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)))
```

Code for problem 1

```r
# Kolmogorov-Smirnov test function
set.seed(123)
kol_smir_test <- function(sample_size = 1000, location = 0, scale = 1) {
  data <- rcauchy(sample_size, location = location, scale = scale)
  ECDF <- ecdf(data)
  empiricalCDF <- ECDF(data)
  D <- max(abs(empiricalCDF - pnorm(data)))
  print(paste("Kolmogorov-Smirnov test statistic (D) : ", D))
  return(D)
  }
# Calling the function
kol_smir_result <- kol_smir_test()
print(kol_smir_result)
# Calculating the p-value
D <- 0.13472806160635
pvalue_3 <- 1- pnorm(sqrt(n) * D)
pvalue_3 <- 1 - pnorm(sqrt(1000) * 0.13472806160635)
print(pvalue_3)
# doing the ks.test to compare results
ks_result3 <- ks.test(data, "pnorm")
print(ks_result3)
```

Output: The p-value from my function is pvalue-3 = 1.019963e-05 and the p-value from the ks.test is ks-result3 = 2.22e-16. I know mine is still higher than the ks.test but is the closest to zero I got after changing the function a lot. Actual output from ks.test: Asymptotic one-sample Kolmogorov-Smirnov testdata: dataD = 0.13573, p-value = 2.22e-16 alternative hypothesis: two-sided

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)
```

Code

```
# Generating the data
set.seed (123)
data <- data.frame(x = runif(200, 1, 10))
data$y <- 0 + 2.75*data$x + rnorm(200, 0, 1.5)
# defining the function
OLS_obj <- function(beta, x, y) {
  y_esp <- beta[1] + beta[2]*x
  error <- y - y_esp
  return(sum(error^2))
}
# Beta values at first
beta_um <- c(0, 0)
# Optimising with BFGS
bfg_otimo <- optim(par = beta_um, fn = OLS_obj, x = data$x, y = data$y, method
    = "BFGS")
# getting the estimated coefficients
bfg_estcoef <- bfg_otimo$par
# Comparing the coefficients from BFGS with the lm
lm_comp <- lm(y ~ x, data = data)
coef_lm <- coef(lm_comp)
#
print(bfg_estcoef)
print(coef_lm)
```

Output: bfg-estcoef = 0.1391778 , 2.7267000 and coef-lm = (Intercept) = 0.1391874 , x = 2.7266985