Problem Set 1

Philip Kruger
18328699
11/02/23

Question 1

First the data, empirical distribution and D-statistic were obtained using the following code(provided in the problem set):

```
set.seed(123) #setting seed
data <- unlist(rcauchy(1000, location = 0, scale = 1)) #initilising data
ECDF <- ecdf(data) #getting the empirical distiribution function
empiricalCDF <-ECDF(data)
D <- max(abs(empiricalCDF - pnorm(data))) #getting the D statistic
```

To find the p-value it requires some inquery of the formula:

$$p(D \leq x) = \frac{\sqrt{2\pi}}{x} \sum_{k=1}^{\infty} e^{\frac{-(2k-1)^2\pi^2}{8x^2}}$$

Firstly we can see that it is an infinite sum which cannot be done in R. However, k is in the negative power thus a larger k would be a smaller contribution. As such, the infinite sum can be approximated by a smaller max k. Secondly, for the $P(D \leq x)$, we can see that this value is largest when x is the maximised. Thus, we will be using the value x = D to find the p-score for this D statistic. This is done with the following R code:

```
a = 0 #initialising "a" which counts the sum
x <- D #setting x in the formula to our D statistic
for (k in 1:10000){ #for loop is the sum in formula, using 10,000 as a stand in for inf
b = exp(-(((2*k)-1)^2)*(pi^2)/(8*(x^2))) #the part getting summed over
a = a+b #increase a for the value
}
c = (sqrt(2*pi)/x)*a #multiply by not summed part
```

This gives us the p-value: 5.652523e-29. which indicates that we can reject the null hypothesis that the data sets come from a normal distribution. This makes sense since the cauchy and normal distribution are different.
We can also test our results by using the ks.test function in R:

```
print(ks.test(empiricalCDF, pnorm(data)))
```

which gives output:

```
Asymptotic two-sample Kolmogorov-Smirnov test

data:  empiricalCDF and pnorm(data)
D = 0.135, p-value = 2.432e-08
alternative hypothesis: two-sided
```

Thus we get the same D-statistic but a different p-value. However both p-values solidly reject the null hypothesis.

Question 2

First the data was loaded in with the following code from the assignment

```
fset.seed(123) #initialisng seed again
data <- data.frame(x = runif(200,1,10)) #generating x column
data$y <- 0 +2.75*data$x + rnorm(200,0,1.5) #generating y column
```

the lm function was ran to compare to the newton-raphson results and the points were graphed with the following code:

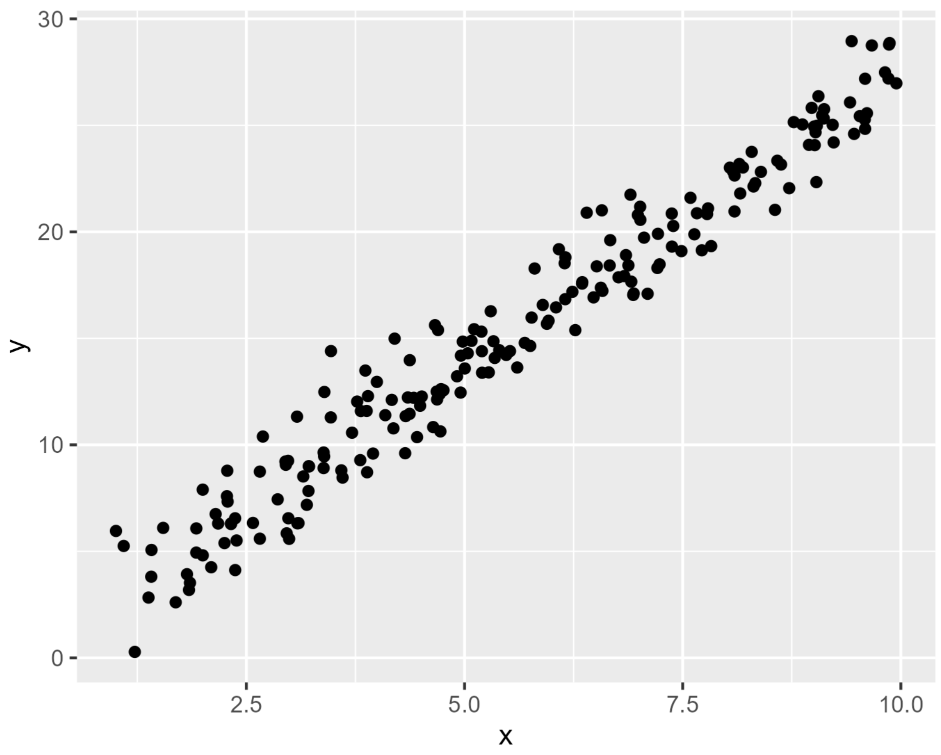
```
print(summary(lm(data$y~data$x))) #getting the a and B with the lm function
ggplot(data, aes(x,y))+ #graphing y against x
geom_point() #even if i didnt have the results from lm. I could use the graph
ggsave("hw1_graph.png", width = 5, height = 4, units = 'in', dpi = 300)#to guess initail parameters
```

From this we got the output and graphs:

```
Residuals:
Min      1Q  Median      3Q     Max
-3.1906 -0.9374 -0.1665  0.8931  4.8032

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.13919    0.25276   0.551    0.582
data$x       2.72670    0.04159  65.564   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.447 on 198 degrees of freedom
Multiple R-squared:  0.956,Adjusted R-squared:  0.9557
F-statistic:  4299 on 1 and 198 DF,  p-value: < 2.2e-16
```



y graphed against x

The above results can also be obtained using the formula for residuals and the optim() function. It can be specified that the Newton-Raphson method is used in optim(). This is done with the following code:

```
ols_reg_func <- function(theta, y, x){ #creating the function to be optomised
a <- theta[1] #it is a linear regression with one independent variable
B <- theta[2] #thus it is only a function of alpha and beta
sigma <- theta[3] #the standard deviation of errors is needed for optomisation too

Res = y-a-B*x #the classic linear regression formula for the residuals
-sum(dnorm(Res,mean = 0, sigma, log = TRUE)) #and the quasi random distribution of residuals
}

a_guess <- 0.1 #from the graph I know that both alpha and beta are low
B_guess <- 0.1 #this is just setting initaial terms to optomise from
sigma_guess <- 0.1 #I think I need to be careful to keep these under the actual values of the parameter
#otherwise the newton raphson will look for a local miinimum many sd away


#putting the function, data and initial guesses into optim() function
optim_results <- optim(fn = ols_reg_func,par = c(a = a_guess, B = B_guess, sigma = sigma_guess),
y = data$y,x = data$x,method = "BFGS") #setting optim to use BFGS

print(optim_results$par)#printing results
```

this gets the results:

```
        a         B     sigma
0.1391868 2.7266986 1.4395451
```

This is fairly similar to the results of a = 0.139, B = 0.252 and Sigma = 1.447 which were obtained from the lm() function.