Problem Set 1

Applied Stats II

Due: February 11, 2024

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 Sunday February 11, 2024. 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)))
```

```
# Function to calculate the p-value using the fix formula
ks_p_value <- function(D) {
  # fist part of the formula
  pi_sqrt_2 <- sqrt(2 / pi)

  # accumulate the sum of k
  sum_val <- 0
  k_max <- 500
  for (k in 1:k_max) {
    sum_val <- sum_val + exp(-2 * (2 * k - 1)^2 * pi^2 / (8 * D^2))
  }

  # Calculate the p-value using the accumulated sum
  p_value <- pi_sqrt_2 * sum_val / D

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

  return(p_value)
}

# Function to perform the Kolmogorov-Smirnov test
ks_test <- function(data) {

  # Create empirical distribution of observed data
  ECDF <- ecdf(data)
  empiricalCDF <- ECDF(data)

  # Generate test statistic D
  D <- max(abs(empiricalCDF - pnorm(data)))

  # Calculate the p-value using the  formula function in last step and D
```

```r
  p_value <- ks_p_value(D)

  # Return the test statistic and p-value
  return(list(D = D, p_value = p_value))
}

set.seed(123)
# Generate 1,000 cauchy random variables
data <- rcauchy(1000, location = 0, scale = 1)

# Perform the Kolmogorov-Smirnov test
ks_test_result <- ks_test(data)

# Print the results
print(ks_test_result)
# Test Statistic (D): 0.1347281  p_value: 5.466419e-59

# valid with ks function
ks_result <- ks.test(data, "pnorm")

# Print the results
print(ks_result)
# Test Statistic (D) = 0.13573, p-value = 2.22e-16
```

Interpret result:

Both the custom K-S test and the built-in K-S test in R showed the same statistical conclusion: the sample data, generated from a Cauchy distribution, do not follow a normal distribution. Because The p-value is extremely small ($5.466419 \times 10^{-59}, 2.22 \times 10^{-16}$, though the p-value from build in ks test is not as small as the p-value from the custom test)

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
# data set generated
set.seed (123)
data <- data.frame(x = runif(200, 1, 10))
data$y <- 0 + 2.75*data$x + rnorm(200, 0, 1.5)

# step 1: function to calculate sum of square residual (ssr)
#data$x - Predictor, data$y - Response

ssr <- function(params) {
  predicted_values <- params[1] + params[2] * data$x
  residuals <- data$y - predicted_values
  sum(residuals^2)
}

# step2: use the optim() function with the BFGS method to
#find the parameter estimates that minimize the sum of squared residuals

estimate_params <- c(0, 0) #  guesses for intercept and slope
optim_results <- optim(par = estimate_params, fn = ssr, method = "BFGS")
bfgs_params <- optim_results$par

#For comparison, fit an OLS model using the lm() function
#Estimate Parameters using lm()
lm_model <- lm(y ~ x, data = data)
lm_params <- coef(lm_model)

print(bfgs_params)
#BFGS estimate for the intercept: 0.1391778  slope: 2.7267000

print(lm_params)
#lm() estimate for the intercept: 0.1391874 slope: 2.7266985
```

Interpret result:

By comparing findings from the output of lm and bfgs, there is a close match between the parameters estimated by the traditional lm() function and through using the BFGS method. The estimate intercept and slope values are nearly the same, with only a tiny difference in the last few decimal places. In general, both BFGS optimization method and lm() are accurate for estimating the parameters of an OLS regression, both methods able to calculates the best possible line that represents the relationship between X and Y in this data