A2

Ko-Cheng Chang

May 2021

1 Introduction

Assignment 2 is separate into two part, in first part, we try to determine the best model within many available model. In second part, we try to estimate parameter with non-parametric and parametric bootstrapped.

2 Q1

2.1 Problem

In first exercise, we generate new predicted data at value of speed $\in [5, 10, 15, 20, 25]$ based on our regression model, then construct 95% of prediction intervals. After that, plot all data into one figure.

2.2 Theory and implementation

First, we first obtain parameter estimates from our car dataset. After that we generate 5000 prediction of new observation from our parameter and error term, $\epsilon_{new} \sim N(0, s^2)$, the s in error term is also obtain from the data. After that we can plot all 95% of prediction interval into one figure.

2.3 Results and discussion

In figure 1, we know the upper and lower bounder at 95% interval of prediction. In the figure, we notice that there are two spot is outside our interval, which is not surprising, since our interval is 95%, there should be some spot oudside our interval. What's surprising is that our lower bounder of distance is below 0 when speed are 10 and 5 which is not possible in reality. It shows that when we use tool, such as this, to exam our data, we cannot only consider on our result, we need to also consider the mean of the data and think the possible of our result.

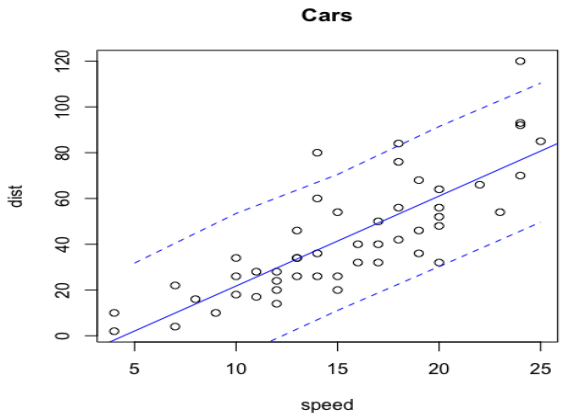

Figure 1: The 95% of prediction interval and regression line of cars data

3 Q2[i]

3.1 Problem

In this question, we try to determine the best polynomial regression in between many degree choices. After that we plot our result into on figure.

3.2 Theory and implementation

Since R^2 will always increase with more degree in polynomial regression. So we need to use pMSE to indicate what is the best model among all. The way to obtain pMSE is randomly sampling our data into two part, training and testing, we construct regression with training data and exam it's with testing data. The formula of pMSE $\frac{1}{m}\Sigma(y_i^{new} - \hat{y}_i(p))^2$, it shows how training model do when comparing with testing. In other word, it simulates how model do when we try to predict new data with model. If we have lower value of pMSE, it shows that the model have better ability to predict future data, hence, it's better model.

3.3 Results and discussion

From figure 2, we find out that pMSE is still decreasing when we increasing degree. However, it doesn't mean we should use degree of 10 as our best model. In figure, we notice that the decreasing level of increasing degree is not significant after 2. The benefit comes with more degree is not massive. According to Occam's principle, we should use 2 instead of 10 as our best model.

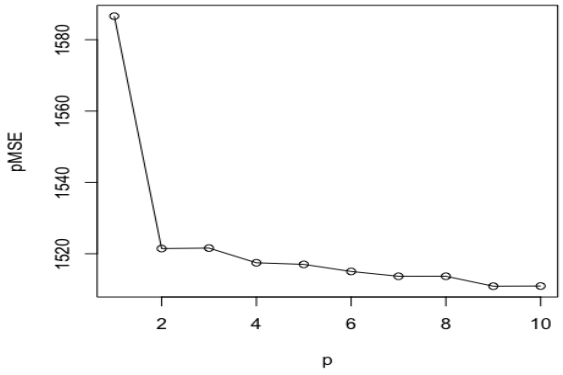

Figure 2: pMSE value from 1 to 10 degree

4 Q2[ii]

4.1 Problem

In above question, we only sample our data once. In this question, we run the our random sampling process 50 times. To determine whether our result in previous question robust or not.

