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:

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

Q1 Answer

The following function implements the Kolmogorov-Smirnov test using a normal distribution as the reference. The null hypothesis states that both samples are from populations with identical distributions. If the P value is small, this indicates that the two groups were sampled from populations with different distributions.

```
library(dplyr)
set.seed(123)
data <- rcauchy(1000, location = 1, scale = 1)
ECDF <- ecdf(data)
empiricalCDF <- ECDF(data)

D <- max(abs(empiricalCDF-pnorm(data)))

ks_test <- function(x, d){
  #Run Kolmogorov-Smirnov test on x = data and d = D test statistic.
  pv <- sum(x >= d)/ nrow(data.frame(x))
  print(paste("Asymptotic two-sample Kolmogorov-Smirnov test.
  D = ", d, "p value ="))
  print(pv)
}

ks_test(empiricalCDF, D)

Output:

> ks_test(empiricalCDF, D)
[1] "Asymptotic two-sample Kolmogorov-Smirnov test.\n  D =  0.361940356685141
```

```
p value ="
[1] 0.639
```

Based on the p value of .639 we fail to reject the null hypothesis that the two samples were drawn from the same distribution at a significance level alpha = .05.

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)
```

Q2 Answer

The following code estimates the OLS regression in R using the BFGS method for the Newton-Raphson algorithm.

```
library(tidyverse)

set.seed(123)

datas <- data.frame(x=runif(200,1,10))

datas$y <- 0 + 2.75*datas$x + rnorm(200,0,1.5)
linear.lik <- function(theta,y,X){
  n <- nrow(X)
  k <- ncol(X)
  beta <- theta [1:k]
  sigma2 <- theta[k+1]^2
  e <- y - X%*%beta
  logl <- -.5*n*log(2*pi)-.5*n*log(sigma2)-((t(e)%*%
                            e)/(2*sigma2))
  return(-logl)}
```

```
ols_R <- optim(fn=linear.lik, par = c(1,1,1), hessian = TRUE,
                    y=datas$y, X=cbind(1,datas$x), method = "BFGS")

ols_R$par

Output:
> ols_R$par
[1]  0.1398324  2.7265559 -1.4390716
```

The lm function was then implemented to demonstrate that the same results were obtained:

```
    lm(datas$y ~ datas$x)
Output:
Coefficients:
(Intercept)      datas$x
     0.1392       2.7267
```