TMA150/MSG400 Assigment 2

Martin Hansson

2020-09-22

1 Introduction

This report covers Assigment 2 in the course TMA150/MSG400. The assignment consists of 2 parts. The first part, *Model-choice*, is solved in R and the second part, *Bootstrap*, is solved in Matlab and R.

2 Exercise 1 - Simulated prediction interval

2.1 Problem

The *cars* data set(recorded in 1920s) contains data of speed(mph) of 50 cars and the distance (ft) taken to stop the car, given the speed. Do linear regression of the data to obtain $\beta 0$, $\beta 1$ and σ. Using these parameters, make new predictions of distance given speed $\in$ [5, 10, 15, 20, 25]. Do this 5000 times for each speed and produce a scatter plot. Also compute a 95% prediction interval for each speed and connect the 5 upper and 5 lower bounds in the plot.

2.2 Theory and implementation

The prediction interval can be computed either with an exact formula or by simulation. In this case, it is computed by simulation for 5000x5 speed values. Then the 2.5% and 97.5% quantiles will represent the lower and upper bound for the 95% prediction interval. The R-function *quantile()* is used to obtain the prediction intervals.

2.3 Results and discussion

The prediction intervals are shown in Table 1 and the scatter plot together with the upper and lower bound are shown in Figure 1.

Table 1: 95% Prediction intervals

Speed	Lower bound	Upper bound
5	-28.0	32.6
10	-8.31	51.8
15	10.7	72.0
20	29.4	90.8
25	50.5	110.2

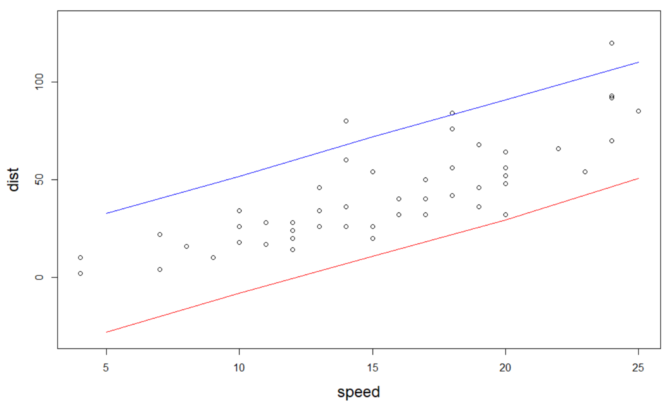

Figure 1: Speed(mph) vs Predicted distance(ft) including upper and lower bound of 95% prediction interval.

Figure 1 shows that 3 out of 50 (or 6%) observations is outside the prediction interval which is in line with expectations. However all 3 observations is above the interval which is a bit surprising. Another observation is that the lower bound is negative for lower speeds which of course is not realistic but a consequence of a linear regression model where the standard deviation is constant regardless of speed. In reality I would guess the standard deviation would increase with speed.

3 Exercise 2 - Polynomial regression

3.1 Problem

Data from United States, recorded between years 2003 and 2009, shows how *wage* for males varies with *age*. The sample size is N=3000. The task of this exercise is described below.

(i) Perform polynomial regression with respect to wage(y) vs age(x) of order p = 1-10. The regression models shall be based on 70% of random observations from the data set and the rest will be used to test the regression models. Compute pMSE and plot it vs p. Then identify the "best" model with respect to model complexity (low is better) and performance of the model, where pMSE is determining the performance.

(ii) Depending on which data points that where sampled to be used for *train* and *test* respectively, the conclusion from (i) could vary. In order to avoid this, do the procedure in (i) 50 times for each p and plot all *pMSE vs p* in a single figure. Again, determine the best p.

(iii) Use the p found in (ii) and plot all data points, the model fit based on p and the exact prediction interval at confidence 95%. Do the same thing for 2 other suboptimal p. State the conclusions.

3.2 Theory and implementation

When performing linear regression, usually R^2 index is used to measure how well the regression model fits the data that the regression are based upon. R^2 can be computed with the following formula:

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

When linear regression is performed, R^2 is minimized. Hence, if we then have a linear regression model of higher order p, it is easy to realize that R^2 becomes equal or less (better) as p increases. For example, a polynomial of degree p=*3* is a special case of a polynomial of degree p=*6*. It is then possible to get a better fit with increasing p. But this only holds for the data that was used fit the model. Now, it is more important to have a good fit for future predictions, i.e. known x and unknown y. This is measured by pMSE (prediction mean-squared-error) which is computed with the following formula.

$$pMSE(p) = \frac{1}{m}\sum_{i=1}^{m}(y_i^{new} - \hat{y}_i(p))^2$$

Increasing p could potentially decrease *pMSE* but not necessarily. In addition it is desirable to have as simple model as possible. So basically if the model does not significantly gain performance with respect to pMSE, the order of the polynomial regression should be kept as low as possible. This principle is called Occam's razor.

