Problem Set 1

Applied Stats II
Shelly Veal-Upham
25337422

Due: February 11, 2026

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).

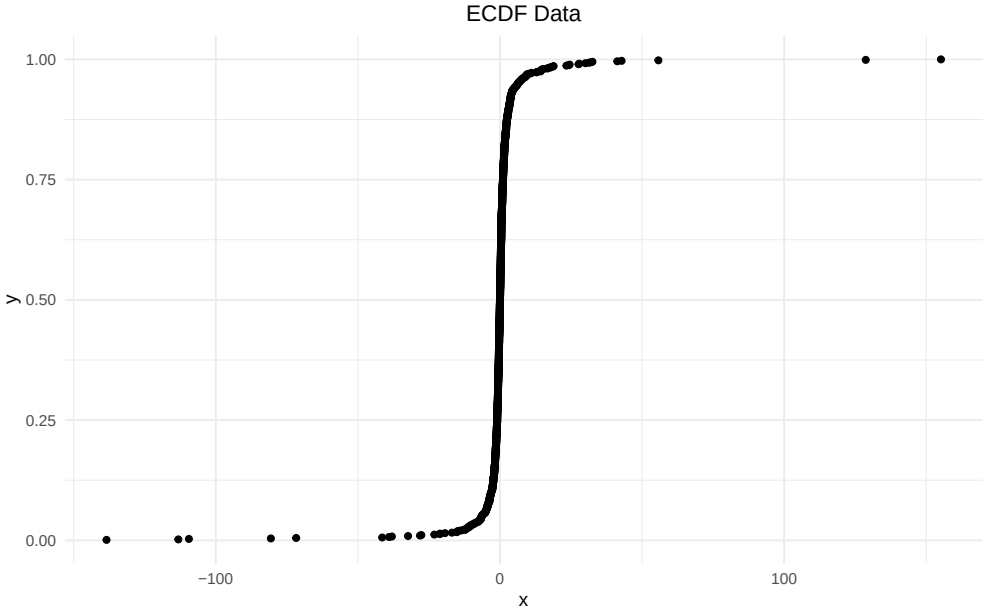
First, let's take a look at the data:

```r
set.seed(123)
# create empirical distribution of observed data
data_1 <- rcauchy(1000, location = 0, scale = 1)
ECDF <- ecdf(data_1)
empiricalCDF <- ECDF(data_1)
# generate test statistic
D <- max(abs(empiricalCDF - pnorm(data_1)))

df <- data.frame(
  x = data_1,
  y = empiricalCDF
)

ECDF_plot <- ggplot(df, aes(x = x, y =y)) + # Visualizing
  geom_point() +
  labs(title = "ECDF Data") +
  theme(plot.title = element_text(hjust = 0.5))
```

Figure 1: ECDF Distribution of Data



The range on the x axis is from the negative hundreds to the positive hundreds, while y only spans from 0 to 1. This makes sense, as the cumulative distribution is made up of probabilities (ranging 0:1). Knowing that there are different kinds of cumulative distributions, I developed the Kolmogorov-Smirnov test by hand with the intention of making it at least a little more flexible than only working with the Empirical CDF:

```
KS_test <- function(F_Fun, x) {

  if (identical(F_Fun, ecdf)) {
    x <- sort(x)
    n <- length(x)
    i <- seq_along(x)
    F_inter <- F_Fun(x)
    FVals <-  F_inter(x)
    }
  else {
    n <- length(x)
    i <- seq_along(x)
    FVals <- F_Fun(x)
  }

  D <- max(abs(FVals - pnorm(x)))
  p_val <- (sqrt(2*pi)/D)*sum(exp(-(2*i-1)^2*pi^2)/(8*(D^2)))

  return(list(
    D = D,
    p_value = p_val))
}

KS_test(F_Fun = ecdf, x = data_1)
ks.test(data_1, "pnorm")
```

