Problem Set 1

Applied Stats II

Due: February 11, 2024///Wei Tang 23362496

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)))
```

```
######################
# Problem 1
######################
# generate 1,000 Cauchy random variables
set.seed(123)
data<-rcauchy(1000, location = 0, scale = 1)
# create empirical distribution of observed data
ECDF <- ecdf(data)
empiricalCDF <- ECDF(data)
# generate test statistic
D <- max(abs(empiricalCDF - pnorm(data)))
print(D)
```

Now we get the test statistic: 0.1347281.
Then we should calculate the p-value based on this statistic.

```
# get the length of data
n <- length(data)
# set the parameters
temp_sum=0
pi=3.14159
e=2.71828
# do a for loop according to the formula to calculate the p-value
for (k in 1:n){
  exponent<- -1*((((2*k-1)**2)*((pi)**2))/(8*(D**2)))
  temp_sum=temp_sum+ (e**exponent)
}
# get p-value
p<-(((2*pi)**0.5)/D)*temp_sum
# do a KS test by R
ks_result <- ks.test(data, "pnorm", alternative = "two.sided", exact = FALSE)
ks_result
# Compare the p-value in two results
print(ks_result$p.value)
print(p)
```

Now we get the p-value from KS test:2.220446e-16
and the p-value from manual: 5.653428e-29

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)
# Variable x: Generate 200 random numbers from a uniform distribution
data <- data.frame(x = runif(200, 1, 10))
# Variable y: Generate values based on a linear relationship
data$y <- 0 + 2.75*data$x + rnorm(200, 0, 1.5)

# Compute the negative log-likelihood for OLS
linear.lik <- function(beta, y, X) {
  # Compute the residuals by subtracting the predicted values from the actual
    values
  resid <- y - X %*% beta
  # Compute the sum of squared residuals
  lik <- sum(resid^2)
  # Return the result
  return(lik)
}
# build model by OLS regression using BFGS algorithm
ols_bfgs <- optim(fn=linear.lik,par=c(1,1),hessian=TRUE,y=data$y,X=cbind(1,
    data$x),method="BFGS")
```

Now we build the regression model by BFGS algorithm. Then we check the parameters.

```r
cat("Coefficients using BFGS algorithm:\n")
cat("Intercept:", ols_bfgs$par[1], "\n")
cat("Slope:", ols_bfgs$par[2], "\n")
cat("\n")
```

Coefficients using BFGS algorithm:
Intercept: 0.1391874
Slope: 2.726699

```r
# build linear model
ols_lm <- lm(y ~ x, data = data)
# Check the result
ols_lm
ols_bfgs
```

Now we build the linear regression model. Then we check the parameters.

```r
#show the result of linear model
cat("Coefficients using linear model:\n")
cat(summary(ols_lm)$coefficients[, 1], "\n")
```

Coefficients using linear model:
Intercept: 0.1391874
Slope: 2.726699

OK. Finally, we compare the parameters between the two models.

```r
#Compare the coefficients
cat("\nComparison in different ways of regression\n")
cat("difference in intercept:", ols_bfgs$par[1] - coef(ols_lm)[1], "\n")
cat("difference in slope:", ols_bfgs$par[2] - coef(ols_lm)[2], "\n")
```

Comparison in different ways of regression
difference in intercept: 1.38468e-08
difference in slope: 5.42236e-10

It seems that the parameters of the two models are equivalent and the experimental results are in line with expectations.