Problem Set 1

Idi Amin Da Silva . 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:

$$p(D \leq x) \frac{\sqrt{2\pi}}{x} \sum_{k=1}^{\infty} e^{-(2k-1)^2 \pi^2 / (8x^2)}$$

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).

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(data)))
```

0.1 Answer Q1

Here but text answer

```
#SPRING 2024 - PROBLEM SET1
################## QUESTION 1 ##########################
# ANSWER:
####
set.seed(123)
#First task is to create Kolmogorov-Smirnov test for goodness of fit to
    Gaussian/Normal distribution that should be normally
#distributed.

ks_test_normal <- function(data) {
#Here I have decided to create Empirical Cumulative Distribution Function(ECDF
    ) that is closely to Cumulative Frequency.
#The ECDF will shows the proportion of the scores that are less than or equal
    to each score.
# Or I Can say:  Create Empirical Distribution of observed data.

ECDF <- ecdf(data)
empirical_CDF <- ECDF(data)

# Generate test statistic

D <- max(abs(empirical_CDF - pnorm(data))) # The line of the code calculate
    the maximum absolute difference between the observed
#ECDF and theoretical CDF for each point or I can say: It quantify/measure the
     discrepancy between observed distribution and
#theoretical normal distribution.
head(D, n=5)
# Calculate p-value using Kolmogorov-Smirnov CDF approximation

```

```r
p_value <- sqrt(2 * pi) * D * exp(-(2 * (1:100)^2 * pi^2) / (8 * D^2)) # This line of the code compute the p_value for
#Kolmogorov_Smirnov test by transforming the mathematical expression/formula into R Code or  This line of the code convert the
#mathematical expression/formula of the p_vale(i.e., p(D<=x)) into Rcode for Kolmogorov-Smirnov CDF formula.
p_value <- 2 * sum(p_value)

# Return test result
my_result <- list(
my_test_statistic = D,
p_value = p_value,
decision = ifelse(p_value < 0.05, "Reject Null Hypothesis", "Fail to Reject Null Hypothesis")
  )

  return(my_result)
}

# Set seed for reproducibility
set.seed(123)

# Generate 1,000 Cauchy random variables
#
cauchy_data <- rcauchy(1000, location = 0, scale = 1) # This line of the code generate 1000 random number for the Cauchy
#distribution; location=0 represent the media of the distribution; scale=1 represent the half width and half maximum of the
#distribution. This mean that the generated random number will be centered around zero (location=0) and the spread of the
#distribution will be controlled by the scale parameter (scale=1)
head(cauchy_data, n=5) # the first 5 : 1.1606428 -3.2922040  0.5505110 -0.4960249 -0.5185047

# Let Perform the Kolmogorov-Smirnov test

my_result <- ks_test_normal(cauchy_data)
head(my_result)
## Answer:
######## ANSWER:
#$test_statistic
#[1] 0.1439423

#$p_value
#[1] 1.379198e-52

#$decision
#[1] "Reject Null Hypothesis"
```

The test statistic denoted by D=0.1439423 quantifies the largest difference between the Empirical Cumulative Distribution Function (ECDF) of the Observed data and the theoret-

ical CDF of the normal/Gaussian distribution. Since p-value $= 1.379198 \times 10^{-52}$ is less than $\alpha = 0.05$. The decision is to reject the Null Hypothesis. In the other words the rejection of the Null Hypothesis implies that the observed data does not fit the normal distribution well according to the Kolmogorov-Smirnov test.

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.

```
### ANSWER
##
# Set seed for reproducibility
set.seed(123)

# Create data frame
data <- data.frame(x = runif(200, 1, 10)) #This line of the code create a data
    frame with independent variable x with 200
#random number/values that are uniformly distributed between 1 and 10, (i.e.,
   n=200, min=1, max=10)
data$y <- 0 + 2.75 * data$x + rnorm(200, 0, 1.5) # This line of the code
   include the linear equation with intercept (beta_0=0),
#slope (beta_1=2.75) and random noise=rnorm(200, 0, 1.5), that generate n=200
   random number from normal distribution with mean=0
#and standard deviation 1.5
###
head(data, n=5)
#Answer:
#          x         y
#1    3.588198    8.801934
#2    8.094746  22.645878
#3    4.680792  12.502141
#4    8.947157  24.083367
#5    9.464206  24.599137
#####
head(data$y, n=5) # The first 5 observations: 8.801934 22.645878 12.502141
   24.083367 24.599137
########
##
# Define the OLS (Ordinary Least Square) goal/objective function
###
my_ols_goal <- function(beta, X, y) { #This line of the code named my_ols_goal
    take 3 parameters: 1st. beta-represent the
  #parameter/coefficient of linear model; 2nd. X-explanatory/independent
    variables in a matrix format; 3rd. y-the
  #outcome/dependent variable with a single column vector.
```

