Problem Set 1

Robert Baker

2024-02-11

Question 1

First, we will set the seed for reproducibility and create the data using the rcauchy function. We will then calculate our test statistic and our p-value. Finally we can check these results vs. the in-built R function ks.test().

```r
# Setting seed
set.seed(909)

# Generating 1,000 Cauchy Random Variables
n <- 1000
empirical <- rcauchy(n, location = 0, scale = 1)

# Creating a K-S function
ksTest <- function (data){
  # Creating an empirical distribution of observed data
  ECDF <- ecdf(data)
  empiricalCDF <- ECDF(data)

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

  # Calculating P-value
  # Firstly, creating an empty vector
  summed <- NULL
  for(i in 1:n){
    summed <- c(summed, exp((-(2 * i - 1)^2 * pi^2) / ((8 * D)^2)))
  }
  pValue <- sqrt(2*pi)/D * sum(summed)
  cat("D =", round(D,2), "\n")
  cat("p-value =", round(pValue,2), "\n")
}
# K-S Test
ksTest(empirical)
```

```
## D = 0.13
## p-value = 0
```

```r
# Checking Test Results
ks.test(empirical, "pnorm")
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
```

```
## data:  empirical
## D = 0.13082, p-value = 2.776e-15
## alternative hypothesis: two-sided
```

We can note the results are very similar and we can assume the slight differences are down to rounding or approximation.

Question 2

We will firstly set the seed for reproducibility and generate the data. We can then write and execute a log-likelihood function. We will estimate the coefficients using the MLE technique. Following this we will check our output versus the OLS method using the R in-built lm() function.

```r
# Setting Seed
set.seed(909)

# Generating Data
data <- data.frame(x = runif(200, 1, 10))
data$y <- 0 + 2.75 * data$x + rnorm(200, 0, 1.5)

# Defining a Log-likelihood Function
norm_log_likelihood <- function(outcome, input, parameter) {
  n <- ncol(input) # Number of coefficients to estimate
  beta <- parameter[1:n] # Extracting coefficients
  sigma <- sqrt(parameter[1 + n]) # Estimating variance
  # Calculating log-likelihood
  -sum(dnorm(outcome, input %*% beta, sigma, log = TRUE))
}

# Estimating Coefficients using MLE
norm_results <- optim(
  fn = norm_log_likelihood,
  outcome = data$y,
  input = cbind(1, data$x),
  par = c(1, 1, 1),
  hessian = TRUE
)

# Displaying Estimated Coefficients
round(norm_results$par, 2)[1:2]
```

```
## [1] -0.11  2.75
```

```r
# Comparing Results with OLS - using lm()
round(coef(lm(y ~ x, data = data)), 2)
```

```
## (Intercept)           x
##       -0.11        2.75
```