4.2 Theory and implementation

We use for loop to run sampling process 50 time, and store each pMSE value into matrix. After that, we plot pMSE value into one figure.

4.3 Results and discussion

From figure 3, we knows that pMSE value of degree is similar to previous one. However, the different between degree of 2 to 10 is much less than previous one, which shows that the benefit when increasing degree is less than previous one. It shows that we should consider degree of 2 as our model should choose.

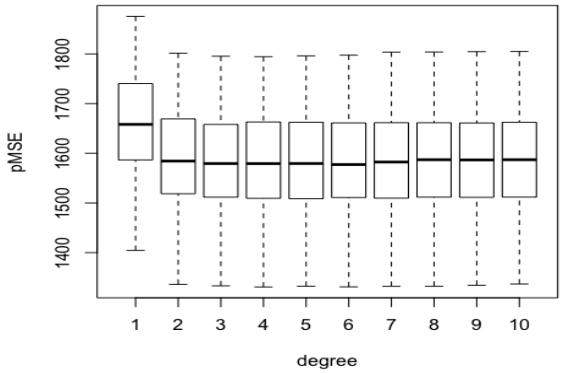

Figure 3: pMSE value from 1 to 10 degree when 50 times repeating

5 Q2[iii]

5.1 Problem

Similar to Q1, in this question, we pick some degree we believe it's the as good in[ii] and Produce a figure with data, fit based on degree and exact prediction interval at 95% of confidence level. After that we construct another two figure with two suboptimal degree.

5.2 Theory and implementation

Based on [ii], I believe that 2 is the best among all degree and 3 and 4 are suboptimal degree. First, I calculate exact prediction interval with R commend "predict", after that, we plot all data into one figure.

5.3 Results and discussion

From all figures, we find out that all regression line and interval is not significant different to others. However, the level of complex is different in our model. When we choose to use degree of 2 as our main model, it's much easier to calculate comparing with 4. Since choosing 4 as our best model doesn't come with massive benefit with it. We should use degree 2 as our model choice.

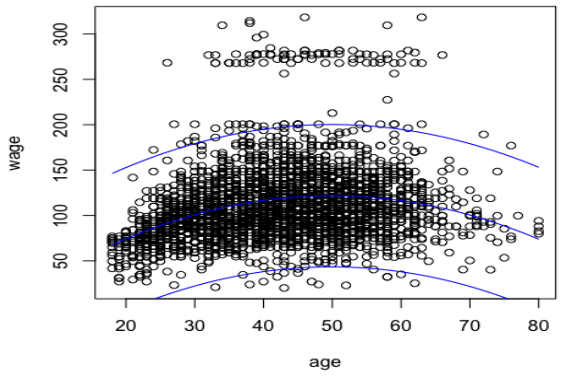

Figure 4: regression line and exact predict interval at 95% of confidence level when degree is 2

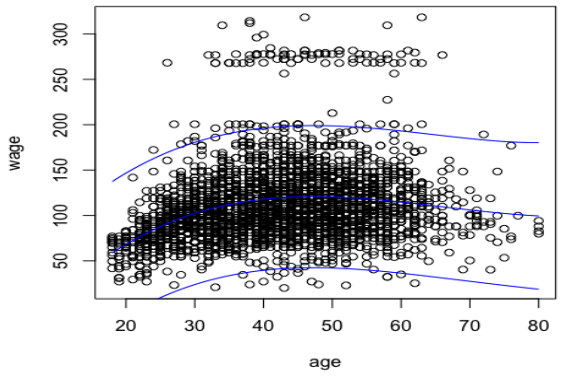

Figure 5: regression line and exact predict interval at 95% of confidence level when degree is 3

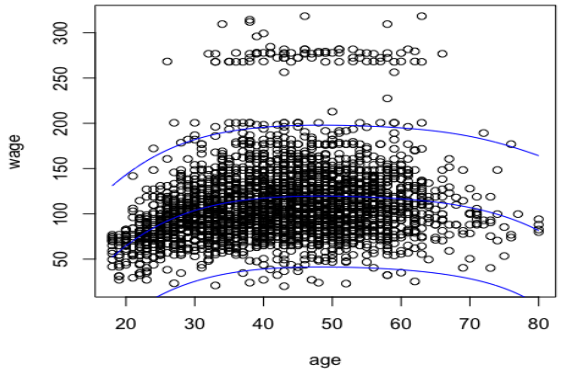

Figure 6: regression line and exact predict interval at 95% of confidence level when degree is 4

6 Q3[a]

6.1 Problem

Derive formula to generate random draws from Gumbel distribution with inversion cdf method.

