Problem Set 1

Student: Shekhar Kedia (23351315)
Applied Stats II

Due: February 11, 2024

Instructions

- Please show your work! You may lose points by simply writing in the answer. If the problem requires you to execute commands in `R`, please include the code you used to get your answers. Please also include the `.R` file that contains your code. If you are not sure if work needs to be shown for a particular problem, please ask.
- Your homework should be submitted electronically on GitHub in `.pdf` form.
- This problem set is due before 23:59 on Sunday February 11, 2024. No late assignments will be accepted.

Question 1

The Kolmogorov-Smirnov test uses cumulative distribution statistics test the similarity of the empirical distribution of some observed data and a specified PDF, and serves as a goodness of fit test. The test statistic is created by:

$$D = \max_{i=1:n} \left\{ \frac{i}{n} - F_{(i)}, F_{(i)} - \frac{i-1}{n} \right\}$$

where F is the theoretical cumulative distribution of the distribution being tested and $F_{(i)}$ is the ith ordered value. Intuitively, the statistic takes the largest absolute difference between the two distribution functions across all x values. Large values indicate dissimilarity and the rejection of the hypothesis that the empirical distribution matches the queried theoretical distribution. The p-value is calculated from the Kolmogorov- Smirnoff CDF:

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

which generally requires approximation methods (see Marsaglia, Tsang, and Wang 2003). This so-called non-parametric test (this label comes from the fact that the distribution of

the test statistic does not depend on the distribution of the data being tested) performs poorly in small samples, but works well in a simulation environment. Write an `R` function that implements this test where the reference distribution is normal. Using `R` generate 1,000 Cauchy random variables (`rcauchy(1000, location = 0, scale = 1)`) and perform the test (remember, use the same seed, something like `set.seed(123)`, whenever you're generating your own data).

As a hint, you can create the empirical distribution and theoretical CDF using this code:

```
  # create empirical distribution of observed data
  ECDF <- ecdf(data)
  empiricalCDF <- ECDF(data)
  # generate test statistic
  D <- max(abs(empiricalCDF - pnorm(data)))
```

Solution:
Firstly using `R`, I create 1,000 Cauchy random variables running the following codes:

```
# Generating 1000 cauchy random variables
set.seed(123)
n <- 1000
empirical <- rcauchy(n, location = 0, scale = 1)
```

Next, I create the function to perform Kolmogorov-Smirnov test running the following codes:
N.B.: The function produces two output, one the test statistics (D) and p-value

```
# Creating function to perform the K-S test
k_s_test <- function (data){
  # creating empirical distribution of observed data
  ECDF <- ecdf(data)
  empiricalCDF <- ECDF(data)
  # generating test statistic
  D <- max(abs(empiricalCDF - pnorm(data)))
  cat("D =", D)
  temp <- c() #Empty vector to store the summation result after each iteration
  for(i in 1:n){
    temp <- c(temp, exp((-(2 * i - 1)^2 * pi^2) / ((8 * D)^2)))
  }
  p_value <- sqrt(2 * pi)/D * sum(temp)
  cat("\np-value =", p_value)
}
```

Now, I compare the results of the test using the created function and using in-built `R` function. The result of the created K-S test function is:

```
# Running the K-S test function
k_s_test(empirical)
```

Output:

```
D = 0.1347281
p-value = 0.003801528
```

And, the result of K-S test using the in-built R function is:

```
# Comparing the output with in-built R-function
ks.test(empirical, pnorm)
```

Output:

```
Asymptotic one-sample Kolmogorov-Smirnov test
data: empirical
D = 0.13573, p-value = 2.22e-16
alternative hypothesis: two-sided
```

We can see that the output of the function created manually and the in-built function is approximately similar. The minor differences in the decimal values is because of rounding and can be ignored.

Question 2

Estimate an OLS regression in R that uses the Newton-Raphson algorithm (specifically BFGS, which is a quasi-Newton method), and show that you get the equivalent results to using lm. Use the code below to create your data.

```
set.seed (123)
data <- data.frame(x = runif(200, 1, 10))
data$y <- 0 + 2.75*data$x + rnorm(200, 0, 1.5)
```

Solution:
Firstly, I run the above code to create the data. Then, I create a function to calculate the sum of squared residuals using the following R codes:

```
# Defining the function to calculate the sum of squared residuals
residuals_ols <- function(coef) {
  sum((data$y - (coef[1] + coef[2] * data$x))^2)
}
```

Then, I estimate the coefficients using the BFGS algorithm and plotting the results on a scatter plot to get an idea of the OLS regression using the following R codes:

