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)))
```

Answer:

```

set.seed(123)

  # Define the Kolmogorov-Smirnov test function
  ks_test <- function(data){

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

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

  # Calculate p-value using Kolmogorov-Smirnov CDF
  n <- length(data)
  p_value <- 1 - pnorm(sqrt(n) * D)

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

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

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

```

statistic [1] 0.1347281
p[1] 1.019963e-05

This p-value represents the probability of observing a test statistic as extreme as D under the null hypothesis that the empirical distribution matches the theoretical 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.

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

Answer:

```

# Define the objective function for OLS
ols_objective <- function(beta, x, y) {
  y_pred <- beta[1] + beta[2] * x
  residuals <- y - y_pred
  sum(residuals^2)}

# Use BFGS algorithm to minimize the objective function
initial_guess <- c(0, 0)  # Initial guess for coefficients
fit_bfgs <- optim(par = initial_guess, fn = ols_objective, x = data$x, y =
    data$y, method = "BFGS")

# Extract coefficients from BFGS result
coefficients_bfgs <- fit_bfgs%$%par

# Compare with lm
lm_result <- lm(y ~ x, data = data)
coefficients_lm <- coef(lm_result)

# Print results
cat("Coefficients from BFGS (Newton-Raphson):\n")
print(coefficients_bfgs)

cat("\nCoefficients from lm (Ordinary Least Squares):\n")
print(coefficients_lm)
```

print(coefficients[1] 0.1391778 2.7267000

print(coefficients(Intercept) x 0.1391874 2.7266985

This represents the estimated value of the dependent variable (y) when the independent variable (x) is zero. In this case, the intercept is approximately 0.1392.

x: Represents the estimated change in the dependent variable (y) for a one-unit increase in the independent variable (x). In this case, the coefficient for x is approximately 2.7267,

These coefficients are consistent with those obtained from the BFGS optimization algorithm