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:

$$p(D \leq x) \frac{\sqrt{2\pi}}{x} \sum_{k=1}^{\infty} e^{-(2k-1)^2 \pi^2 / (8x^2)}$$

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)))
```

My answer:

```
ks_test_normal <- function(observed_data, seed = 123) {
  set.seed(seed)

  # Generate Cauchy random variables
  cauchy_data <- rcauchy(1000, location = 0, scale = 1)

  ECDF <- ecdf(observed_data)
  empirical_CDF <- ECDF(observed_data)

  theoretical_CDF <- pnorm(observed_data)

  # Calculate test statistic
  D <- max(abs(empirical_CDF - theoretical_CDF))

  # Calculate p-value using Kolmogorov-Smirnov CDF
  p_value <- sqrt(2 * pi) / D * sum(exp(-(2 * (1:1000) - 1)^2 * pi^2 / (8 * D
    ^2)))

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

# Example usage:
observed_data <- rcauchy(1000, location = 0, scale = 1)
result <- ks_test_normal(observed_data, seed = 123)
print(result)
```

```
## $test_statistic
## [1] 0.1398682
##
## $p_value
## [1] 7.338799e-27
```

I use this function, ks_test_normal to take observed data as input, generates Cauchy random variables, computes the empirical distribution, and then calculates the test statistic and

p-value.In this question, p-value calculation involves an infinite sum, so it might require numerical approximation methods.

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
# Set the random seed for reproducibility
set.seed(123)

# Create a data frame with two columns, where 'x' is generated from a uniform distribution in the range [1, 10],
# and 'y' is generated based on a linear relationship with added noise
data <- data.frame(x = runif(200, 1, 10))
data$y <- 0 + 2.75 * data$x + rnorm(200, 0, 1.5)

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

# Use the BFGS optimization algorithm to fit the model with initial parameters set to c(0, 1)
result_optim <- optim(par = c(0, 1), fn = ols_obj_function, x = data$x, y = data$y, method = "BFGS")

# Get the parameter estimates from the optimization results
coef_optim <- result_optim$par

# Fit a linear model using the lm function, y ~ x
model_lm <- lm(y ~ x, data = data)

# Get the parameter estimates from the lm function
coef_lm <- coef(model_lm)

# Print the coefficients obtained using the BFGS optimization algorithm
cat("Coefficients using BFGS optimization:\n")
print(coef_optim)

# Print the coefficients obtained using the lm function
cat("\nCoefficients using lm function:\n")
print(coef_lm)
```

The result of compiling:

```
## Coefficients using BFGS optimization:
```

```
print(coef_optim)
```

```
## [1] 0.139187 2.726699
```

```
cat("\nCoefficients using lm function:\n")
```

```
##
## Coefficients using lm function:
```

```
print(coef_lm)
```

```
## (Intercept)           x
##   0.1391874   2.7266985
```

I use this function,$ols_obj_function$ that computes the sum of squared residuals for given coefficients. The optim function is then used to minimize this objective function with the BFGS algorithm. The results are compared with the lm function.