Problem Set 1

Tianxin Zhang/21352232/Applied Stats II

Due: February 19, 2023

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 19, 2023. 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)))
```

```
library("stargazer")
set.seed(123)

# create empirical distribution of observed data

data1 <- rcauchy(1000, location = 0, scale = 1)

ECDF <- ecdf(data1)
empiricalCDF <- ECDF(data1)
# generate test statistic
D <- max(abs(empiricalCDF - pnorm(data1)))
D
# test statistic D =  0.1347281

# p-value calculation: method 1:
ks_test1 <- ks.test(data1, "pnorm")
ks_test1$p.value
# p vluae is very close to 0

# p-value calculation: method 2:
# create a randomly distributed data (size: 1000):
data_norm <- rnorm(1000, mean=500, sd=5)

ks_test2 <- ks.test(data1, data_norm)
ks_test2$p.value
# p-value is 0

# Get the ks.test() here: https://www.statology.org/kolmogorov-smirnov-test-r/
```

Conclusion: p-value is roughly 0 by using both methods. Therefore, we can reject the hypothesis that the empirical distribution does not match the queried theoretical distribution.

Note: I don't know why I got the result of 2.2e-16 by method 1, ks.test(data1, "pnorm"), though it is very close to 0; and got an exactly 0 by method 2, ks.test(data1, data2) where data2 is generated by rnorm which gave 1000 randomly distributed data. But basically, the result given by method1 is very close to the result given by method 2.

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)

lm_test <- lm(y ~ x, data = data)
lm_test$coefficients  # roughly 2.727
stargazer(lm_test, title="OLS Results")


linear.lik <- function(theta, y, X){
  n <- nrow(X)
  k <- ncol(X)
  beta <- theta[1 : k]
  sigma2 <- theta[k + 1]^2
  e <- y - X%*%beta
  logl <- -.5*n*log(2*pi) -.5*n*log(sigma2) - ((t(e) %*%
                                  e ) / ( 2 * sigma2 ) )
  return(-logl)}

linear.MLE <- optim(fn=linear.lik, par=c(1, 1, 1), hessian =TRUE, y =data$y, X
   = cbind (1 ,data$x), method = "BFGS")
linear.MLE$par # coefficient is roughly equal to 2.727
stargazer(linear.MLE, title="MLE Results")
```

Conclusion: the coefficients for X generated by both OLS regression and MLE are roughly equal to 2.727

Table 1: OLS Results

	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

Table 2: MLE Results

0.140	2.727	-1.439

Table 3: MLE Results

356.653

Table 4: MLE Results

146	100

Table 5: MLE Results

1