In order to solve the tasks in this section the data are split up in 2 subsets. 70% of randomly selected data points from *wage/age* are used to fit the polynomial models and the rest are used to test the model, i.e. compute pMSE (which is done by formlula above). The following main R-function are used.

- **sample.int** - to select train data set
- **lm** - to perform linear regression
- **quantile** - to find prediction intervals

3.3 Results and discussion

(i) Table 2 shows the resulting values for pMSE and Figure 2 shows the corresponding plot. Using Occam's razor, it seems obvious that p=*2* is the best option. For higher order it does not improve performance significantly.

Table 2: pMSE

p	1	$\boxed{2}$	3	4	5	6	7	8	9	10
pMSE	1586	1521	1521	1517	1517	1515	1513	1513	1510	1510

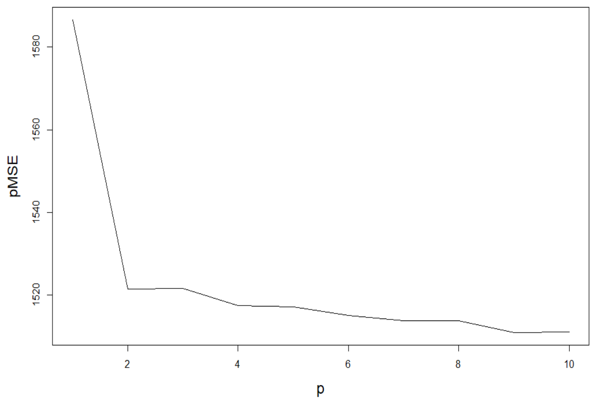

Figure 2: E2-1 pMSE (prediction mean-squared-error) vs polynomial order p

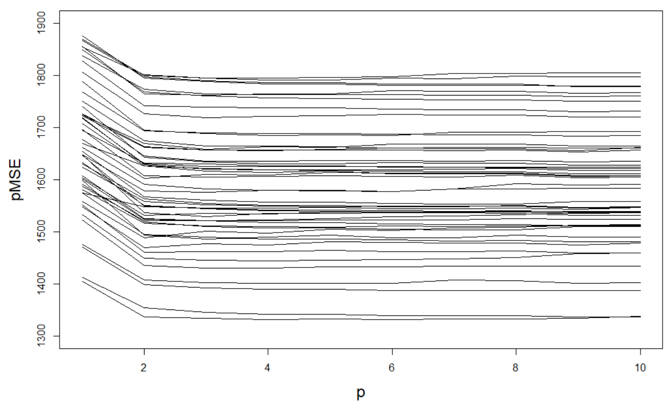

Figure 3: E2-2 pMSE vs polynomial order p for 50 different data sets

(ii) In Figure 3 *pMSE* vs *p* is shown for all 50 simulations. With the same motivation as in previous exercise, *p=2* is the best choice. It can also be seen that the pMSE is dependent within a data set, i.e. a high pMSE(2) tends to give a high pMSE(3) etc. I would guess that those with low pMSE-values (good performance) have better spread of various data points.

(iii) Figure 4 - Figure 6 shows the data points together with the prediction interval for different p's. It shows that different p's give similar results *within the observation span*, hence it is another argument that *p=2* is the best alternative in order to reduce model complexity. It can also be observed that there are some wages far outside the prediction interval but since they are relatively few, the impact on the prediction interval and model fit will be small.

