Problem Set 1

Applied Stats II

Due: February 14, 2022 (Dongli Wu - 21362988)

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 class on Monday February 14, 2022. No late assignments will be accepted.
- Total available points for this homework is 80.

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).

Write an `R` function that implements this test where the reference distribution is normal. 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(empirical)))

# make the hypotheses for the test:
  # Null hypothesis(Ho): the empirical distribution of the observed data
  matches the specified PDF.
  # Alternative hypothesis(H1): at least one value doesn't match the
  specified distribution.

  # set seed for random number generation
  set.seed(123)

  # generate 1000 random cauchy density varialbes
  y <-rcauchy(1000, location = 0, scale = 1)
  y

  # write the function to find the largest absolute difference between the
  empirical distirbution (use "ecdf(y)(y)") and the theoretical normal
  distribution (use "pnorm(y)") across all x position.

  D <- y[which.max(abs(ecdf(y)(y)-pnorm(y,mean(y),sd = sd(y))))]

  # set up the observed data as data frame.
  df <-data.frame(x = y)

  # plot the Empirical distribution and the Theoretical distribution, also
  mark the maximum difference between them.

  library(ggplot2)

  ggplot(df, aes(x)) +
    stat_ecdf(aes(colour = "Empirical")) +
    stat_function(fun = pnorm, args = list(mean = mean(y), sd = sd(y)),
                  aes(colour = "Theoretical")) +
  annotate(
    "segment",
    x = D, xend = D,
```

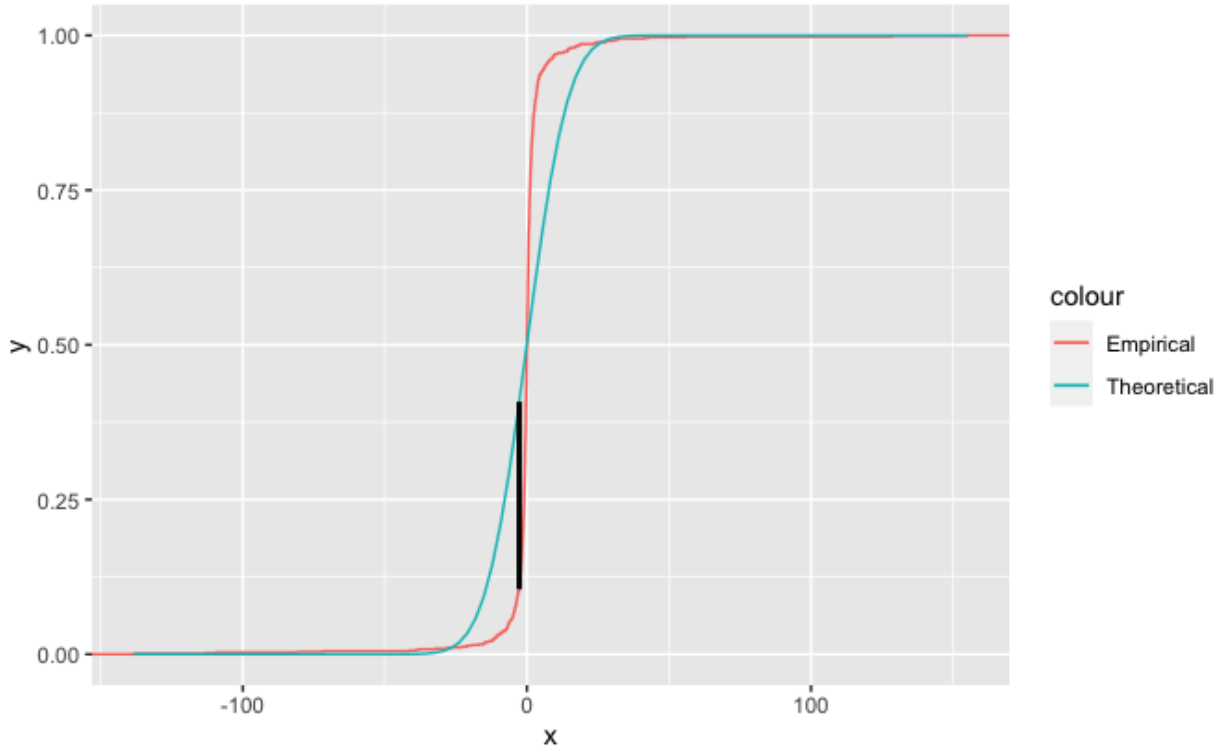
```
      y = ecdf(y)(D), yend = pnorm(D, mean(y), sd(y)),
      size =1)

  # The maximum difference shows on the gragh, D = 0.3 (=0.4-0.1)

  # Now we can use "ks.test()" to get P-value and test if the D value the
same result as above

  ks.test(y,"pnorm",mean(y),sd(y))

  # D = 0.30395, D-value equals to the result above.
  # p-value < 2.2e-16, P-value more less than 0.05, so the null hypothesis
could be rejected. The empirical distribution of the observed data doesn't
 match the theoretical distribution.
```



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.

```
set.seed(123)
data <- data.frame(x = runif(200, 1, 10))
data$y <- 0 + 2.75*data$x + rnorm(200, 0, 1.5)
```

```

    # set up the data

    set.seed(123)
    data <-data.frame(x = runif(200, 1, 100))
    data$y <- 0 + 2.75*data$x +rnorm(200, 0, 1.5)

    # write the function of the OLS regression with /theta=(\beta, \sigma)
    linear.lik <- function(theta, y, x){
      n <- nrow(x)
      k <- ncol(x)
      beta <- theta[1:k]
      sigma2 <- theta[k+1]^2
      e <- y-x%*%beta
      logl <- -0.5*n*log(2*pi)-0.5*n*log(sigma2)-((t(e)%*%e)/(2*sigma2))
      return(-logl)
    }



    # it creates a surface at different values of \beta and \sigma

    surface <- list()
    k <- 0
    for (beta in seq (0, 5, 0.1)){
       for (sigma in seq(0.1, 5, 0.1)){
         k <- k + 1
         logl <- linear.lik(theta = c(0, beta, sigma), y = data$y, x = cbind
    (1, data$x))
         surface[[k]] <- data.frame(beta = beta, sigma = sigma, logl = -logl)
        }
    }
    surface <- do.call(rbind, surface)
    library(lattice)

    wireframe(logl ~ beta*sigma, surface, shade = TRUE)

    # use "optim()" to find the parameters of the maximum point on the surface

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

```
40
41    linear.MLE$par
42
43    #[1]  0.118865  2.747869 -1.441841
44    # the result explores the parameters,$ \beta=2.75 and \sigma^2=118 $
45
46    summary(lm(y ~x, data))
47
48    #Coefficients:
49    #            Estimate Std. Error t value            Pr(>|t|)
50    #(Intercept) 0.118004   0.218720    0.54                0.59
51    #x           2.747882   0.003781  726.81 <0.0000000000000002
52
53    #(Intercept)
54    #x           ***
55
56
57    # Conclusion: lm turns out the equivalent results to the Newton-Raphson
      algotiyhm.
```