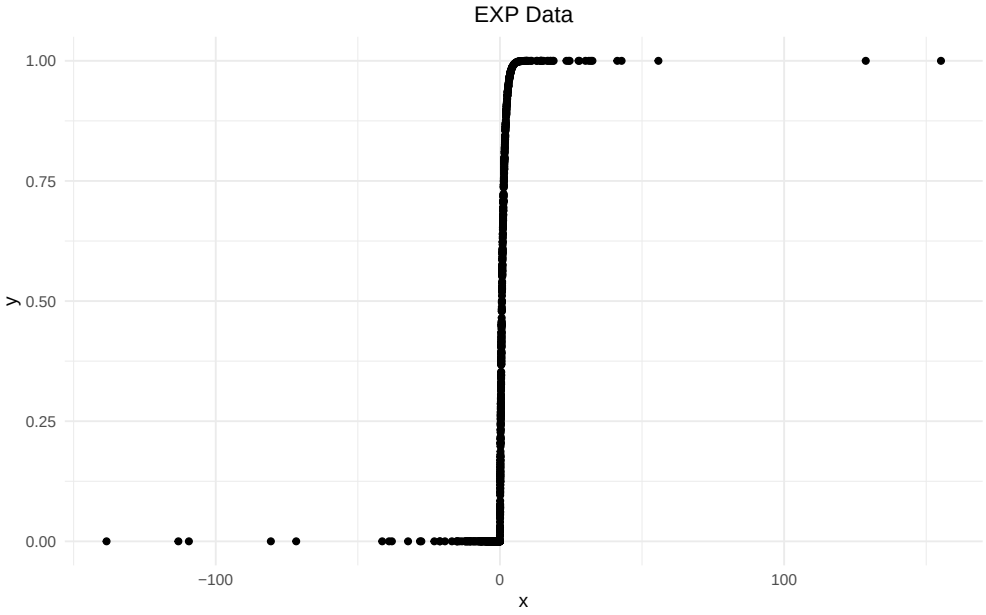
As expected, the outputs of my KS test function and the built-in `ks.test()` are nearly identical:

$$\texttt{KS\_test()}\text{: } D = 0.1347, p = 0.0066$$
$$\texttt{ks.test()}\text{: } D = 0.1357, p = 2.2e^{-16}$$

There is a notable difference in the p-values– the built-in `ks.test()` results in a p-value much closer to 0 than my function, which is likely a result of my crude approximation of the infinite sum using a sum of only the length of the data.

Perhaps in futility I decided to create my KS test function such that it might accommodate other Cumulative Distribution Functions besides ECDF, such as `pexp()`:

Figure 2: ECDF Distribution of Data




```r
pexp_y <- pexp(data_1)

df_pexp <- data.frame(
  x = data_1,
  y = pexp_y
)

exp_plot <- ggplot(df_pexp, aes(x = x, y =y)) + # Visualizing
  geom_point() +
  labs(title = "EXP Data") +
  theme(plot.title = element_text(hjust = 0.5))

KS_test(F_Fun = pexp, x = data_1)
ks.test(data_1, pexp)
```

Again, the outputs are nearly identical:

$$\texttt{KS\_test()}\text{: } D = 0.4993, p = 0.0001$$
$$\texttt{ks.test()}\text{: } D = 0.4930, p = 2.2e^{-16}$$

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`.

Utilizing the tutorial material from this week as a rough template, I developed the following code for my OLS by hand:

```
1  set.seed (123)
2  data_2 <- data.frame(x = runif(200, 1, 10))
3  data_2$y <- 0 + 2.75*data_2$x + rnorm(200, 0, 1.5)
4
5  squared_errors <- function(outcome, input, parameter) {
6    input_mat <- as.matrix(input)
7    n <- nrow(input_mat)
8    k <- ncol(input_mat)
9    beta <- parameter[1:k]
10   sum_sq_e <- sum((outcome - input_mat%*%beta)^2)
11   return(sum_sq_e)
12 }
13
14 results_OLS <- optim(fn=squared_errors, outcome=data_2$y,
15                       input=cbind(1, data_2$x), par=c(1,1),
16                       hessian=T, method="BFGS")
17
18 results_OLS$par
19 lm(data_2$y ~ data_2$x)
```

The output of my function and the `lm()` function are identical, and well approximate the β values we would expect from our original formula (seen in line 3 of the above `R` code):

`results_OLS$par`: $\beta_0 = 0.1392, \beta_1 = 0.2.727$
`lm(data_2$y ~ data_2$x)`: $\beta_0 = 0.1392, \beta_1 = 0.2.727$