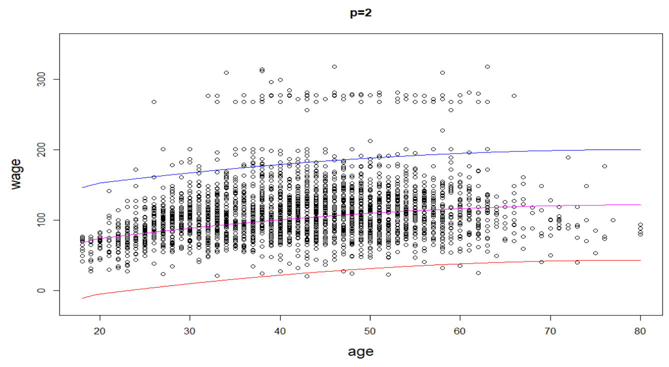

Figure 4: E2-3 Data, model fit and prediction interval for p=2

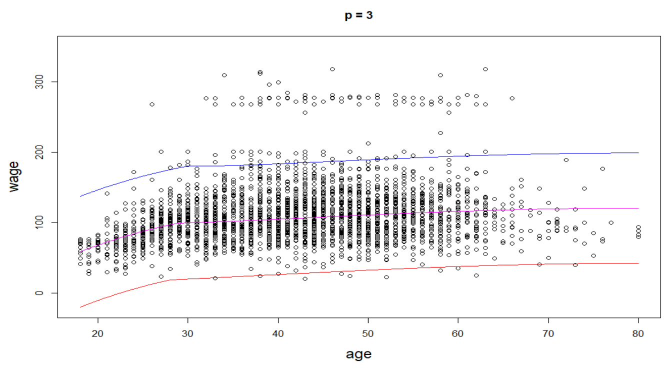

Figure 5: E2-3 Data, model fit and prediction interval for p=3

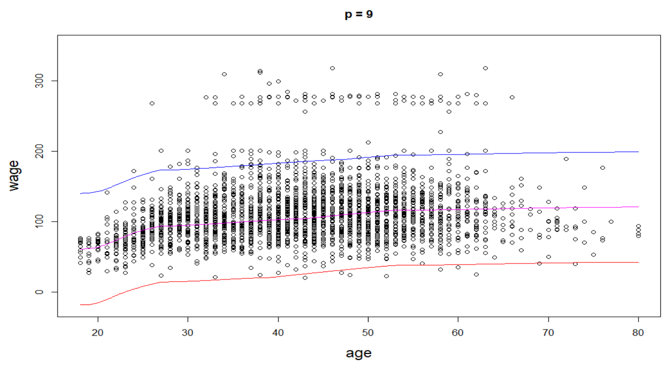

Figure 6: E2-3 Data, model fit and prediction interval for p=9

4 Exercise 3&4 - Bootstrap

4.1 Problem

The data file *atlantic.txt* contains the significant-wave-height recorded 14 times a month during several winter months in the north Atlantic. It was found that a good fit to the empirical distribution of the data is given by a Gumbel distribution. It has distribution function

$$F(x;\mu,\beta) = \exp\left(-\exp(-\frac{x-\mu}{\beta})\right), x \in \mathbf{R}$$

The provided Matlab function **est_gumbel.m** can be used to estimate the parameters from a data set.

a. Derive the inverse function $F^{-1}(u)$ used to generate random draws for a Gumbel distribution.

b. Use F^{-1} together with est_gumbel.m to simulate a sample of Gumbel data of size n (same number as atlantic) and then use *qqplot* to compare the distributions.

c. Provide parametric Bootstrap 95% confidence interval for the parameters, using the percentile method, based on B = 10,000 simulations.

d. The expected 100-year return value of the significant-wave-height gives the largest expected value during a 100-year period. The Tth return value is given b y$F^{-1}(1-1/T;\mu,\beta)$. Given 3x14 observations during a year, provide bootstrapped 95% confidence interval for a 100-year return value, using the percentile method.

e. Based on (d) and if the data was observed close to the coast, what barrier height would you recommend to protect a city by the coast?

f. Do a non-parameteric boostrapping, by sample n data points with replacement 10,000 times from the atlantic data set and then use the est_gumbel.m to estimate μ and β for each data set and finally compute a 95% confidence interval.

Exercise 4: Use the data set from exercise 2 and perform a simulation that produces 90% bootstrap confidence intervals for each regression parameter of the order p = 3 polynomial, based on B = 2000 bootstrap samples. Compare with the results returned from applying *confint* to the regression parameters.

4.2 Theory and implementation

The general idea about bootstrapping is to approximate the unknown distribution F by sampling data from it (with replacement) and use this approximation $\hat{F}$ instead of F. There are 2 main typs of boot-strapping; *parametric* and *non-parametric.* The *non-parametric* samples data directly from the observed data points while *parametric* samples data from a known distribution in which the parameters are determined from the observed data points. In practice (for both methods) you sample a data set of n, B times where n is the data size of the distribution you want to approximate. Then you estimate your parameter for each of the B data sets. These B parameter-values gives the approximation. The confidence intervals with percentile method is approximated using the quantile of interest.

