Problem Set 1

Applied Stats II

Due: February 14, 2022

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).

Solution

The Cauchy distribution is equivalent to Student's-t with 1 degree of freedom. It is also equivalent to the distribution of the ratio of independent standard normal variables. The parameters of a Cauchy distribution are the location, which tells us where the peak occurs, and the scale, which is half of the inter-quartile range. I do not expect the CDF to match that of a normal distribution.

H_0: the data are normally distributed
H_a: the data are not normally distributed

First, generate 1,000 Cuachy random variables in R to put through the function:

```
rand_cau <- rcauchy(1000, location = 0, scale = 1)
```

My function will take two arguments: `data` is the vector of variables, and `alpha` is the significance level. I included `alpha` so that I could return a conclusion with the function.

Regarding the summation included in the formula for $p(D \leq x)$, I cannot implement a function to perform a sum to infinity, so I need to run it from k = 0 to k = some value, as an approximation. As it turns out, values of $e^{-(2k-1)^2\pi^2/(8x^2)}$ are infinitessimally small for the test-statistic generated by the Cauchy distribution in question, so summing to large values of k makes no difference to my conclusion in this case. My function sums as far as k = 1 million, in case I need to re-use it.

The output of my function will comprise: a statement of the null hypothesis, a test-statistic, a p-value, and a conclusion based on the inputted `alpha`.

```
my_ks_test <- function(data, alpha){  # Argument = data and significance level

  ECDF <- ecdf(data)                  # Use r's ecdf function to calculate
                                      # a CD function for the data
  ECD_val <- ECDF(data)               # Apply the CD function to the data
  D <- max(abs(ECD_val - pnorm(data)))# D is the difference between the
                                      # calculated ECDF and normal CDF

  k = 1000000                         # Summation will run from 1 to 1 million
  sigma = 0                           # Set initial value of summation
  for(i in 1:k){                      # Iterate from 1 to 100
                                      # Sum terms:
    sigma = sigma + exp((-(2 * i - 1)^2 * pi^2) / ((8 * D)^2))
  }
  p_value <- sqrt(2*pi)/D * sigma     # Final calculation for p-value

  if(p_value < alpha){                # Include a conclusion
    conc <- sprintf("Reject H0 at the %f significance level", alpha)
```

```
  }                                    # sprintf combines a string with a float
  else{
    conc <- sprintf("Cannot reject H0 since p-value is > %f", alpha)
  }
  output <- c("H0 is that the data are normally distributed.",
              sprintf("D = %f", D), sprintf("p-value = %f", p_value), conc)
  return(output)
}
```

Now to test the function on the random Cauchy variables. I have elected to use alpha = 0.05.

```
my_ks_test(rand_cau, 0.05)
```

This is the output from my function:

```
"H0 is that the data are normally distributed."
"D = 0.134728"
"p-value = 0.003802"
"Reject H0 at the 0.050000 significance level"
```

The in-built K-S test in R returned `D = 0.13573, p-value = 2.22e-16`. The test statistic is extremely close to the test statistic that I calculated. R's p-value is much smaller. It makes sense that R's p-value is smaller rather than larger, since capping the summation omits an infinte number of miniscule negative terms. However, it seems *so much* smaller that either my approximation is not very useful, or there is an error that I cannot see. It could be that successive terms in the summation are too small to register given the level of accuracy of the functions I have used, such as `sqrt`, `pi`, and `exp`.

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`.

