1 PS1 Answers

Aryan Goyal
Student number: 18306046

2 Question 1

Write an R function that implements the Kolmogorov-Smirnov 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).

2.1 Answer 1

First I set the seed for reproducibility:

```
set.seed(100)
empirical <- rcauchy(1000, location = 0, scale = 1)
```

Next, I write a function to manually conduct the Kolmogorov-Smirnov test in R:

```
KS_function <- function (data){
  ECDF <- ecdf(data)
  empiricalCDF <- ECDF(data)

# generate test statistic
  D <- max(abs(empiricalCDF - pnorm(data)))

# Calculating the p-value
  i_values <- 1:1000
  sum1 <- sum(exp(-((2 * i_values - 1)^2 * pi^2) / ((8 * D)^2)))
  p_value <- sqrt(2 * pi) / D * sum1

# Return result
  result <- list(
  Test_statistic = D,
  P_value = p_value
)
  return(result)
}

# Performing the K-S test with our data
KS_function(empirical)
```

The function returns a list containing the test statistic ('D') and the p-value. The test statistic is 0.13997 and the p-value is 0.00683 Hence, the p-value is below the 0.05 level of significance

I compare the KS_function created by me, with the in-built function in R to check the results:

```
ks.test(empirical,"pnorm")
```

In this case, the test statistic is 0.14097 which is very close to the fuction and the p-value is 2.2e-16

When comparing the in-built function with the by-hand function, the results we get are quite similar. Both the p-values are below the 0.05 level of significance and the test statistics are also very similar (0.13997 for by-hand and 0.14097 for the in-built function)

3 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.

3.1 Answer 2

```
# Set seed for reproducibility and using the provided code to create the
# data
set.seed(191)
data <- data.frame(x = runif(200, 1, 10))
data$y <- 0 + 2.75 * data$x + rnorm(200, 0, 1.5)
```

Next, I initiate a function to calculate the log-likelihood for OLS

```
#Log-likelihood function for OLS (taking help from Tutorial 2 script)
log_likelihood <- function(outcome, input, parameter) {
  y_hat <- beta[1] + beta[2] * x
  residuals <- y - y_hat
  log_likelihood1 <- 0.5 * sum((residuals / 1.5)^2) + 0.5 * length(residuals) * log(2 * pi *
  return(log_likelihood1)
}
```

I set up the initial guess for coefficients for iteration and then optimize them using the BFGS algorithm through the optim() function

```
# Initial guess for coefficients for iteration
initial_guess <- c(0, 0)
```

```
# Optimize using BFGS
?optim
result_bfgs <- optim(par = initial_guess,fn = log_likelihood,
                     x = data$x, y = data$y, method = "BFGS")
```

Finally, I extract the coefficients from my function and compare them to the in-built lm function in R:

```
# Extract coefficients from BFGS result
coefficients_bfgs <- result_bfgs$par
print(coefficients_bfgs)

# OLS Regression using lm
ols_lm <- lm(y ~ x, data = data)

# Extract coefficients from lm result
print(coef(ols_lm))
```

Using the BFGS method, the intercept is 0.15734 and β_1 is 2.76752

Whereas, the intercept is 0.15208 and β_1 is 2.76760 using the in-built lm() function in R.
Hence, the by-hand method gets close to the in-built function in R