6.2 Theory and implementation

If the inverse function of cdf is exist, then we can use uniform random variable, $U(0,1)$, to simulate the random samples from data. The idea behind this is that if $X \sim F$ and inverse function exit. We can find that $F^{-1}(U) \sim F$. We just need to sample $u \sim U(0,1)$ to simulate a draw $x^* \sim F$ where $x^* = F^{-1}(U)$, x^* is the draw we want. Therefore, we have $X = F^{-1}(U)$ and thus $U = F(X)$

6.3 Results and discussion

The cdf of Gumbel distribution is $\exp(-\exp(-\frac{x-\mu}{\beta}))$. Therefore, the equation is

$$F(X) = \exp(-\exp(-\frac{X-\mu}{\beta})) = U \tag{1}$$

Then we can have

$$-\exp(-\frac{X-\mu}{\beta}) = ln(U) \tag{2}$$

and

$$-\frac{X-\mu}{\beta} = ln(-ln(U)) \tag{3}$$

Thus

$$X = \mu - \beta ln(-ln(U)) \tag{4}$$

The equation 4 is the formula to generate random draws from Gumbel distribution.

7 Q3[b]

7.1 Problem

With formula in (a) and MLE method, we simulate the a sample of size n(same size as altantic) from atlantic data, and check distribution of simulated data and atlantic data their approximately agree.

7.2 Theory and implementation

First, we need to use MLE method to estimate parameter μ and β. After that, we put estimated parameter into our formula in (a) to obtain exact formula. With this formula, we can simulated sample draws from same distribution.

7.3 Results and discussion

In figure 7, we can find that the approximately agree between simulated and atlantic data is close. In other word, they are close to each other, and maybe draw from same distribution. Thus, we can use this data to simulated more random sample from the same distribution.

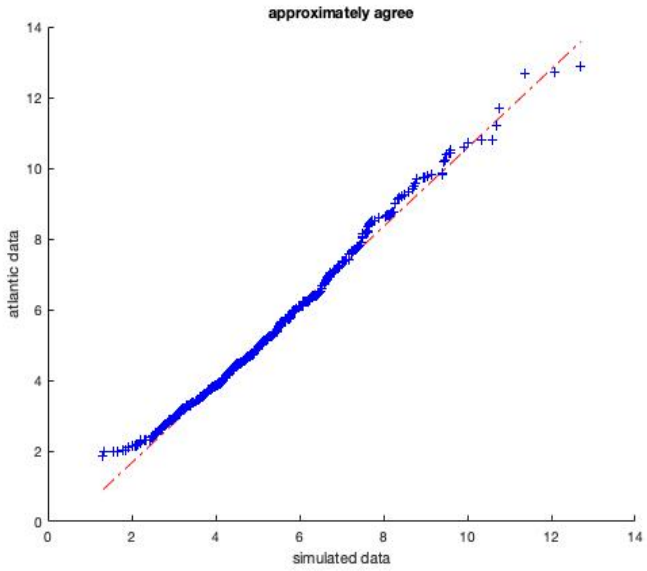

Figure 7: The approximately agree between simulated and atlantic data

8 Q3[c]

8.1 Problem

Calculate the 95% confidence intervals for the parameters with percentile method.

8.2 Theory and implementation

With method in (b) we can simulate the sample draws from same distribution. After repeat 10000 time, we can obtain 10000 estimated parameters. We can obtain confidence interval with 95% with estimated parameters we have, by calculated the percentile in our estimated parameters. This indicates the 95% confidence interval of our estimated parameter.