Solution

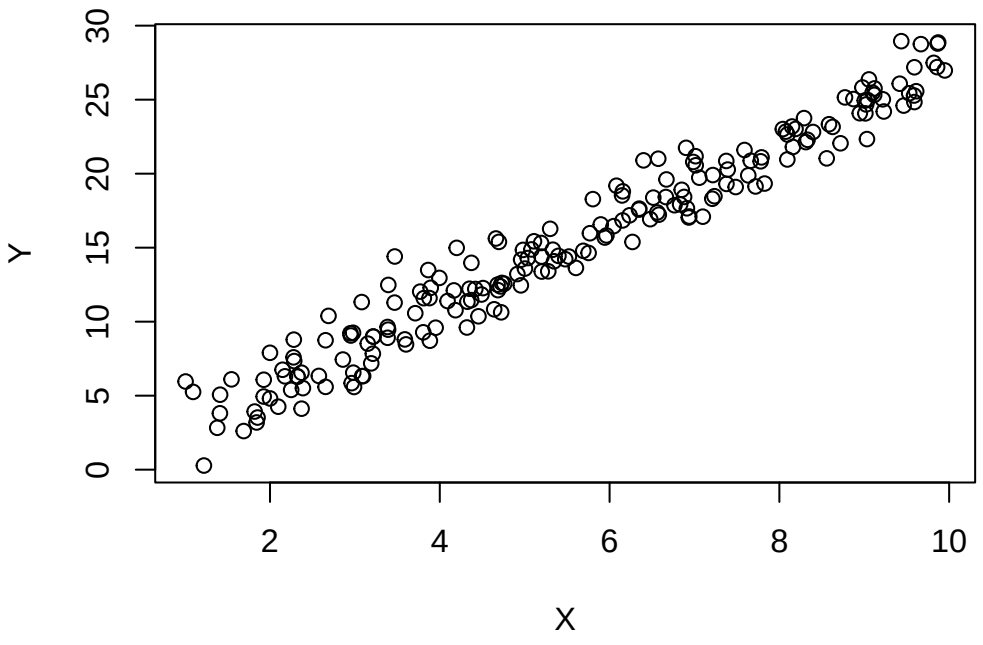
First, generate a data set, x, and a complementary set of y's using known parameters: $\beta_0 = 0$, $\beta_1 = 2.75$, $\sigma = 1.5$.

```
set.seed(123)

data <- data.frame(x = runif(200, 1, 10))      # Random x's
data$y <- 0 + 2.75*data$x + rnorm(200, 0, 1.5)# y = f(x), beta0=0, beta1=2.75
                                               # normally distributed errors
```

Since we generated the relationship we can look at the data.

```
plot(data$x, data$y,  ylab = "Y", xlab = "X")
```



We will assume a normal distribution and therefore create a `function` in R for the appropriate log likelihood function:

$$log(L) = \sum_{i=1}^{\infty} -\frac{n}{2}log(2\pi) - \frac{n}{2}log(\sigma^2) - \frac{1}{2\sigma^2}(y_i - x_i\beta)^2$$

In the code below, `theta` is a vector of parameters and `X` is a dataframe formed using `cbind(1, data$x)`. Both of these objects will be defined later when the function is optimised. The column of 1's would come into play when `X` is multiplied by the vector of beta's, except that in this case $\beta_0 = 0$.

```
like_fn <- function(theta, y, X){      # Arguments are:
                                       # 1. theta (vector of parameters)
                                       # 2. y = f(x)
                                       # 3. X = cbind(1, data$x)

  n <- nrow(X)                         # Number of x values
  k <- ncol(X)                         # Number of betas
  beta <- theta[1:k]                   # Dataframe of the betas in theta
  sigma_sq <- theta[k+1]^2             # Last element in theta, squared
  m <- y - X%*%beta                    # %*% multiplies 2 matrices

                                       # LLF for normal distribution:
  logl <- -0.5*n*log(2*pi) - 0.5*n*log(sigma_sq) - ((t(m)%*%m)/(2*sigma_sq))
                                       # t(m) transposes m to then square
  return(-logl)
}
```

Once the log likelihood function is defined it can be passed into R's `optim` function which will return an approximation for the vector of parameters, `theta`.

`optim` depends on a set of initial values, `par = c(...)`, and the output is sensitive to this set. My first solution used the `BFSG` method in `optim`, but returned a negative value for σ unless the initial values were quite close to the real values, particularly β. Instead, I have used the `L-BFSG-B` method which allows me to insist that σ is positive. Using `L-BFSG-B`, I can use `par = c(1, 1, 1)` to get estimates that are close to the real parameters.

```
OLS <- optim(fn = like_fn, par = c(1, 1, 1) , hessian = TRUE,
             y = data$y , X = cbind(1, data$x), method = "L-BFGS-B",
             lower = 0)
OLS
```

The estimated parameters are $\beta_0 = 0.139$, $\beta_1 = 2.727$, and $\sigma = 1.440$, all of which are close to the real values.

We can compare this to outputs from the `lm` function.

```
model_summary <- summary(lm (y ~ x, data))
model_summary$sigma                    # Get sigma
```

The estimated parameters are $\beta_0 = 0.139$, $\beta_1 = 2.727$, and $\sigma = 1.447$, which are almost identical to the BFSG method.
I tried using a different `set.seed()` and found that this removed the problem stated above regarding negative σ. I also created a wire frame when investigating the negative σ issue, but the shape of the frame is not correct.

```

surface <- list()
e <- 0
for (beta in seq(0, 5, 0.1)){
  for (sigma in seq(0.1, 5, 0.1)){
    e <- e+1
    logL <- like_fn(theta = c(0, beta, sigma), y = data$y,
                    X = cbind(1, data$X))
    surface[[e]] <- data.frame(beta = beta, sigma = sigma, logL = -logL)
  }
}
surface <- do.call(rbind, surface)
library(lattice)
wireframe (logL ~ beta*sigma, surface, shade = TRUE)
```