Problem Set 1

Applied Stats II

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

Answer:

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

ks_test <- function(x){
  #Use length function to find n, the size of the data vector.
  n <- length(x)
  #Code provided in the hint, used to find D the ks test statistic.
  # create empirical distribution of observed data
  ECDF <- ecdf(x)
  empiricalCDF <- ECDF(x)
  # generate test statistic
  D <- max(abs(empiricalCDF - pnorm(x)))
  #Function for finding p value
  #transcribed into R code from the Marsaglia formula provided in the
  question.
  p_value <- function(x) {
    (sqrt(2*pi)/x)*sum(exp(-(2 * (1:floor(x)) - 1)^2 * pi^2) / (8 * x^2))
  }

  p_value_result <- p_value(sqrt(n) * D)

  return(list(test_statistic = D, p_value = p_value_result))
}

#Test built function.
ks_test(data)

#Test results against ks test built into R.
ks.test(data, "pnorm")

```

The function I have written returns the following answers:
Test Statistic = 0.1347281
P Value = 2.095613e-07

The ks.test function built into R returns the following answers:
Test Statistic = 0.13573
P Value = 2.22e-16

The test statistics are both roughly **0.135** which is similar to a satisfactory degree, and within the margin for error which would be expected when employing two different formulations of the same test.

The p values are off by a factor of nearly one billion, or by eight decimal places. This is to be expected though, as the formula that Marsaglia et al. provide for calculating the ks test p value, which the R code has been built off, claims in the abstract to only be accurate to the seventh digit "we provide a quick approximation that gives accuracy to the 7th digit".[1] Which is exactly what the function I have written demonstrates, providing a p value the same as the p value calculated by ks.test function built into R, up to the seventh digit.

At a reasonable level as significance (e.g. $\alpha = 0.05$) the two answers are functionally the same.

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)
```

[1]"Evaluating Kolmogorov's Distribution", Marsaglia et al. Pp. 1

Answer:

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

#view the data for reference.
View(data)

#Define the objective function for OLS regression.
#beta =  the coefficients of the linear regression model.
#x = independent variable.
#y =  dependent variable.
Objective_ols_function <- function(beta, x, y) {
  #Linear regression equation.
  y_hat <- beta[1] + beta[2] * x
  #Calculate residuals.
  sum((y - y_hat)^2)
}

#Initial values for coefficients, set to zero.
initial_beta <- c(0, 0)

#Use BFGS method, in the optim function, to minimize the objective function.
ols_result <- optim(par = initial_beta, fn = Objective_ols_function, x =
  data$x, y = data$y, method = "BFGS")

#Estimated coefficients, subset from ols_results.
ols_coefficients <- ols_result$par

#Print estimated coefficients.
print(ols_coefficients)

#Compare with lm function.
lm_result <- lm(y ~ x, data = data)
print(summary(lm_result))

```

In order to estimate an OLS regression that uses the Newton-Raphson (BFGS) algorithm, first a function for calculating the residuals of a linear regression equation must be written.

Then the initial values of the coefficients are set to zero.

Both the newly written function and the initial values are used as arguments in the optim function in R, along with the method being set to "BFGS".

The estimated coefficients are stored within the parameters of the ols result, and can be subset using the dollar sign function.

Now the estimated coefficients can be printed and compared with the coefficients calculated using the lm function.

The estimated coefficients
= 0.1391778
= 2.7267000

Coefficients derived from lm function.
Intercept = 0.13919
slope = 2.72670

Both the estimated results and those derived from the lm function are equivalent to four decimal places.

Bibliography:

- Lam, A. (2022) BFGS in a Nutshell: An Introduction to Quasi-Newton Methods, Medium. Available at: https://towardsdatascience.com/bfgs-in-a-nutshell-an-introduction-to-quasi-newton-methods-21b0e13ee504 (Accessed: 10 February 2024).
- Nelder, J.A. and Mead, R. (1965) 'A Simplex Method for Function Minimization', The Computer Journal, 7(4), pp. 308–313. Available at: https://doi.org/10.1093/comjnl/7.4.308. btd (2024)
- 'optim: Streamlining Machine Learning Optimization in R', Medium, 6 January. Available at: https://baotramduong.medium.com/r-for-data-science-optimization-with-optim-e3aa19557848 (Accessed: 10 February 2024).
- Bélisle, C.J.P. (1992) 'Convergence theorems for a class of simulated annealing algorithms on R d ', Journal of Applied Probability, 29(4), pp. 885–895. Available at: https://doi.org/10.2307/3214721.
- Fletcher, R. (1964) 'Function minimization by conjugate gradients', The Computer Journal, 7(2), pp. 149–154. Available at: https://doi.org/10.1093/comjnl/7.2.149.

- Byrd, R.H. et al. (1995) ‘A Limited Memory Algorithm for Bound Constrained Optimization’, SIAM Journal on Scientific Computing, 16(5), pp. 1190–1208. Available at: https://doi.org/10.1137/0916069.
- Dr. Jeffrey Ziegler’s lecture material.