8.3 Results and discussion

The 95% confidence interval for β is $[1.3162, 1.3621]$ and for μ is $[3.92, 3.9863]$. In other word, we have high confidence that β and μ will located in this interval.

9 Q3[d]

9.1 Problem

We need to estimated the 95% confidence interval for expected 100-year return value of the significant-wave-height.

9.2 Theory and implementation

First, we can reuse the estimated parameters from (c). By these estimated parameters, we can obtain 10000 estimated expected 100-year return value. Therefore, we can obtain its 95% confidence interval with percentile method.

9.3 Results and discussion

The 95% confidence interval for 100-year return value is $[15.0147, 15.488]$. In other word, we have high confidence that expected 100-year return value will located in this interval.

10 Q3[e]

10.1 Problem

Based on (d), if we were to advise the city council that how tall at a minimum should barrier be in order to protect the city from the highest(expected) wave.

10.2 Theory and implementation

The question want to ask us what is the minimum tall in order to protect city from highest wave. If we obtain 100-year return value, the city will be safe as long as barrier is at least same tall as this value.

10.3 Results and discussion

From (d), we know that the 95% confidence interval for expected wave. We have highly confidence that the real value will inside this interval. Therefore, if we want to protect city from the wave, the minimum height of barrier in order to protect city should be at least same as upper bounder we get from (d). Thus, it's 15.4488

11 Q3[f]

11.1 Problem

Using non-parametric bootstrap method to estimate the parameter in (c).

11.2 Theory and implementation

Since we don't have the assumption about the distribution of the data. We can only re-sample the data we have, treating them as new sample to generate the confidence interval for our parameter.

11.3 Results and discussion

The 95% confidence interval for β is $[1.3208, 1.3616]$ and for μ is $[3.9306, 3.9856]$. We can find out that the interval is not much different than we obtain in (c). This shows that both non-parametric and parametric provide similar result. Therefore, if we find it's difficult to use parametric method, non-parametric is also great.

12 Q4

12.1 Problem

Try to estimated the confidence interval of parameters in linear regression with polynomial of order is 3, $p = 3$.

12.2 Theory and implementation

Due to the setup in question, I decide to use non-parametric method to estimate our parameter. I resample the data we have in Wage, by these resample data, I can obtain new estimated parameter. By repeating 2000 times, we can have list of estimated parameters. After that, I use percentile method. To obtain the 90% confidence interval of each parameter and compare them with result we obtain from confint.

12.3 Results and discussion

The confidence interval are

β_0 are $[-103.84584, -47.24865]$ (bootstrap) and $[-111.74419, -38.74364]$ (confint).
β_1 are $[8.118609, 12.407817]$ (bootstrap) and $[7.548809, 12.831174]$ (confint).
β_2 are $[-0.2206401, -0.1202371]$ (bootstrap) and $[-0.2286770, -0.1073802]$ (confint).
β_3 are $[0.0005003143, 0.0012375806]$ (bootstrap) and $[0.0004048014, 0.0012941030]$ (confint).

From above we know that the different between two method is not huge. Therefore, bootstrap is a good alternative method to obtain confidence interval when traditional method doexn't work.