Main R-function used in Exercise 4:

- **est_gumbel.m** - o estimate gumbal parameters from a data set
- **qqplot** - to compare if 2 data sets follows same distribution
- **quantile** - to find confidence interval with percentile method
- **rand.sample** - to sample data from distribution

Main R-function used in Exercise 4:

- **sample.int** - to sample data from distribution (with option replace=T)
- **quantile** - to find confidence interval for regression parameters with percentile method
- **confint** - to find confidence interval regression parameters

4.3 Results and discussion

a.

$$u = F(x) = \exp\left(-\exp(-\frac{x-\mu}{\beta})\right) \Rightarrow$$

$$-\ln u = \exp(-\frac{x-\mu}{\beta}) \Rightarrow$$

$$-\ln(-\ln u) = \frac{x-\mu}{\beta} \Rightarrow$$

$$-\beta\ln(-\ln u) = x-\mu \Rightarrow$$

$$\boxed{x = \mu - \beta\ln(-\ln u)}$$

b. Figure 7 shows a comparison of the quantiles between the observed data (X) and the simulated data (Y). It looks like the observed data follows a Gumbel distribution quite well but simulated data have slightly lower density for higher waves.

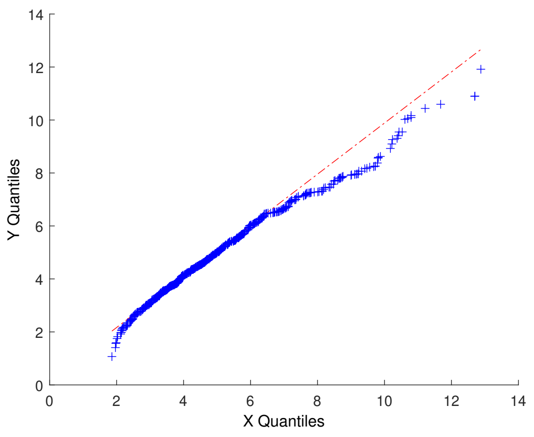

Figure 7: E3b qqplot, X = Observations, Y = Simulated from Gumbel

c. Table 3 shows the 95% confidence interval for the bootstrap when sampled from Gumbel distribution and and B=10,000.

Table 3: 95% confidence interval

	μ	β
Lower	4.0232	1.3902
Upper	4.2750	1.5800

d. The 95% confidence interval for the 100-year wave came out to be [**15.6961 m;17.3847 m**].

e. To be on the safe side for such a catastrophic event, I would use the upper limit for a 99% confidence interval which by simulation comes out to be **17.66 m**. So in the end, I would advice them to build a 18 m tall barrier.

f. Table 4 shows the 95% confidence interval for the bootstrap when sampled from observed data points and B=10,000. The upper and lower limits are very similar for the non-parametric and the parametric model used in c. It seems like we are ending up with the same results with both methods, given the same distribution, i.e. Gumbel distribution.

Table 4: 95% confidence interval

	μ	β
Lower	4.0242	1.3911
Upper	4.2737	1.5806

Exercise 4: In Table 5 we can see that the confident intervals are larger for the *bootstrap* compared to *confint*. Another observation is that the intervals are symmetric with *confint* but non-symmetric for *bootstrap*. *Confint* are assuming a normal distribution for the data set, which means that the intervals, per definition, becomes exactly symmetric. For *bootstrap*, no such assumption is made, which means that the shape of the distribution will be more like the data set. We can see from Figure 4 that the data has an upper tail (some high wages) which will be taken into account by *bootstrap*, hence non-symmetric.

