# 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 $i$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:

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

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).

**My Answer:**

Utilizing the Kolmogorov-Smirnoff test, 1,000 Cauchy random variables will be tested against 1,000 normally distributed random variables. The null hypothesis states that our empirical data (Cauchy data) come from a normal distribution and that the maximum absolute difference between distributions is equal to 0. The alternative hypothesis states that the empirical data do not come from a normal distribution and that the maximum absolute difference between the two distributions is greater than 0.

$$H_0 : \text{D - } d_{obs} = 0, \; H_a : \text{D - } d_{obs} > 0.$$

In order to test my hypotheses, I created a function in R that determines the test statistic and p-value for comparing an empirical set of data to a normal reference distribution. As the empirical CDF is a step-wise function, the calculation for the test statistic is most accurately completed when considering the gap between the two distributions at each step and just before the step occurs. The value d_plus is the maximum distance at each step, which is how much the ECDF exceeds the theoretical CDF. The value d_minus is the maximum distance between the theoretical CDF and ECDF just prior to the step. 1/length(data) is taken from each ECDF data point prior to each step, as that is the amount that the ECDF increases at each step. I chose to utilize a line of code cited in Marsaglia, Tang, and Wang (2003), that efficiently determines the p-value given a sample larger than 99, as well as a value of D*D*n that is larger than 3.76, both of which apply to the data we have.

```r
# Function
ks_test <- function(data) {
  n <- length(data)
  k <- 1:n

  # Create empirical distribution of observed data
  ECDF <- ecdf(data) # Empirical distribution function
  Fn <- ECDF(data)

  # Create theoretical distribution (normal distribution of data)
  F0 <- pnorm(data)

  # Generate the test statistic by comparing the similarity between the
  # empirical  and theoretical CDFs
  d_plus <- max(Fn - F0)
  d_minus <- max(F0 - (Fn - 1/n))
  D <- max(d_plus, d_minus)

```

```
  # Scale D
  s <- n*D^2

  # Calculate the p-value
  if (s > 7.24 || (s > 3.76 && n > 99)) {
    p_value <- 2 * exp(-(2.000071 + .331/sqrt(n) + 1.409/n) * s)
  } else {
    return(message("n and/or s are too"))
  }

  # Print results
  print(paste("P-value = ", p_value))
  print(paste("D = ", D))
}
```

Implementing my function on 1,000 Cauchy random variables to test my hypothesis, and comparing this to the built-in ks.test() function in R to ensure my function is accurate.

```
# Set seed to obtain reproducible values
set.seed(123)

# Generate data using 1000 samples from a Cauchy distribution
data <- rcauchy(1000, location = 0, scale = 1)

# Execute functions and compare my function with the ks.test() function
ks_test(data)
ks.test(data, "pnorm")
```

**ks_test(data):** P-value = 1.60e-16; D = 0.136

**ks.test(data):** P-value = 2.22e-16; D = 0.136

These values confirm that my function accurately completes the Kolmogorov-Smirnoff test. There is only a small variation in the p-value generated using my function and the ks.test() function, that could be attributed to approximations made. The small p-value also allows for the conclusion that the null hypothesis can be rejected. This means that the empirical data do not come from a normal distribution and the maximum absolute difference between the two distributions is significantly greater than 0. This can be confirmed by the fact that the Cauchy distribution is not a normal distribution.

# 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
# Set Seed
set.seed (123)

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

# Establish y
data$y <- 0 + 2.75*data$x + rnorm(200, 0, 1.5)
```

**My Answer:**

In order to estimate an OLS regression in `R` that uses the Newton-Raphson algorithm, an OLS likelihood function needs to be established. This likelihood function will measure the likelihood of observing the outcome at given parameter values (beta and sigma), allowing for the maximum likelihood to be found, and subsequently the optimal parameter values. In order to create this function in R, the n is established as the number of columns in the input, which are the number of columns in the model matrix, equivalent to the number of parameters to be determined. The beta object will then take the 1:n values in the parameter and use them as the starting parameter values. The sigma is established as the exponentiated value of the final parameter, which is the error. In order to not introduce NA values due to taking the square root of the final parameter sigma, exponentiation was used. The log-likelihood values for the normal density distribution are determined using the dnorm function, taking the outcome, as well as the predicted means of each observation, calculated by using matrix multiplication on the input and beta matrices. Sigma is the standard deviation of the distribution. A negative is used prior to the summation of the likelihood values, as the optim() function takes the minimum of the likelihood function to find the optimal parameter.

```r
# OLS Likelihood Function
ols_likelihood <- function(parameter, outcome, input) {
  n <- ncol(input) # Establish n
  beta <- parameter[1:n] # Starting values for beta
  sigma <- exp(parameter[1+n]) # Standard deviation using the error

   # Take the sum of normal density values for each data point
  -sum(dnorm(outcome, input %*% beta, sigma, log=TRUE))
}
```

The optim() function is subsequently utilized, implementing the ols_likelihood function, with the outcome being the y-values of the data, and the input being a matrix consisting of a

column of 1's and a column of the x-values of the data. The initial values of the parameters are set at 1, hessian is set to TRUE to return a hessian matrix, and the method used is BFGS, which is the Newton-Raphson method.

```
# Run likelihood function in optim
results_norm <- optim(fn = ols_likelihood,
                      outcome = data$y,
                      input = cbind(1, data$x),
                      par = c(1,1,1), # Initial values of parameters
                      hessian = TRUE,
                      method = "BFGS")
```

Finally, the parameter results using MLE and the Newton-Raphson method can be compared to the results using the lm() function.

```
# Run lm()
lm_result <- lm(data = data, data$y ~ data$x)

# Compare results
results_norm$par
lm_result
```

```
--- results_norm$par ---
[1] 0.1392 2.7267

--- lm Results ---
 Coefficients:
(Intercept)       data$x
   0.1392          2.7267
```

As seen above, using the lm() function and estimating an OLS regression using maximum likelihood and the Newton-Raphson algorithm result in the same parameter values. This is due to the fact that under a normal distribution, and specifcally with normally distributed error terms, finding the maximum log-likelihood estimates for the parameters reduces down to the same mathematical formula as determining the parameters based on minimizing the least squares estimates.