Appendix - code

```
setwd("~/      /Stochastic data processing and simulation")
set.seed(321)
# Ex1

# predict the B
attach(cars)
m1 <- lm(dist~speed) # extract betahat
betahat_0 <- m1$coefficients[1]
betahat_1 <- m1$coefficients[2]

# obtain s
s <- summary(m1)$sigma

# compute predictions
B <- matrix(0,5000,5)
set.seed(321)
for (i in 1:5000){
  B[i,1] <- betahat_0 + betahat_1 * 5 + rnorm(1, 0, s)
  B[i,2] <- betahat_0 + betahat_1 * 10 + rnorm(1, 0, s)
  B[i,3] <- betahat_0 + betahat_1 * 15 + rnorm(1, 0, s)
  B[i,4] <- betahat_0 + betahat_1 * 20 + rnorm(1, 0, s)
  B[i,5] <- betahat_0 + betahat_1 * 25 + rnorm(1, 0, s)
}
B <- as.data.frame(B, col.names = names('speed_5','speed_10','speed_15','
    speed_20', 'speed_25'))
colnames(B) <- c('speed_5','speed_10','speed_15','speed_20', 'speed_25')

# compute 2.5 and 97.5 empirical quantiles for each set
speed_5_int <- quantile(B$speed_5, c(.025, .975))
speed_10_int <- quantile(B$speed_10, c(.025, .975))
speed_15_int <- quantile(B$speed_15, c(.025, .975))
speed_20_int <- quantile(B$speed_20, c(.025, .975))
speed_25_int <- quantile(B$speed_25, c(.025, .975))

#upper and lower bounder
low_bound <- c(speed_5_int[1], speed_10_int[1], speed_15_int[1],
    speed_20_int[1], speed_25_int[1])
up_bound <- c(speed_5_int[2], speed_10_int[2], speed_15_int[2],
    speed_20_int[2], speed_25_int[2])

#Produce the scatterplot of the data and adding regress line
plot(speed, dist, main="Cars")
abline(m1, col="blue")

# adding the upper and lower bounders
x = seq(5,25, by = 5)
lines(x, low_bound, col="blue", lty=2)
lines(x, up_bound, col="blue", lty=2)


# Ex2(i)
Wage<-read.table("Wage.txt",header=TRUE)
# select training and test sample
N <- 3000
n <- floor(0.7* N)
set.seed(321)
```

```
indeces <- sample.int(N, n)

# classify the training and test sample
training <- Wage[indeces,]
testing <- Wage[-indeces,]

# gen matrix for MSE
pMSE <- seq(0,0, length.out = 10)
 for (i in 1:10){
  m2 <-lm(wage~poly(age,i,raw=TRUE), data = training)
  y_new_hat <- predict(m2, newdata = testing)
  sum_testing <- sum((testing$wage - y_new_hat)^2)
  pMSE[i] <- sum_testing / (N - n)
 }

# plot pMSE
plot(1:10, pMSE, ylab = 'pMSE', xlab = 'p')
lines(1:10, pMSE)
View(pMSE)

# Ex2(ii)
pMSE <- matrix(0,50,10)
colnames(pMSE) <- c(1:10)
set.seed(321)
for (m in 1:50){
  indeces <- sample.int(N, n)
  training <- Wage[indeces,]
  testing <- Wage[-indeces,]
  for (i in 1:10){
    m2 <-lm(wage~poly(age,i,raw=TRUE), data = training)
    y_new_hat <- predict(m2, newdata = testing)
    sum_testing <- sum((testing$wage - y_new_hat)^2)
    pMSE[m,i] <- sum_testing / (N - n)
  }
}

# plot figure
temp <- as.data.frame(as.table(pMSE))
drops <- c("Var1")
temp <- temp[ , !(names(temp) %in% drops)]
plot(temp, xlab = 'degree', ylab = 'pMSE')


# Ex2(iii)
# construct model with optimal and suboptimal p
xstar <- seq(18,80, length.out = 1000) # x vector

# p = 2
m3 <-lm(wage~poly(age,2,raw=TRUE), data = Wage)
mypred_1 <- predict(m3, newdata = data.frame(age=xstar), interval = "
    prediction")
plot (Wage) # plot figure
lines(xstar, predict(m3, data.frame(age=xstar)), col="blue")
lines(xstar, mypred_1[,2], col="blue")
lines(xstar, mypred_1[,3], col="blue")

# p = 3
m4 <-lm(wage~poly(age,3,raw=TRUE), data = Wage)
```

```
mypred_2 <- predict(m4, newdata = data.frame(age=xstar), interval = "
    prediction")
plot (Wage) # plot figure
lines(xstar, predict(m4, data.frame(age=xstar)), col="blue")