Table 5: 90% with Boot Strap and confint()

	β_0	β_1	β_2	β_3
BS_Lower	-103.8	8.12	-0.221	0.000500
BS_Upper	-47.25	12.41	-0.120	0.00124
BS_median	-76.57	10.28	-0.170	0.000866
BS_symm	-75.55	10.26	-0.170	0.000869
confint_Lower	-111.7	7.55	-0.229	0.000405
confint_Upper	-38.74	12.83	-0.107	0.00129
confint_median	-75.24	10.19	-0.168	0.000849
confint_symm	-75.24	10.19	-0.168	0.000849

Appendix - code

Exercise 1 R-code

```
# load the "cars" dataset (provided within R program)
data(cars)
attach(cars)

#Assigment 2 - Exercise 1
x_mean<-mean(speed)
A<-(speed-x_mean)*dist
B<-(speed-x_mean)^2
beta1=sum(A)/sum(B)
beta0=mean(dist)-beta1*mean(speed)
y_est=beta0+beta1*speed #estimation with lm-model
s=sqrt(sum((dist-y_est)^2)/(length(speed)-2)) #std error

#prediction for new speeds
n=5000
speed_new=(rep(1,n))%*%t(seq(5,25,5)) #matrix with new speeds
set.seed(321)
err=matrix(rnorm(length(speed_new),0,s),nrow=n,ncol=5)
dist_pred=beta0+beta1*speed_new+err #new distances
lower=numeric(5)
upper=numeric(5)
for (i in 1:5) {
  lower[i]=quantile(dist_pred[,i],probs=0.025) #lower bound for PI
  upper[i]=quantile(dist_pred[,i],probs=0.975) #upper bound for PI
}

plot(speed,dist,ylim=c(-30,130),cex.lab = 1.5) #scatter plot
lines(speed_new[1,],lower,type="l",col='red') #plotting lower bound
lines(speed_new[1,],upper,type="l",col='blue') #plotting upper bound
```

Exercise 2+4 R-code

```
Wage<-read.table("wage.txt",header=TRUE)
y=Wage$wage
x=Wage$age
nx=length(x)

#Exercise 2(i)
##########################################################################
n_train=floor(0.7*nx) #n
n_test=nx-n_train #N-n
pMSE=numeric(10)
set.seed(321)
indeces <- sample.int(nx,n_train) #find ntrain random indeces from data set
x_tr=x[indeces] #vector with train x-values
x_test=x[-indeces] #vector with test x-values
y_tr=y[indeces] #vector with train y-values
y_test=y[-indeces] #vector with test y-values
for (i in 1:10) {
  m=lm(y_tr~poly(x_tr,i,raw=TRUE)) #poly regression of order i
  y_est=predict(m,newdata = data.frame(x_tr = x_test)) #model pred train x-
      values
  pMSE[i]=sum((y_est-y_test)^2)/n_test #pMSE for order i
}
plot(seq(1,10,1),pMSE,type="l",xlab='p',ylab='pMSE',cex.lab = 1.5)
```

```
#Exercise 2(ii)
##################################################################
n_att=50 #number of attemps
pMSE=matrix(nrow=10,ncol=n_att)
set.seed(321)
for (j in 1:n_att) {
  indeces <- sample.int(nx,n_train) #find ntrain random indeces from data
     set
  x_tr=x[indeces] #vector with train x-values
  x_test=x[-indeces] #vector with test x-values
  y_tr=y[indeces] #vector with train y-values
  y_test=y[-indeces] #vector with test y-values
  for (i in 1:10) {
    m=lm(y_tr~poly(x_tr,i,raw=TRUE)) #poly regression of order i
    y_est=predict(m,newdata = data.frame(x_tr = x_test)) #model pred train
       x-values
    pMSE[i,j]=sum((y_est-y_test)^2)/n_test #pMSE for order i
  }
}
plot(seq(1,10,1),pMSE[,1],type="l",ylim=c(1300,1900),xlab='p',ylab='pMSE',
   cex.lab = 1.5)
for (i in 2:50) {
  lines(seq(1,10,1),pMSE[,i],type="l")
}
p=seq(1,10,1)%*%t(rep(1,n_att)) #matrix with p-values
plot(p,pMSE)

#Exercise 2(iii) p=2
##################################################################
m2=lm(y~poly(x,2,raw=TRUE)) #poly regression of order 2
m3=lm(y~poly(x,3,raw=TRUE)) #poly regression of order 3
m9=lm(y~poly(x,9,raw=TRUE)) #poly regression of order 9
y2=predict(m2,newdata=data.frame(x=seq(min(x),max(x),.1)),interval="
    prediction")
y3=predict(m3,newdata=data.frame(x=seq(min(x),max(x),.1)),interval="
    prediction")
y9=predict(m9,newdata=data.frame(x=seq(min(x),max(x),.1)),interval="
    prediction")


plot(x,y,ylim=c(-20,350),xlab='age',ylab='wage',cex.lab = 1.5,main='p = 2')
lines(seq(min(x),max(x),.1),sort(y2[,1]),type="l",col = "magenta")
lines(seq(min(x),max(x),.1),sort(y2[,2]),type="l",col = "red")
lines(seq(min(x),max(x),.1),sort(y2[,3]),type="l",col = "blue")

plot(x,y,ylim=c(-20,350),xlab='age',ylab='wage',cex.lab = 1.5,main='p = 3')
lines(seq(min(x),max(x),.1),sort(y3[,1]),type="l",col = "magenta")
lines(seq(min(x),max(x),.1),sort(y3[,2]),type="l",col = "red")
lines(seq(min(x),max(x),.1),sort(y3[,3]),type="l",col = "blue")

plot(x,y,ylim=c(-20,350),xlab='age',ylab='wage',cex.lab = 1.5,main='p = 9')
lines(seq(min(x),max(x),.1),sort(y9[,1]),type="l",col = "magenta")
lines(seq(min(x),max(x),.1),sort(y9[,2]),type="l",col = "red")
lines(seq(min(x),max(x),.1),sort(y9[,3]),type="l",col = "blue")

#Exercise 4 - Bootstrapping beta0-beta3
##################################################################
```

```
B=2000
beta=matrix(,nrow=B,ncol=4)
set.seed(321)
for (i in 1:B) { #loop for bootstrapping
  index=sample.int(nx,nx,replace=T) #indeces for sampling with replacement
  xs=x[index]
  ys=y[index]
  m=lm(ys~poly(xs,3,raw=TRUE))
  beta[i,1]=m$coefficients[1] #Extraxt coefficients
  beta[i,2]=m$coefficients[2]
  beta[i,3]=m$coefficients[3]
  beta[i,4]=m$coefficients[4]
}
#90% confint for bootstrapping parameters
CI_boot=matrix(,nrow=3,ncol=4)
CI_boot[,1]=quantile(beta[,1],probs=c(0.5,0.05,0.95))
CI_boot[,2]=quantile(beta[,2],probs=c(0.5,0.05,0.95))
CI_boot[,3]=quantile(beta[,3],probs=c(0.5,0.05,0.95))
CI_boot[,4]=quantile(beta[,4],probs=c(0.5,0.05,0.95))
#Comparing with confint 90 %
m0=lm(y~poly(x,3,raw=TRUE))
ci=confint(m0,level=0.9)
```

