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:

```r
# create empirical distribution of observed data
ECDF <- ecdf(data)
empiricalCDF <- ECDF(data)
# generate test statistic
D <- max(abs(empiricalCDF - pnorm(data)))
```

First, let's set up our null hypothesis and alternative hypothesis:

we want to know whether the empirical distribution (basically this is a 1,000 Cauchy random variables, generated by `R`) matches the queried theoretical distribution (in this question, this is a normal distribution). So, we can set up our H_0 and H_1.

H_0: The empirical distribution **follows** a normal distribution.
H_1: The empirical distribution **does not follows** a normal distribution.

```r
######################
# Problem 1
######################

# The function of Kolmogorov-Smirnov test with normal distribution
ks_normal <- function(data) {
  # This function is used to test if a distribution follows normal distribution.

  # Parameters:
  # - data: The empirical distribution

  # Returns:
  # - D: largest absolute difference between the two distribution
  # - p_value: Probability that the null hypothesis is correct

  # create empirical distribution of observed data
  ECDF <- ecdf(data)
  empiricalCDF <- ECDF(data)
  # generate test statistic
  D <- max(abs(empiricalCDF - pnorm(data)))
  # R can not deal with infinite, so I use 1000 instead
  n <- 1000
  # calculate p-value
  p_value <- sqrt(2 * pi) / D * sum(exp(-((2 * (1:n) - 1)^2 * pi^2) / (8 * D^2)))
  # return results
  return(list(D = D, p_value = p_value))
}
# set seed as 123
set.seed(123)
# generate 1,000 Cauchy random variables
data <- rcauchy(1000, location = 0, scale = 1)
# use function to drive Kolmogorov-Smirnov test
ks_results <- ks_normal(data)
# print the test results
paste("D:", ks_results$D)
paste("p_value:", ks_results$p_value)
```

We got outputs from `R`:

```
[1] "D: 0.13472806160635"
[1] "p_value: 5.65252281681864e-29"
```

We can see that the p-value (= 5.65252281681864e-29) is below the $\alpha = 0.05$ threshold, so we would say that we find sufficient evidence to reject the null hypothesis that the empirical distribution follows a normal distribution.

So, our conclusion is to accept alternative hypothesis H_1:
The empirical distribution **does not follows** a normal distribution.

We can also check answers by using the `ks.test` function in `R`:

```r
# check by ks.test function in R
ks_check <- ks.test(data, "pnorm")
print(ks_check)
```

We got outputs from R:

```
Asymptotic one-sample Kolmogorov-Smirnov test

data:  data
D = 0.13573, p-value = 2.22e-16
alternative hypothesis: two-sided
```

Compared with the function generated D (≈ 0.13573), our manually calculated D (≈ 0.13473) is nearly the same. And both p-value (2.22e-16 & 5.65252281681864e-29) are very small and below the $\alpha = 0.05$ threshold.

Here is the explain of these 2 methods get the similar but not the same results:

- In `ks.test` command, R will use ∞ in formula. But in our handwritten formula, we replaced ∞ with a sufficiently large n, which is 1,000 in this case.

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
#####################
# Problem 2
#####################

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

Draw a scatter plot to see the relationship between the generated x and y.

```r
# Draw a scatter to see the distribution
pdf("q2_plot1.pdf")
q2_scatter <- ggplot(data, aes(x = x, y = y)) +
  geom_point(shape=1, size=2.8, color="#0F4C75") + # set point shape and color
  theme_bw() + # set the cooooolest style
  theme(panel.grid = element_blank()) # no gird
print(q2_scatter)
dev.off()
```

And we get this plot in R:

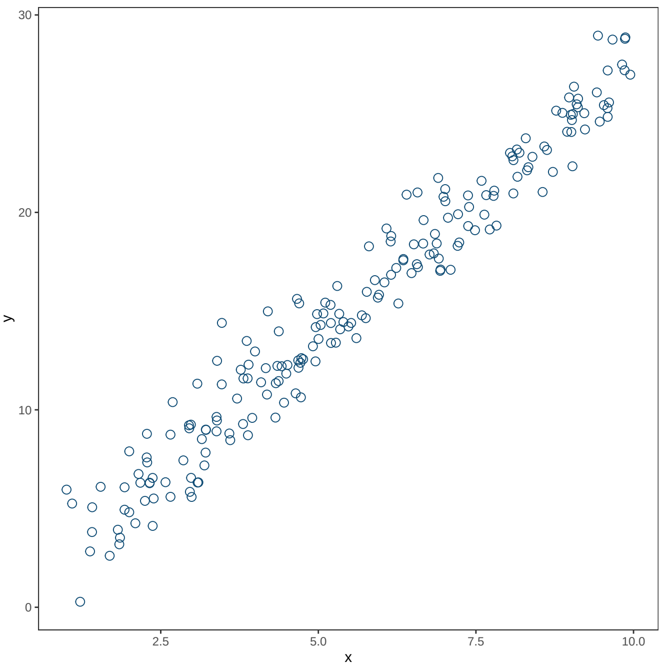

Figure 1: Scatter - Problem 2 data

To use Newton-Raphson algorithm in `R`, firstly define a loss function, secondly set default slope and intercept both as 0 according to the scatter, then use BFGS method in optim function. Here are codes:

```r
# Define OLS loss function
# Loss function is used to measure the differences between
# predicted value and real value.

loss_function <- function(theta, x, y){
  # This function is used to define OLS loss function

  # Parameters:
  # - theta: parameter vector, in this case is the intercept and coefficients.
  # - x: independent variables value, in this case is a matrix.
  # - y: dependent variable value, in this case is a vector.

  # Returns:
  # - loss: the differences between predicted values and observed values.
  # in this case is the sum of squared residuals (SSR).
  predicted <- theta[1] + theta[2] * x # y = a + bx
  loss <- sum((predicted - y)^2) # SSR formula
  return(loss)
}

# Define parameters we need in optim function.
# Optim function can used to minimize or maximize an object function

initial_coef <- c(0, 0) # set the default slope and intercept to 0
method <- "BFGS" # define the method as BFGS

newton_results <- optim(initial_coef,
                loss_function, # use the loss function
                x = data$x, y = data$y, # set the variables values
                method)
```

The slope and coefficient were stored in par vector in newton_results variable, so use this code to print:

```r
# print Newton-Raphson results
cat("In Newton-Raphson algorithm, my results are: \n")
cat("The intercept is:", newton_results$par[1], "\n")
cat("The slope is:", newton_results$par[2], "\n")
```

We got the answers:

```
In Newton-Raphson algorithm, my results are:
The intercept is: 0.138529
The slope is: 2.726599
```

To show the equivalent results with `lm`, I use these codes:

```r
# Finally, let's compare with lm methods.
lm_results <- lm(y ~ x, data)
summary(lm_results)
# print OLS results
cat("In lm methods, my results are: \n")
cat("The intercept is:", lm_results$coefficient[1], "\n")
cat("The slope is:", lm_results$coefficient[2], "\n")
```

We got the answers:

```
In lm methods, my results are:
The intercept is: 0.1391874
The slope is: 2.726699
```

Create a table to see the differences between Newton-Raphson Algorithm and lm Methods:

	Intercept	**Slope**
Newton-Raphson Algorithm	0.138529	2.726599
lm Methods	0.1391874	2.726699

Compared with the Newton-Raphson Algorithm intercept (≈ 0.14) and slope (≈ 2.73), lm methods' intercept (≈ 0.14) and slope (≈ 2.73) is near the same. And we can write our regression formula like this:

$y = 0.14 + 2.73x$

Here is the explain of these 2 methods get the similar but not the same results:

- Due to differences in numerical computation precision, computational methods, and implementation details of the optimization algorithms, minor differences may arise. These differences are typically negligible and do not significantly affect the final model fitting results.