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:

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

Question 1 Answer

Using R, the following function was created:

```
ksfunc <- function(dat) {
  # create empirical distribution of observed data
  ECDF <- ecdf(dat)
  empiricalCDF <- ECDF(dat)

  # generate statistic
  D <- max(abs(empiricalCDF - pnorm(dat)))

  # generate p-value from base R function
  res <- ks.test(empiricalCDF, pnorm(dat))

  # generate p-value
  k <- 1
  fun <- function(n, x){
    exp(((-((2*n)-1)^2)*(pi^2))/((8)*(x^2)))
  }
  mul <- ((sqrt(2*pi))/(D))
  ans <- 0
  tol <- 1e-06

  while(TRUE) {
    next_term <- fun(k, D)
    next_term <- next_term*mul
    ans <- ans + next_term
    if(abs(ans)<tol){
      break
    }
```

```
    k <- k+1
  }

  ans <- ans*mul
  results <- data.frame(stat = D, p.value = res$p.value, func = ans)
  return(results)

}
```

Using the function mentioned above to generate 1,000 Cauchy random variables, this data was then used to test for normality using the "ksfunc()" function and produced the following output:

```
stat      p.value          func
1 0.1347281 2.432277e-08 1.051657e-27
```

The "stat" value indicates the D statistic obtained from the first formula, while the "p.value" number indicates the P-value from the test obtained from R's built in ks.test() function. With the p-value being less than .05, we can reject the null hypothesis that the two CDF's are the same, meaning the data is not normal.
The "func" stat is meant to be the same as the "p.value" stat. Unfortunately, I could not implement the KS PDF function provided above into R to calculate the p-value. I tried to create the infinite sequence with a tolerance of (`1e-06`), but the first term seemed to be lower than the tolerance, so the function would stop running after the first term of the sequence.

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)
```

Question 2 Answer:

First, the linear likelihood function was created, which finds the log of the likelihood of the PDF of the linear function, being

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y_i - x_i\beta}{2\sigma^2}}$$

This was done using the `linearl` function as shown below:

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

The optimisation function in R was then used to find the MLE of the parameters, by utilising a Hessian matrix and the Newton-Raphson algorithm. The data was created using the code in the above question and passed through the `optim` function.

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

This gave the following output:

```
[1] 0.1391893 0.2516985 1.4395471
```

An OLS linear regression was then run on the same data using `lm(data_ex$y ~ data_ex$x)`, which generated the following table: As shown, the X value, or $\beta 0$ value, in the table corresponds to the second value in the output from the `linear.MLE` output, while the Constant, or $\beta 1$ value corresponds to the first value of the output. This displays how OLS and MLS give the same output for these types of linear equations.

Table 1: Regression

	Dependent variable:
	y
x	0.252^{***}
	(0.042)
Constant	0.139
	(0.253)
Observations	200
R^2	0.156
Adjusted R^2	0.152
Residual Std. Error	1.447 (df = 198)
F Statistic	36.629^{***} (df = 1; 198)
Note:	*p<0.1; **p<0.05; ***p<0.01