Exercise 3 Matlab-code

```
%%Exercise 3b
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
wh0=load('atlantic.txt'); %read wave height from data file
n=length(wh0);
gamb_est0=est_gumbel(wh0); %get mu and beta from est_gumbel.m
beta0=gamb_est0(1); %beta of data set
mu0=gamb_est0(2); %mu of data set
u=rand(n,1); %n samples of between 0-1
wh1=mu0-beta0*log(-log(u)); %wave height from distribution
qqplot(wh0,wh1);

%%Exercise 3c
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rng(123)
B=10000;
beta=zeros(B,1);
mu=zeros(B,1);
for i=1:B
    u=rand(n,1);
    wh=mu0-beta0*log(-log(u));
    gamb_est=est_gumbel(wh);
    beta(i)=gamb_est(1);
    mu(i)=gamb_est(2);
end
CI_beta=[quantile(beta,0.025),quantile(beta,0.975)]
CI_mu=[quantile(mu,0.025),quantile(mu,0.975)]

%%Exercise 3d - 100 year wave. Use mu and beta from 3c
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
T=4200;
h100=zeros(B,1);
for i=1:B
    h100(i)=mu(i)-beta(i)*log(-log(1-1/T));
end
```

```
CI_h100=[quantile(h100,0.025),quantile(h100,0.975)] %95% conf interval for
    100 year wave

%%Exercise 3e - Check 99% CI
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
I_h100=[quantile(h100,0.005),quantile(h100,0.995)]

%%Exercise 3f - Non-parametric Gumbel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rng(123)
B=10000;
for i=1:B
   wh=randsample(wh0,n,true); %n sample from atlantic data with replacement
   gamb_est=est_gumbel(wh); %get mu and beta with est_gumbel.m
   beta(i)=gamb_est(1);
   mu(i)=gamb_est(2);
end
CI_beta=[quantile(beta,0.025),quantile(beta,0.975)]
CI_mu=[quantile(mu,0.025),quantile(mu,0.975)]
```