Problem Set 1

Applied Stats II — Caoimhe Ni Mhaonaigh

Due: February 14, 2022

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 class on Monday February 14, 2022. No late assignments will be accepted.
- Total available points for this homework is 80.

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:

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

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).

Write an `R` function that implements this test where the reference distribution is normal. 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)))
```

To preform this test, I will firstly establish the null and alternative hypotheses for the Kolmogorov-Smirnov test.

The null hypothesis is that the data follows a certain distribution
The alternative hypothesis is that the data does not follow a certain distribution.

To understand the ECDF distribution I used the following code to visualise it:

```
plot(ECDF, verticals=TRUE, do.points=FALSE, col="blue")
```

This produced the following graph:

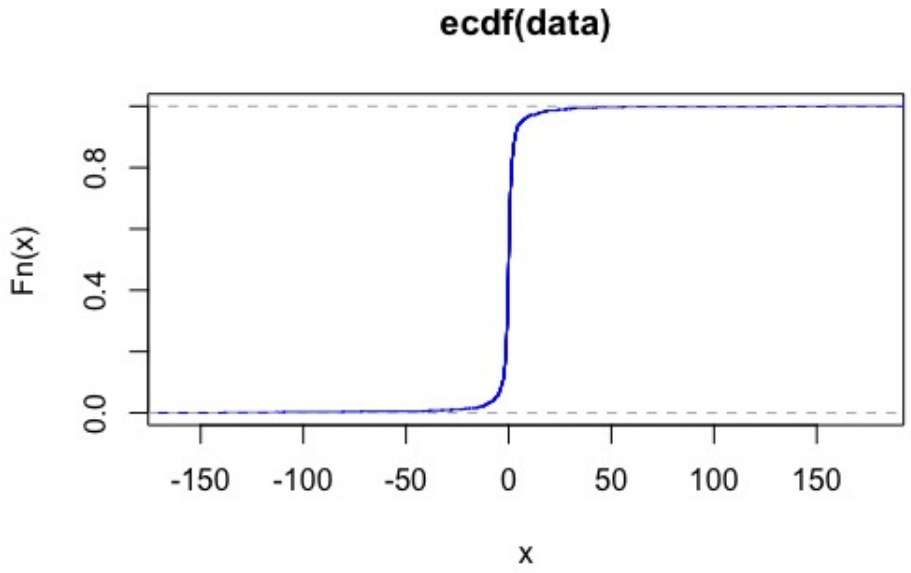

Figure 1: Plotting the ECDF

Following on from this, to further understand the data, I used the built R function: "ks.test" to examine what output I am looking for, when writing my own function.

In this particular question we are looking at whether the data follows a normal distribution. Therefore we are comparing the data to a "pnorm" distribution.

```
set.seed(123)
options(scipen = 999)
data <- rcauchy(1000, location = 0, scale = 1)
ks.test(data, "pnorm")
```

This produces a Kolmogorov-Smirnov test to examine the goodness of fit:

```
One-sample Kolmogorov-Smirnov test

data:  empirical
D = 0.13573, p-value = 0.000000000000000222
alternative hypothesis: two-sided
```

Next, I used the code given above to find the test statistic: If the test statistic "D" is large it suggests that the distance between the distributions is large and will lead to the rejection of the null hypothesis that the data follows a certain, in this case normal, distribution.

```
set.seed(123)
options(scipen = 999)
data <- rcauchy(1000, location = 0, scale = 1)
ECDF<- ecdf(data)
empiricalCDF <- ECDF(data)

# generate test statistic
D<- max(abs(empiricalCDF - pnorm(data)))
```

This gave the following output:

```
0.1347281
```

Which matches the test statistic that the built in function produced. This is a small test statistic, which would suggest we fail to reject the null hypothesis, but firstly we need to find a p value at an alpha confidence level, to be sure that we can reject the null.

To find the p value I wrote a function to implement the following two formulae:

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

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

```
kolmogorov_smirnov <- function(data) {
  D<- max(abs(empiricalCDF - pnorm(data)))
  for(k in range(1:length(data))){
    q <- -((2*k-1)^2)*(pi^2)
    z <- sqrt(2*pi)
```

```
    for (x in data){
      p <- ((z/x)*sum(exp(q)/(8*(x^2))))
      return(c(D, p))
      }
    }
  }
```

When using the data set from above this gave produces the following output:

```
>kolmogorov_smirnov(data)
[1] 0.134728061606 0.000007928068
```

This p value is below the alpha level of 0.05, which suggests that we fail to reject the null hypothesis that the data follows a normal distribution.

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`.

The following data that was used:

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

First I estimated the Regression using the built in lm function:

```
coef(lm(data$y ~ data$x))
```

This produced the following output:

```
coef(lm(data$y ~ data$x))
(Intercept)       data$x
  0.1391874    2.7266985
```

Then by using the optim function I was able to estimate the intercept and beta values of the following function:

```
OLS_estimate <- function(outcome, input, parameter) {
  n      <- nrow(input)
  k      <- ncol(input)
  beta   <- parameter[1:k]
  sigma2 <- parameter[k+1]^2
  e      <- outcome - input%*%beta
  logl   <- -.5*n*log(2*pi)-.5*n*log(sigma2) - ( (t(e) %*% e)/ (2*sigma2) )
  return(-logl)
}
```

By running the function through the optim function and using the Newton-Raphson algorithm, it showed that the intercept and beta produced in the manual function matches the output using the lm function:

```
results_norm <- optim(fn=OLS_estimate,
                      outcome=data$y,
                      input=cbind(1, data$x),
                      par=c(1,1,1),
                      hessian=T,
                      method="BFGS")
results_norm$par
[1]  0.1398324  2.7265559 -1.4390716
```

We can see that this is the same as the output produced by the built in function