Problem Set 1

Applied Stats II

Due: February 12, 2023

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 12, 2023. 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:

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

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).

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

```
set.seed(123)
# create empirical distribution of observed data
ECDF <- ecdf(data)
empiricalCDF <- ECDF(data)
# generate test statistic
D <- max(abs(empiricalCDF - pnorm(data)))
```

My solutions to Question 1

```
#Making data reproducible
set.seed(123)

# Creating data comprising 1,000 Cauchy random variables
data  <- rcauchy(1000, location = 0, scale = 1)

# Creating my own Kolmogorov-Smirnov test function
kol_smir.test <- function(x, y) {
  ECDF <- ecdf(x)
  empiricalCDF <- ECDF(x)
  D <- max(abs(empiricalCDF - pnorm(x)))
  p_value <- pt(q=D, df=length(x)-1, lower.tail = FALSE)
  return(c("D" = D, "p-value" = p_value))
}

# Performing my Kolmogorov-Smirnov test
kol_smir.test(data,"pnorm")
```

Because the p-value is above 0.05, we can't reject the null hypothesis. I.e., there's insufficient evidence to say data isn't normally distributed.

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)
```

My solutions to Question 2

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

# Creating a function to minimise residuals to use in the Newton-Raphson
  algorithm
minimise_residuals <- function(data, par) {
  with(data, sum((par[1] + par[2] * x - y)^2))
}

# Estimating an OLS regression using the Newton-Raphson algorithm
optim(par = c(0,1), fn=minimise_residuals, data=data,
method = "BFGS")

# Comparing the results of the Newton-Raphson algorithm to the lm() function
lm(y ~ x, data=data)
```

Using the BFGS method of the optim() function, we get the equivalent results as using the lm() function: y = 0.1392 + 2.7267x.