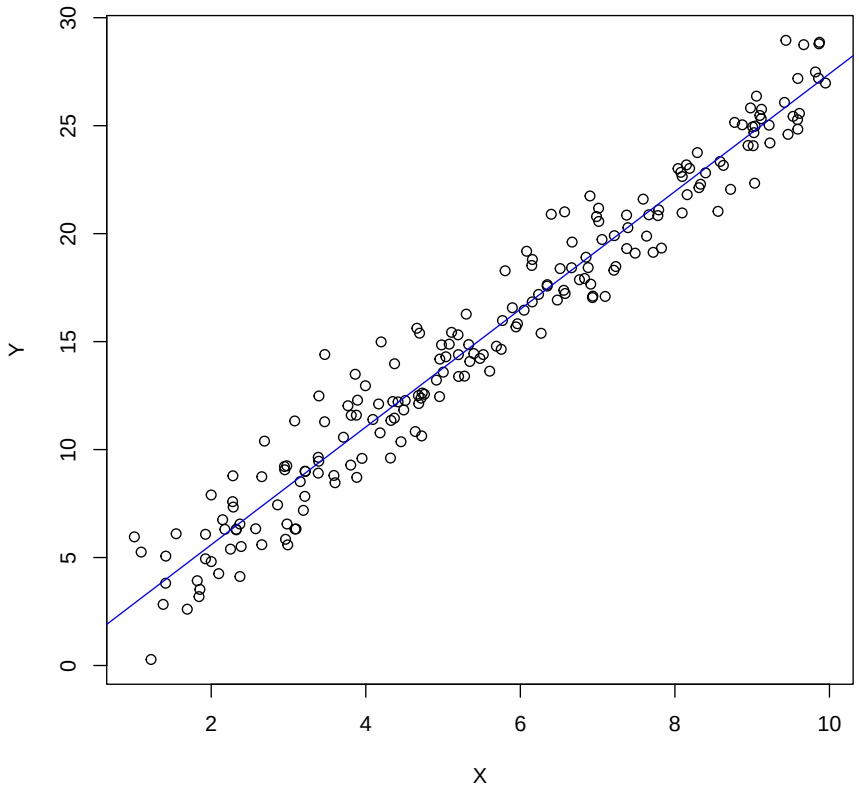
```
# Estimating coefficients using BFGS algorithm
bfgs_ols <- optim(par = c(intercept = 0, slope = 0), fn = residuals_ols,
    method = "BFGS")
bfgs_ols$par #Displaying the intercept and slope

# Plotting the graph using output from BFGS algorithm
pdf("plot_Y_X_bfgs.pdf")
plot(data$x, data$y, ylab = 'Y', xlab = 'X')
abline(bfgs_ols$par, col = "blue")
dev.off()
```

Output:

```
intercept      slope
0.1391778    2.7267000
```

Scatter plot showing relationship between Y and X and the OLS regression line using BFGS algorithm:



Finally, I estimate the coefficients using the in-built lm() function to ensure I get the same results and also, plotting the output to see the similarity between the two models.
Following R codes is used:

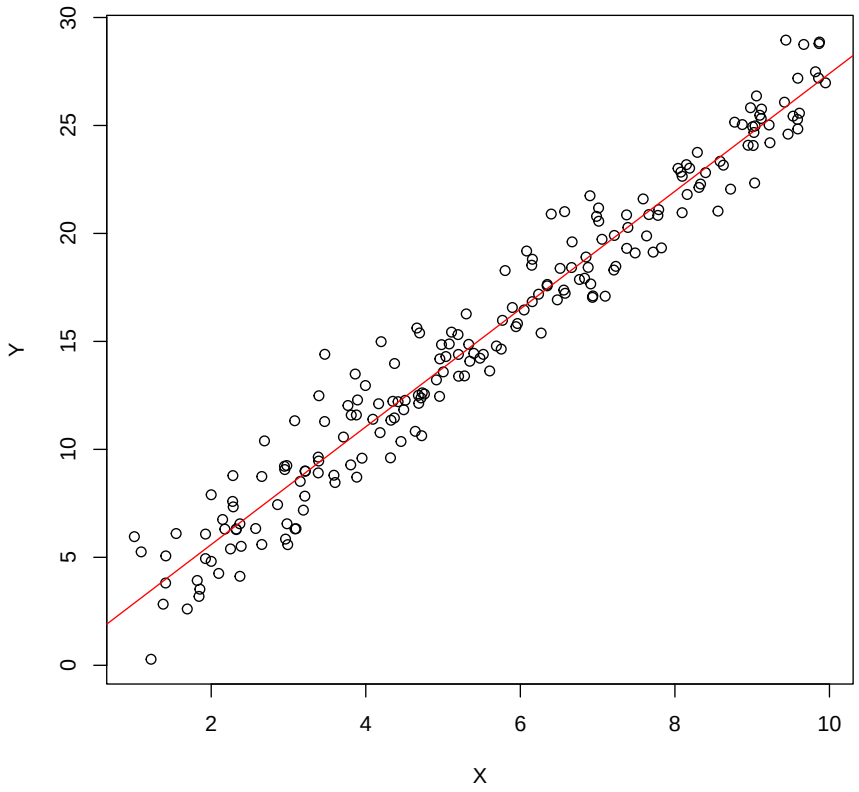
```r
# Confirming the same result with lm()
lm_coef <- coef(lm(y ~ x, data = data))
lm_coef #Displaying the intercept and slope

# Plotting the graph using output from lm()
pdf("plot_Y_X_lm.pdf")
plot(data$x, data$y, ylab = 'Y', xlab = 'X')
abline(coef(lm(y ~ x, data = data)), col = "red")
dev.off()
```

Output:

```
(Intercept)           x
0.1391874     2.7266985
```

Scatter plot showing relationship between Y and X and the OLS regression line using lm() function:



We can see that both the models (using BFGS algorithm and lm() function in R) produced equivalent results to estimate the OLS regression.