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).

Answer 1

Below is an R function that implements this test where the reference distribution is normal, followed by an application to 1,000 Cauchy random variables.

```
ks_test_normal <- function(data, mean = 0, sd = 1) {
  ###############################################################
  # One-Sample Kolmogorov{Smirnov Test
  # -------------------------------------------------------------
  # This function tests whether the empirical distribution of
  # the observed data matches a Normal(mean, sd^2) distribution.
  #
  # H0: The data follow a Normal(mean, sd^2) distribution
  # H1: The data do NOT follow that Normal distribution
  #
  # The test statistic:
  # D = max | F_n(x) - F(x) |
  # where:
  #   F_n(x) = empirical CDF
  #   F(x)   = theoretical Normal CDF
  ###############################################################

  # Sample size
  n <- length(data)

  # Empirical CDF
  ECDF <- ecdf(data)
  empiricalCDF <- ECDF(data)

  # Theoretical Normal CDF
  theoreticalCDF <- pnorm(data, mean = mean, sd = sd)

  # KS statistic
  D <- max(abs(empiricalCDF - theoreticalCDF))

  # Asymptotic p-value approximation:
  # P(D_n >= d)  2 * sum_{k=1}^ (-1)^(k-1) * exp(-2 k^2 D^2 n)
  k <- 1:100
  p_value <- 2 * sum((-1)^(k-1) * exp(-2 * k^2 * D^2 * n))

  # Ensure p-value is in [0,1]
  p_value <- max(min(p_value, 1), 0)

  return(list(
    D_statistic = D,
    p_value = p_value
  ))
}

set.seed(123)

# Generate 1000 Cauchy observations
data <- rcauchy(1000, location = 0, scale = 1)

# Run KS test against standard normal
result <- ks_test_normal(data, mean = 0, sd = 1)

result
```

The resulting test statistic and p-value are:

$$D_{\text{statistic}} = 0.1347281, \qquad p\text{-value} = 3.425363 \times 10^{-16}.$$

The null hypothesis is that this sample comes from a N(0,1) distribution, while the alternative is that it does not. Since the p-value is effectively zero at any conventional significance level (such as 0.05 or 0.01),I can reject the null hypothesis. The empirical distribution of these 1,000 Cauchy observations is clearly inconsistent with the standard normal distribution, which is exactly what we would expect given the much heavier tails of the Cauchy distribution compared to the normal.

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.

Answer 2

We now estimate an OLS regression in R using the BFGS quasi-Newton algorithm and show that we obtain results equivalent to `lm`. We first generate the data

$$y_i = 2.75x_i + \varepsilon_i, \quad \varepsilon_i \sim N(0, 1.5^2), \quad x_i \sim \text{Unif}(1, 10),$$

for $i = 1, \ldots, 200$.

The OLS objective function is the sum of squared residuals (SSR):

$$\text{SSR}(\beta) = \sum_{i=1}^{n} (y_i - \beta x_i)^2.$$

Instead of using the closed-form OLS solution, we numerically minimize this function using BFGS.

```r
set.seed(123)

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

# OLS objective function (sum of squared residuals)
ols_objective <- function(beta, x, y) {
  sum((y - beta * x)^2)
}

# Estimate using BFGS
optim_results <- optim(
  par = 0,
  fn = ols_objective,
  x = data$x,
  y = data$y,
  method = "BFGS"
)

# Extract estimated coefficient from BFGS
beta_bfgs <- optim_results$par

# Estimate using lm() with no intercept
lm_results <- lm(y ~ 0 + x, data = data)

# Extract coefficient estimate from lm()
beta_lm <- coef(lm_results)

# Compare estimates
beta_bfgs
beta_lm
```

The numerical estimates from the two methods are:

$$\hat{\beta}_{\text{BFGS}} = 2.74764, \qquad \hat{\beta}_{\text{lm}} = 2.74764.$$

For the OLS regression, the BFGS-based estimate and the lm() estimate are both 2.74764 matching to numerical precision. This shows that minimizing the sum of squared residuals with the BFGS quasi-Newton algorithm recovers the same coefficient as the closed-form OLS solution used internally by lm(). Since the SSR objective in this simple linear, no-intercept model is a quadratic and strictly convex function of , it has a unique global minimum. Any well-behaved gradient-based optimizer such as BFGS is guaranteed to converge to that same minimum, provided it is started from a reasonable initial value.