Problem Set 1

Lily Rice - 16304845 - 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 19, 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)))
```

ANSWER: First, I ran all code listed above. This was done as shown below:

```r
# First, set.seed() as instructed
set.seed(123)
# Next, create my data using the Cauchy random variables, as instructed
data <- rcauchy(1000, location = 0, scale = 1)
# create empirical distribution of observed data (unchanged by me, from PS01
    doc)
ECDF <- ecdf(data)
empiricalCDF <- ECDF(data)
# generate test statistic (as shown on PS01 doc)
D <- max(abs(empiricalCDF - pnorm(data)))
# this returns a D = 0.134728...
```

Then, I estimated the p-value for this data using the ks.test() function in R. Jeff's equation for D uses the absolute value, which means we are conducting a two-tailed ks.test, which also means that rejection of the null hypothesis is determined by this p-value being lower than "alpha" (usually 0.05). I ran this ks.test with the following code:

```r
kstest_on_data <- ks.test(data, "pnorm")
list(kstest_on_data)
# Note: produces D = 0.136 (approximately the same value produced by Jeff's
    function)
# ; and p = <2.22e-16.
```

I was sure that this was the p-value I should be looking for when estimating it manually, as this test produced the correct D estimate (that matched the estimate from Jeff's test statistic equation).

This p-value of less than 2.22e-16 is extremely small, and definitely below the "default" significance level (0.05). This is grounds to reject our Null Hypothesis, that there is no difference between the reference function (the normal distribution) and the emperical cumulative distribution function (in other words, is grounds to reject the hypothesis that our data's CDF is normal).

I then attempted to estimate the p-value manually. I tried multiple different methods: including writing a new function, trying MLE to do so, pnorm(), and others, but was untimately unsuccessful at getting a function that produced a p-value that was appropriate. I have not included this code in this .pdf, however it is in the .R file if needed (and for my reference when correcting my work).

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.

I used the code listed to create my data first:

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

Then, I ran my OLS regression that uses the Quasi-Newton method on this data (first plotting my data just to have a look; then by coding the relationship between variables using R, and finally by running the regression on this coded linear likelihood equation). These steps can be seen in the code below:

```r
# First, plotted the data:
plot(data$x, data$y, ylab = 'Y', xlab = 'X')
# now going to code this as a function in R with theta as a function of beta
    and sigma
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)
}

# NOW (that we have the function coded into R), I will do the regression:
linear.MLE <- optim(fn = linear.lik, par=c(1,1,1), hessian=TRUE, y=data$y,
                    X=cbind(1, data$x), method ="BFGS")
```

This OLS regression returned the following parameters:

```r
linear.MLE$par
# returns an intercept = 0.1398; beta = 2.7266; and sigma estimate = -1.4391
```

Then, I checked my estimates from the OLS regression by running a lm regression using the following code:

```r
lm_regression <- lm(data$y ~ data$x)
```

Which output the following summary statistics:

```
Call:lm(formula = data$y ~ data$x)
Residuals:
Min      1Q  Median      3Q     Max
 -3.1906 -0.9374 -0.1665  0.8931  4.8032

Coefficients:
           Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.13919    0.25276   0.551    0.582
data$x       2.72670    0.04159  65.564   <2e-16 ***---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.447 on 198 degrees of freedom
Multiple R-squared:  0.956,Adjusted R-squared:  0.9557
F-statistic:  4299 on 1 and 198 DF,  p-value: < 2.2e-16
```

This confirmed that my OLS regression using Quasi-Newton methods and my lm regression output equivalent results, as both output an intercept of 0.139 and a beta (slope) of 2.727. One difference between these two is that we don't get an estimate of sigma squared with the lm() method.