

# 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).

As a hint, you can create the empirical distribution and theoretical CDF using this code:

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

# Solution - Problem 1

The function below translates the formula for the p-value calculated from the Kolmogorov–Smirnov CDF into R code. Since each term in the final sum decreases as its index  $k$  increases (because the terms are of the form  $e$  to a negative power and  $e$  is greater than 1), raising them to increasingly negative powers causes the terms to converge to 0. To ensure that the function has a stopping point (and does not run infinitely), I set a `max_k` value. The chosen value of 1000 is arbitrary, but sufficiently large to ensure that any additional terms beyond this point would not substantially change the resulting p-value and its interpretation.

As the individual terms are calculated, they are stored in a vector, after which their sum is computed. This sum is then used in the final formula that returns the desired p-value.

```
1 set.seed(123)
2 # generate data
3 data <- rcauchy(1000, location = 0, scale = 1))
4 # create empirical distribution of observed data
5 ECDF <- ecdf(data)
6 empiricalCDF <- ECDF(data)
7 # generate test statistic
8 D <- max(abs(empiricalCDF - pnorm(data)))
9
10 # max_k is an arbitrary value
11 # as k increases the term gets closer to 0
12 ks_pvalue <- function(d, max_k = 1000) {
13   k <- 1:max_k
14   # save individual terms in a vector
15   terms <- exp(-((2*k - 1)^2 * pi^2) / (8*d^2))
16   # sum the elements of the vector
17   s <- sum(terms)
18   # implement final p-value formula
19   p_value <- sqrt(2*pi)/d * s
20   return(p_value)
21 }
22
23 p <- ks_pvalue(D)
24 p
```

```
[1] 5.652523e-29
```

The obtained p-value is much smaller than the 0.05 significance threshold, so we can reject the null hypothesis. Since this is a Kolmogorov–Smirnov test, the null hypothesis is that the data were sampled from the same distribution as the reference distribution. This result checks out, given that our empirical distribution consists of 1,000 Cauchy random variables and the reference distribution is normal. Because the Cauchy distribution is different from the normal distribution (it can have heavier tails), the conducted KS test detects this dissimilarity and allows us to reject the null hypothesis.

## 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.

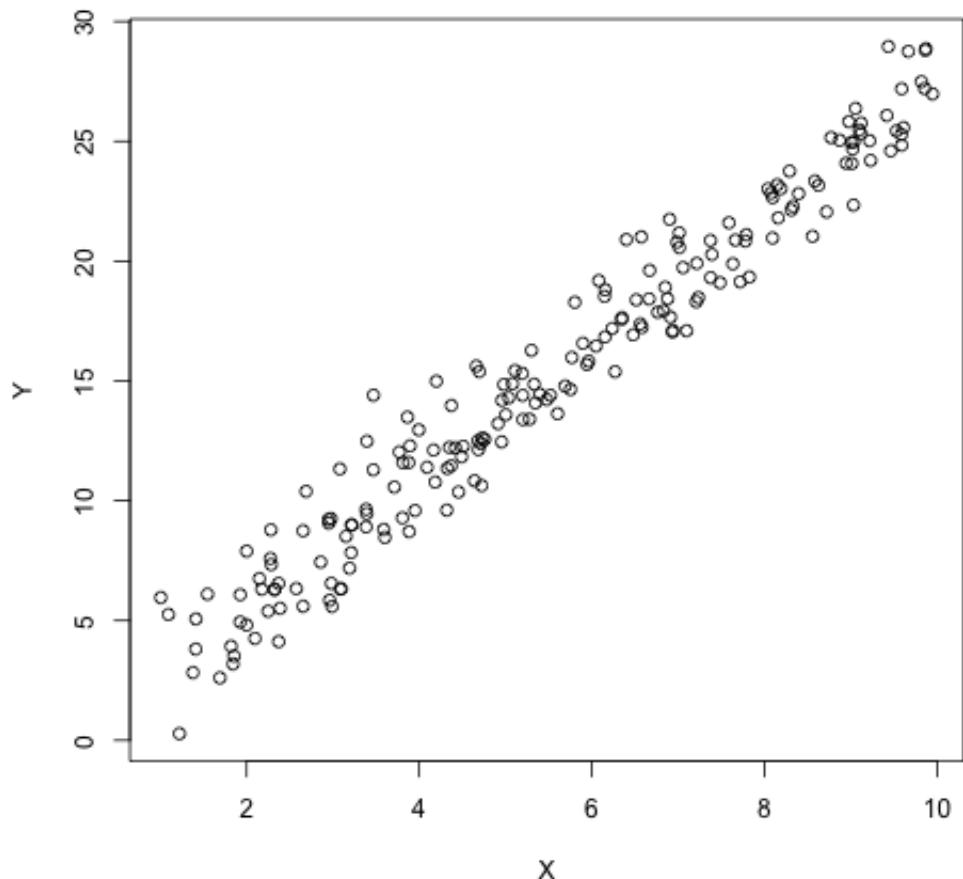
```
1 data <- data.frame(x = runif(200, 1, 10))
2 data$y <- 0 + 2.75 * data$x + rnorm(200, 0, 1.5)
```

## Solution - Problem 2

Before running the regression, we will take a look at the data to make sure the model choice makes sense:

```
1 plot(data$x, data$y, ylab = "Y", xlab = "X")
```

Figure 1: scatterplot of X and Y



```

1 # implement the log of the likelihood function
2 linear.lik <- function(theta, y, X) {
3   n <- nrow(X) # no. observations
4   k <- ncol(X) # no. independent variables (+ intercept)
5   beta <- theta[1:k] # coefficients
6   sigma2 <- theta[k+1]^2
7   e <- y - X %*% beta # residuals
8   logl <- -.5*n*log(2*pi) -.5*n*log(sigma2) - ((t(e) %*% e) / (2*sigma2))
9   return (-logl) # negative because optim finds the minimum
10 }
11
12 # maximising the likelihood function calculated above
13 linear.MLE <- optim(fn = linear.lik, par = c(0,1,1), hessian = TRUE,
14 y = data$y, X = cbind(1, data$x), method = "BFGS")
15
16 # look at the parameters obtained
17 cat("Intercept:", linear.MLE$par[1], "\n",
18     "Slope:", linear.MLE$par[2], "\n")

```

Intercept: 0.13917  
Slope: 2.726695

```

1 lm_model <- lm(y ~ x, data = data)
2 summary(lm_model)

```

Coefficients:  
Estimate Std. Error t value Pr(>|t|)  
(Intercept) 0.13919 0.25276 0.551 0.582  
x 2.72670 0.04159 65.564 <2e-16 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

The main difference between the MLE and OLS approaches lies in the assumptions they make. In the MLE framework, we have to assume a specific probability distribution for the outcome variable. For a linear model, we assume a normal distribution with mean  $X\beta$  and variance  $\sigma^2$ . The MLE approach then estimates the coefficients  $\beta$  and the variance  $\sigma^2$  by maximising the likelihood function, so by finding the values that make the observed data most probable. This is done by constructing the log-likelihood function for  $\beta$  and  $\sigma$  and optimising it using the Newton-Raphson algorithm.

In contrast, the ordinary least squares (OLS) method, implemented in the `lm` function, estimates  $\beta$  by minimising the sum of squared residuals. While OLS does not require a specific distribution for the outcome, it does assume that the errors are normally distributed. For a linear model that fulfils this assumption, OLS is equivalent to MLE. This explains why the coefficients obtained using `lm` and those obtained by optimising the log-likelihood function using `optim` are very close in value, as we expected.