Problem Set 1

Applied Stats II

Maiia Skrypnyk 23371609

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:

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

Given the above-mentioned formulas for the test-statistic and the p-value, I followed the next algorithm to create a function for the Kosmogorov-Smirnov test in R:

1. Create empirical distribution of observed data;
2. Generate test statistic;
3. Indicate number of observations n;
4. Indicate a sequence of indices k;
5. Calculating the p-value[1];
6. The function should return values for the test-statistic D and the p-value.

```r
kosmogorov <- function(data) {
 ECDF <- ecdf(data) #Empirical distribution
 empiricalCDF <- ECDF(data)
 D <- max(abs(empiricalCDF - pnorm(data))) #Test statistic (reference
   distribution is normal)
 n <- length(data) #Number of observations
 k <- 1:n #Indices
 dsum <- sum(exp(-(2*k-1)^2*pi^2/(8*D^2))) #Sum of exponential terms for the
   p-value
 p_value <- sqrt(2*pi)/D*dsum #Final p-value
 return(list(D = D, p_value = p_value))
}
```

Setting the seed and generating 1,000 Cauchy random variables:

```r
set.seed(123)
variables <- rcauchy(1000, location = 0, scale = 1)
```

Performing the Kosmogorov-Smirnov test using the manually coded function:

```r
kosmogorov(variables)
```

[1]Calculating the sum of exponential terms first for further simplicity

```
$D
[1] 0.1347281

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

Comparing the results to the built-in R function:

```r
ks.test(variables, 'pnorm')
```

```
Asymptotic one-sample Kolmogorov-Smirnov test

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

Conclusions:

1. The values of the test-statistic D are quite close to each other, so we might suggest the difference occured due to rounding during the computing process.
2. The calculated p-values are extra small and close to zero ("essentially zero"). At the conventional threshold of statistical significance 0.05, or even 0.01, such small p-values (though different numbers) indicate that we have evidence to reject the null hypothesis of these two distributions' similarity – which makes total sense.

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 we know, OLS (Ordinary Least Squares) method minimizes the sum of squared errors (SSE). Sum of squared errors = the sum of squared differences between actual and predicted values of Y,so it is the sum of the squared residuals.

General formula for the linear model is given by $\hat{y} = \beta_0 + \beta_1 x$, and for our data it is $y = 0{+}2.75x$.

So, the logic behind my manually coded OLS regression function (pre-optimisation) would be:

1. Estimate the predicted value of y via the (general) linear model formula;
2. Calculate the residuals;

3. Calculate the SSE.
4. The function should return the SSE.

```r
ols <- function(beta, x, y) {
  y_hat <- beta[1] + beta[2] * x #Predicted values of the DV
  res <- y - y_hat #Residuals
  SSE <- (sum(res^2)) #Sum of squared errors
  return(SSE)
}
```

Using BFGS:

```r
ols_bfgs <- optim(par = c(0, 3),
                  ols,
                  x = data$x,
                  y = data$y,
                  method = "BFGS")
```

Exploring results:

```r
print(ols_bfgs$par)
```

```
[1] 0.1391874 2.7266985
```

Comparing the results to the `lm` function:

```r
lm1<- lm(data$y ~ data$x)
print(lm1)
```

```
Call:
lm(formula = data$y ~ data$x)

Coefficients:
(Intercept)          data$x
     0.1392          2.7267
```

The results are almost identical (again, "almost" due to rounding), and also close to the coefficients in the initial equation of $y = 0{+}2.75x$.

Also, as we have discussed during the lectures, the OLS method for a linear model is equivalent to the maximum likelihood estimation (MLE). Let's compare and prove this.

```r
linear.lik <- function(theta, y, X) {
  n <- nrow(X) #Number of observations
  k <- ncol(X) #Number of predictors
  beta <- theta[1:k] #Coefficients for predictors
  sigma2 <- theta[k + 1]^2 #Variance
  e <- y - X %*% beta #Residuals
  logl <- -0.5*n*log(2*pi)-0.5*n*log(sigma2)-((t(e) %*% e)/(2*sigma2)) #Log-
    likelihood function
  return(-logl) #Negative log-likelihood
}
```

Using BFGS:

```r
linear.MLE <- optim(fn = linear.lik,
                    par = c(0, 3, 1),
                    hessian = TRUE,
                    y = data$y,
                    X = cbind(1, data$x),
                    method = "BFGS")
```

Exploring results:

```r
print(linear.MLE$par)
```

```
[1] 0.1391874 2.7266985
```

The results are similar to what we got by using both the manually coded OLS regression function and the `lm` function.