Problem Set 1

Applied Stats II
Dan Zhang 23335541

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 d 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:
The whole procedure will be including 4 parts: generate ECDF and reference CDF, calculate D statistic, calculate p value and use R function to check the results.

First, generate Cauchy distributed data as the observated data, and then use ecdf function to create ECDF for observed data. Next, Use pnorm( ) to get the CDF of the normal distribution for observed data. As there is not specified, so I use the standard normal distribution. Set seed for reproducibility.

```
# Set seed for reproducibility
set.seed(123)

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

# Sort the data
sorted_data <- sort(data)

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

Calculate the D statistic using the formula provided. The result is D=0.1357281.

```
# Calculate the D statistic
n <- length(data)
D <- max(c(abs((1:n)/n - theoretical_CDF), abs(theoretical_CDF - (0:(n-1))/n))
    )
```

Calculate p value. The formula provided above is based on infinite series. I truncate it after a reasonable number of terms. Here I choose 1000 terms for demonstration. This number might need to be increased for higher accuracy. The result is p-value=1.52187e-28.

```
# Calculate the p-value for the D statistic
p_value_calculate <- function(D) {
  # As the series is infinite, here choose k_max = 1000 to truncate it.
  k <- 1:1000
```

```r
  sum_exp <- sum(exp(-((2*k - 1)^2 * pi^2) / (8 * D^2)))
  p_value <- (sqrt(2 * pi) / D) * sum_exp
  # Make sure p-value not exceed 1
  min(p_value, 1)
}

# Calculate the p-value for the computed D statistic
p_value <- p_value_calculate(D)
```

Finally, use the KS test in R to check the results.

```r
# Do KS test to double check
# Compare the Cauchy distributed data against a normal distribution with mean
    and sd estimated from the data
results <- ks.test(data, "pnorm", mean = 0, sd = 1)
```

The KS test results are showed below:

```
Asymptotic one-sample Kolmogorov-Smirnov test

data:  data
D = 0.13573, p-value < 2.2e-16
alternative hypothesis: two-sided
```

As we can see from the results that the p-value is significantly smaller than 0.05. So we can reject the null hypothesis that the distribution of the empirical Cauchy data matches normal distirbuted.

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.

Answer:
The whole procedure will be 2 steps:

Step 1: Use BFGS to estimate the OLS regression.

Step 2: Use lm to get the equivalent results First, set seed and create data.

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

Then estimate OLS model using BFGS algorithm. I use optim function in R and specify the BFGS algorithm to estimate model parameters. This includes towrite a loss function, so I define the residual sum of squares as the loss function, and then use optim function to minimize this loss function.

```r
# Estimate OLS model using optim()
# Define the loss function as the sum of squared residuals
loss_function <- function(params, data) {
  with(data, sum((y - (params[1] + params[2] * x))^2))
}

# Initial parameter guesses
initial_params <- c(intercept = 0, slope = 1)
# Use BFGS algorithm via optim()
optim_results <- optim(
  par = initial_params,
  fn = loss_function,
  data = data,
  method = "BFGS",
  hessian = TRUE
)

# Convert optim results to tidy format
optim_coefs <- setNames(optim_results$par, c("(Intercept)", "x"))
```

Next, use lm in R to estimate OLS regession and compare the results for two models.

```r
# Estimate OLS model using optim()
# Define the loss function as the sum of squared residuals
loss_function <- function(params, data) {
  with(data, sum((y - (params[1] + params[2] * x))^2))
}

# Initial parameter guesses
initial_params <- c(intercept = 0, slope = 1)
# Use BFGS algorithm via optim()
optim_results <- optim(
  par = initial_params,
  fn = loss_function,
  data = data,
  method = "BFGS",
  hessian = TRUE
)

# Convert optim results to tidy format
optim_coefs <- setNames(optim_results$par, c("(Intercept)", "x"))
```

The results are as followed:

Table 1: Estimate OLS regression using lm

(Intercept)	x
0.139	2.727

Table 2: Estimate OLS regression using BFGS algorithm

	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

As we can see from the results, the estimate results for OLS regression by using BFGS algorithm and lm are the same.