# 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 $i$th 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:

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

My Answer for Question 1

```
  set.seed(123) # as suggested in Question 1 to create reproducible results
# and so we get the same answers

# need to generate 1,000 Cauchy random variables
r <- 1000 # call this r
data1 <- rcauchy(r, location = 0, scale = 1)

# generate the Empirical Cumulative Distribution Function for the
# empirical distribution of observed data
ECDF <- ecdf(data)
empiricalCDF <- ECDF(data)

# generate the Test Statistic using the P value (p) we were given in R
D <- max(abs(empiricalCDF - pnorm(data)))
pi <- 3.142
p_value <- sqrt(2*pi)*sum((1)^2)*(pi)^2/(8*(D)^2)

# incorporated the given function
# Kolmogorov-Smirnov test (KS)

KS <- function(data) {
  ECDF <- ecdf(data)
  empiricalCDF < ECDF(data)
  # create the Test Statistic
  D <- max(abs(empiricalCDF - pnorm(data)))

  sqroot <- sqrt(2*pi)
  power_value <- ((1)^2)*((pi)^2)/(8*(D)^2)
  p_value <- sqroot*sum(exp(power_value))
  output <- list(D, p_value)
  return(output)
}

ks.test(data1, "pnorm") # using the KS test R function


```

```r
# RESULT
# One-sample Kolmogorov-Smirnov test
# data:  data1
# D = 0.13573, p-value = 2.22e-16
# alternative hypothesis: two-sided

# Outputs
# D = 0.13573
# p-value = 2.22e-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`. Use the code below to create your data.

```r
# generate the Empirical Cumulative Distribution Function for the
# empirical distribution of observed data
ECDF <- ecdf(data)
```

My Answer for Question 2

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

# I can estimate the Regression using the lm() function
coef(lm(data$y ~ data$x))
# this gives
# (Intercept)       data$x
# 0.1391874    2.7266985

```

```r

# create log normal likelihood function
# using Tutorial 3 and the given code above to derive log-likelihood function

#
norm_likelihood1 <- function(outcome, input, parameter) {
  n <- nrow(input) # number of rows
  k <- ncol(input)
  beta   <- parameter[1:k] # numbers or Betas or
  sigma1 <- parameter[k+1]^2 # sigma helps to test the significance
  e      <- outcome - input%*%beta
  logl   <- -.5*n*log(2*pi)-.5*n*log(sigma1) - ( (t(e) %*% e)/ (2*sigma1) )
  return(-logl)
}

# generate the norm_likelihood function and
# show that you get the equivalent results to using lm
# another way to generate the log likelihood function

norm_likelihood2 <- function(outcome, input, parameter) {
  n <- ncol(input)
  beta <- parameter[1:n]
  sigma <- sqrt(parameter[1+n])
  -sum(dnorm(outcome, input %*% beta, sigma, log=TRUE))
}

# print our estimated coefficients (intercept and beta_1)
# here the above functions norm_likelihood1  and norm_likelihood2
# can be run through the built in optim() function using the
# Newton Raphson algorithm which is an iterative procedure that can be used
# to calculate MLEs
results_norm1 <- optim(fn=norm_likelihood1, outcome=data$y, input=cbind(1,
    data$x), par=c(1,1,1), hessian=T, method="BFGS")
results_norm2 <- optim(fn=norm_likelihood2, outcome=data$y, input=cbind(1,
    data$x), par=c(1,1,1), hessian=T, method="BFGS")

results_norm1$par
# Intercept = 0.1396113
# Beta = 2.7266403

results_norm2$par
# Intercept = 0.1404966
# Beta = 2.7265080

# confirm that we get the same thing in with glm()
coef(lm(data$y~data$x))

# Results
# (Intercept)       data$x
# 0.1391874    2.7266985
```