lines(xstar, mypred_2[,2], col="blue")
lines(xstar, mypred_2[,3], col="blue")

# p = 4
m5 <-lm(wage~poly(age,4,raw=TRUE), data = Wage)
mypred_3 <- predict(m5, newdata = data.frame(age=xstar), interval = "
    prediction")
plot (Wage) # plot figure
lines(xstar, predict(m5, data.frame(age=xstar)), col="blue")
lines(xstar, mypred_3[,2], col="blue")
lines(xstar, mypred_3[,3], col="blue")

%Q3(a
out = est_gumbel(atlantic); %MLE to estimate parameter
beta = out(1);
mu = out(2);

%Q3(b
u = rand(582,1); % sample from uniform distribution
x = mu - beta*log(-log(u)); %simulated draws with inverse method
qqplot(x, atlantic) % compare the result
title('approximately agree')
xlabel('simulated data')
ylabel('atlantic data');

%Q3(c
B = 10000;  % the number of bootstrap simulations
boots_beta = zeros(B,1); % replace = 0 for sampling without replacement
boots_mu = zeros(B,1);
n = 582; %the number of data
rng(123);
for ii=1:B
    % simulate bootstrapped data
    u = rand(n,1);
    x = mu - beta*log(-log(u));
    out = est_gumbel(x);
    boots_beta(ii) = out(1);
    boots_mu(ii) = out(2);
end
a = 0.05; % set significant level
beta_lower = prctile(boots_beta,a/2); % calculate upper and lower bounder
    with prcentile method
beta_upper = prctile(boots_beta,(1-a)/2);
mu_lower = prctile(boots_mu,a/2);
mu_upper = prctile(boots_mu,(1-a)/2);


%Q3(d
T = 3*14*100; % T value
year_return = boots_mu - boots_beta*log(-log(1-(1/T))); %put the estiamted
    parameter in (c) with T
year_return_lower = prctile(year_return,a/2); % calculate upper and lower
    bounder with prcentile method
year_return_upper = prctile(year_return,(1-a)/2);
```

```
%Q3(e


%Q3(f
non_boots_beta = zeros(B,1); % replace = 0 for sampling without replacement
non_boots_mu = zeros(B,1);
replace = 1; % replace = 0 for sampling without replacement
rng(123);
for ii=1:B
    % simulate bootstrapped data
    x = randsample(atlantic,n,replace);
    out = est_gumbel(x);
    non_boots_beta(ii) = out(1);
    non_boots_mu(ii) = out(2);
end

non_beta_lower = prctile(non_boots_beta,a/2); % calculate upper and lower
   bounder with prcentile method
non_beta_upper = prctile(non_boots_beta,(1-a)/2);
non_mu_lower = prctile(non_boots_mu,a/2);
non_mu_upper = prctile(non_boots_mu,(1-a)/2);

# Q4
Wage<-read.table("Wage.txt",header=TRUE)
a = 0.1 # significant level
n = 3000 # number of sample
B = 2000 # time of resample
boot_betahat_1 <- matrix(0,2000,4) # place to store our result
m6 <-lm(wage~poly(age,3,raw=TRUE), data = Wage)  # original regression

set.seed(321) # resampleing process
for (i in 1:B){
  x <- sample.int(n, size = n, replace = TRUE)
  resample <- Wage[x,]
  m7 <-lm(wage~poly(age,3,raw=TRUE), data = resample)
  boot_betahat_1[i,1] <- summary(m7)$coefficients[1]
  boot_betahat_1[i,2] <- summary(m7)$coefficients[2]
  boot_betahat_1[i,3] <- summary(m7)$coefficients[3]
  boot_betahat_1[i,4] <- summary(m7)$coefficients[4]
}

# gen confidence interval with boot and confint

boot_inter_beta0 <- quantile(boot_betahat_1[,1], c(a/2, 1-(a/2)))
boot_inter_beta1 <- quantile(boot_betahat_1[,2], c(a/2, 1-(a/2)))
boot_inter_beta2 <- quantile(boot_betahat_1[,3], c(a/2, 1-(a/2)))
boot_inter_beta3 <- quantile(boot_betahat_1[,4], c(a/2, 1-(a/2)))

inter_beta0 <- confint(m6, level = 0.9)[1,1:2]
inter_beta1 <- confint(m6, level = 0.9)[2,1:2]
inter_beta2 <- confint(m6, level = 0.9)[3,1:2]
inter_beta3 <- confint(m6, level = 0.9)[4,1:2]
```