Problem Set 1

Vismante Dringelyte 17341481

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)))
```

I used the hint as a starting point for my function.

```
ks_test <- function(data) {

  ECDF <- ecdf(data)
  empiricalCDF <- ECDF(data)

  D <- max(abs(empiricalCDF - pnorm(data)))
```

Then to calculate the p-value, I translated the formula into R code. I chose to use 1000 as the limit for the function.

```
  # calculate p-value
  n <- length(data)
  p_value = (sqrt(2 * pi)) / D * sum(exp(-((2 * (1:1000) - 1)^2 * pi^2) / (8 *
    D^2)))

  result <- list(
    t_stat = D,
    p_value = p_value
  )

  return(result)
}
```

This gave me these results:

```
  $t_stat
  [1] 0.1347281

  $p_value
  [1] 5.652523e-29
```

The tiny p-value suggests that the data does not follow standard normal 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.

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

As instructed, I used the code above to generate the data. I defined the function for OLS. It calculates the sum of squared residuals for the coefficients `beta` and `x` and `y`.

```r
ols_function <- function(beta, x, y) {
  residuals <- y - (beta[1]+beta[2]* x)
  sum(residuals^2)
}
```

I use the BFGS algorithm to minimise the OLS function, setting the initial guess for the coefficients as `(0,0)` and using the Hessian matrix.

```r
ols_bfgs <- optim(fn = ols_function, par = c(0,0), hessian = TRUE, x = data$x,
    y = data$y, method = "BFGS")


ols_bfgs$par
```

This gives me the coefficients

```
> ols_bfgs$par
[1] 0.1391778 2.7267000
```

Comparing it to the `lm` results:

```r
lm_results <- lm(y ~ x, data = data)
```

They appear to be the same:

Table 1:

	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