate size of each part floor(18/18)=1  
 r<-n-m\*K # 18-1\*18=0  
 I<-sample(n,n) #random reordering of the indices   
 Ind<-list() #will be list of indices for all K parts  
 length(Ind)<-K # 18 lists  
 for (k in 1:K) {  
 if (k <= r) kpart <- ((m+1)\*(k-1)+1):((m+1)\*k) # not applicable here since r is 0  
 else kpart<-((m+1)\*r+m\*(k-r-1)+1):((m+1)\*r+m\*(k-r))   
 Ind[[k]] <- I[kpart] #indices for kth part of data  
 }  
 Ind  
}  
  
Nrep<-20 #number of replicates of CV  
n.models = 2 #number of different models to fit and compare  
fn2 <- function(x1,p) p[1]\*x1/(p[2]+x1)  
fn4 <- function(x1,p) p[1]+p[2]\*sqrt(x1)  
n<-nrow(data)  
K<-n #K-fold CV on each replicate, using n fold here  
yhat=matrix(0,n,n.models)  
MSE<-matrix(0,Nrep,n.models)  
  
for (j in 1:Nrep) {  
 Ind<-CVInd(n,K)  
 for (k in 1:K) {  
 out<-nls(Y~fn2(X,p),data=data[-Ind[[k]],],start=list(p=c(gamma0,gamma1)))  
 yhat[Ind[[k]],1]<-as.numeric(predict(out,data[Ind[[k]],]))  
 out<-nls(Y~fn4(X,p),data=data[-Ind[[k]],],start=list(p=c(coef(fit6)[1],coef(fit6)[2])))  
 yhat[Ind[[k]],2]<-as.numeric(predict(out,data[Ind[[k]],]))  
 } #end of k loop  
 MSE[j,]=apply(yhat,2,function(indvyhat) sum((data$Y-indvyhat)^2))/n  
} #end of j loop  
  
MSEAve<- apply(MSE,2,mean); MSEAve #averaged mean square CV error

## [1] 0.2943015 1.1106553

MSEsd <- apply(MSE,2,sd); MSEsd #SD of mean square CV error

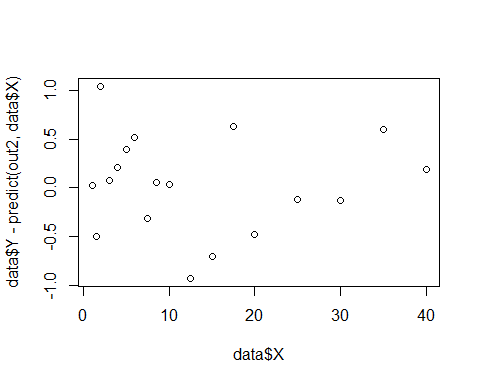
## [1] 0 0

r2<-1-MSEAve/var(data$Y); r2 #CV r^2

## [1] 0.9924874 0.9716484

## 7

# original model  
plot(data$X, data$Y-predict(out2,data$X))



# square root model  
plot(data$X, data$Y-predict(out4,data$X))

