Problem Set 1

Applied Stats II

Due: February 11, 2026

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 Wednesday February 11, 2026. 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:

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

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)))
```

```
KolSmi_test <- function(data) {
  i <- 1:length(data)
  expo <- exp(-((2*i - 1)^2) * pi^2)/(8 * (d_value^2))
  p_value <- (sqrt(2 * pi)/d_value) * sum(expo)
  return(p_value)
}

KolSmi_test(data)
#[1] 0.006626909
ks.test(data, "pnorm")
#D = 0.13573, p-value < 2.2e-16
```

The null hypothesis that is being tested with the Kolmogorov-Smirnov test is that the emperical CDF and the normal CDF have similar theoretical cumulative distribution. The function created above found a p value of 0.0066 which is less than 0.05, therefore, we reject the null hypothesis that they have a similar distribution. Below that result, is the built in R function that calculates the d value and the p value of this function. The d value is similar to that manually calculated above, however, the p value differs greatly. Ultimately, the p values are still less that 0.05 which allows the null hypothesis to be rejected.

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)

new_raph <- function(outcome, input, parameter) {
```

```
  residu <- data$y - (parameter[1] + parameter[2]*input)
  sum_resud <- sum(residu^2)
}
results_nr <- optim(fn=new_raph, outcome=data$y, input = data$x, par=c(0,1),
    method="BFGS")
results_nr$par
#  results_nr$par
#[1] 0.139187 2.726699

coef(lm(data$y~data$x))
#(Intercept)       data$x
#0.1391874    2.7266985
```

To create this function, I put in the parameters discussed in the tutorials (outcome, input, and parameter). Next step was to import the basic linear equation and add the manipulation that Newton-Raphson does in order to get its results. Lastly, using the optim function which finds the optimized parameters given the type of algorithm specified. Finally, checking my results against the lm function, I fould that I got the same (just rounded slightly different) intercept and slope.