Problem Set 1

Hanyu Li (25346841)

February 10, 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)))

```

Answer

First, we defined the function `KST` to return the test statistic D and calculate the p-value.

```
set.seed(123)
# define the Kolmogorov-Smirnov test function
KST <- function(data) {
# create empirical distribution of observed data
  ECDF <- ecdf(data)
  empiricalCDF <- ECDF(data)
# define theoretical distribution
  theoretical_dis <- pnorm(data, 0, 1)
# generate test statistic
  D <- max(abs(empiricalCDF -theoretical_dis))
# generate p value
  sum_item <- 0
  for ( k in 1:100) {
    num <- -((2*k-1) ^ 2 * pi^2)
    den <- 8 * D^2
    item <- exp(num / den)
    sum_item <- sum_item + item
  }
  p_value <- (sqrt(pi * 2)/D) * sum_item
  cat("Test statistic: ", D, "\n")
```

```
  cat("P-value: ", p_value, "\n")
  }
```

Second, test data from a Cauchy distribution was generated to perform the test. We set the significance level at $\alpha = 0.05$. The hypotheses are shown below:

- H_0: The distribution of the test data (empirical statistics) matches the normal distribution (queried theoretical distribution).
- H_a: The distribution of the test data (empirical statistics) shows dissimilarity to the normal distribution (queried theoretical distribution).

Then, the Kolmogorov-Smirnov test was run, yielding a test statistic of $D = 0.1347281$ and a p-value of 5.65×10^{-29}.

```
# generate test statistic
set.seed(123)
test_data <- rcauchy(1000, location = 0, scale = 1)
# run the test
test <- KST(test_data)
```

From the results, we can say that: since the p-value is far below our specified threshold of 0.05, we have enough evidence to reject the null hypothesis and conclude that the distribution of the test data shows dissimilarity to the 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.

```
  for ( k in 1:100) {
    num <- -((2*k-1) ^ 2 * pi^2)
    den <- 8 * D^2
```

Answer

According to the Newton-Raphson algorithm, to estimate the OLS model using MLE, we proceed with the following steps:

First, we assume the data follow a normal probability distribution, with mean $x\beta$ and variance σ^2, i.e., $y \sim N(x\beta, \sigma^2)$.

Next, we express it as the equivalent probability density function:

$$f(y_i \mid \beta, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - x_i\beta)^2}{2\sigma^2}}$$

To calculate the likelihood function, we take the product of probabilities to obtain the likelihood function:

$$L = (2\pi)^{-n/2} \cdot (\sigma^2)^{-n/2} \cdot e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - x_i\beta)^2}$$

To facilitate calculation in R, we take the logarithm of the likelihood function to obtain the log-likelihood form:

$$\log(L) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - x_i\beta)^2$$

Due to the computational characteristics of R, we finally return the negative log-likelihood.

```r
# define log-likelihood function
linear.lik <- function(theta, y, x) {

  n <- length(x)
  inter <- theta[1]
  beta <- theta[2]
  sigma <- abs(theta[3])
  y_hat <- inter + beta*x
  error <- y - y_hat
  sigma_2 <- sigma^2

  log_lik <- -0.5*n*log(2*pi)-0.5*n*log(sigma_2)-(sum(error^2)/(2*sigma_2)
   )
  return(-log_lik)
}
```

Next, we randomly assign initial values to the intercept, coefficients, and sigma in the original function, and apply the Newton-Raphson algorithm for iteration. The final model parameters are:

- final intercept: 0.1391874
- final_beta: 2.726699
- final_sigma: 1.439545

```r
# give a random guess for parameters
set.seed(123)
init_theta <- runif(3,0,5)

# uses the Newton-Raphson algorithm
lik_result <- optim(par = init_theta,
                    fn = linear.lik,
                    y = data$y,
                    x = data$x,
                    method = "BFGS",
                    hessian = TRUE)

cat("final intercept:", lik_result$par[1], "\n",
    "final_beta:     ", lik_result$par[2], "\n",
    "final_sigma:    ", abs(lik_result$par[3]), "\n")
```

Finally, we run the OLS regression model for cross-validation. The regression results are as follows:

```r
# compare with OLD model result
OLS_ml <- lm(y ~ x, data = data)
summary(OLS_ml)
```

Table 1: OLS Regression Results

	Dependent variable:
	y
x	2.727***
	(0.042)
Constant	0.139
	(0.253)
Observations	200
R^2	0.956
Adjusted R^2	0.956
Residual Std. Error	1.447 (df = 198)
F Statistic	4,298.687*** (df = 1; 198)
Note:	*p<0.1; **p<0.05; ***p<0.01

We find that the intercept and coefficients calculated via Maximum Likelihood Estimation are consistent with the results from directly running the OLS regression model. The intercept is approximately 0.139, and the independent variable coefficient is approximately 2.726.