```r
  y_hat <- X %*% beta # This line of the code calculate the predicted value(y_hat) by performing the multiplication (%*%)
  #between explanatory variable X and the coefficient of the vector beta.
  residuals <- y - y_hat # This line of the code calculate the residual: difference between observed and predicted values for
  #each observation.
  return(sum(residuals^2)) # This line of the code provide the Residual Sum of Square (RSS), which is a measure of the the
  #Strength/Goodness of fit of the linear model.
}

# Use BFGS (Broyden - Fletcher - Goldfarb -Shannon) algorithm for optimization
fit_bfgs <- optim(par = c(0, 0), fn =my_ols_goal, method = "BFGS", X = cbind(1, data$x), y = data$y) # This line of the code
#uses the optim() function to perform the optimization applying the BFGS algorithm.
head(fit_bfgs) # par=c(0,0)- specifies the initial values of the coefficients to be optimized i.e., intercept and slope
## fn=my_ols_goal- specifies the objective function to be minimized; X=cbind(1, data$x)-represent the design matrix where the
#column of 1 represent the intercept and data$x represent the independent variable.
# y=data$y - represent the response/outcome variable.
############################
# Answer:
#
#$par
# Intercept: Beta_0 = 0.1391778; Slope(Coefficient for x): Beta_1 = 2.7267000

# $value
# 414.4577 - Represent the value of the objective function at the optimization solution.

# $counts
# function  gradient # The functions and gradients  were evaluate respectively 21 and 6  times to perform the optimization.
#   21        6

# $convergence
# 0 # Indicates whether the optimization algorithm converge to a solution. In this case the value of zero indicate the
#convergence to a solution

####
# 2nd method: will display only two parameters: Intercept and Slope Using the BFGS algorithm.

cat("BFGS Optimization Results:\n")
cat("Intercept:", fit_bfgs$par[1], "\n") # Answer: Intercept: Beta_0 = 0.1391778
```

```r
cat("Slope (Coefficient for x):", fit_bfgs$par[2], "\n")#Answer: Slope (Coefficient for x): Beta_1=2.7267
#########
####
# Fit the same model using lm for comparing both models in term of estimates parameters.
fit_lm_model <- lm(y ~ x, data = data)
summary(fit_lm_model)
##################
# Answer:
##
#Call:
#  lm(formula = y ~ x, data = data)
#
#Residuals:
#   Min      1Q  Median      3Q     Max
#-3.1906 -0.9374 -0.1665  0.8931  4.8032
#
#Coefficients:
#             Estimate    Std. Error    t value     Pr(>|t|)
#(Intercept)  0.13919     0.25276       0.551       0.582
#x            2.72670     0.04159       65.564      <2e-16 ***
#  ---
#  Signif. codes:  0    ***    0.001    **    0.01    *    0.05    .    0.1
            1
#
#Residual standard error: 1.447 on 198 degrees of freedom
#Multiple R-squared:  0.956,  Adjusted R-squared:  0.9557
#F-statistic:  4299 on 1 and 198 DF,  p-value: < 2.2e-16
#####
# We can see clearly that the both models (i.e., Newton-Raphson algorithm and Ordinary Least Square OLS) to fit the regression
# model with lm() function provide the same parameters estimators: beta_0_hat=0.13919  and beta_1_hat=2.72670
## The predicted Equation for the fitting line is given by: y_hat= beta_0_hat + beta_1_hat*x=0.13919 + 2.72670 *x.
#################################
#Data Visualization - Scatter plot x=runif(200, 1, 10) and y=data$y= 0 + 2.75* data$x + rnorm(200, 0, 1.5)
## Data Visualization: Scatter plot
##
ggplot(data=data, aes(x=runif(200, 1, 10)  , y=y))+
  geom_point(size=2)+
  geom_smooth(method="lm", formula=y ~ x, se=T)+
  theme_bw() + theme(panel.grid=element_blank()) +
  labs(y="linear Equation plus noise",title="Scatter plot of x=runif(200, 1, 10) Vs
  y=data$y =- 0 + 2.75 * data$x + rnorm(200, 0, 1.5)")
```

- Residual standard error: 1.447 on 198 degrees of freedom

Table 1: Regression Results

	Estimate	Std. Error	t value	$\Pr(\|t\|)$
(Intercept)	0.13919	0.25276	0.551	0.582
x	2.72670	0.04159	65.564	2×10^{-16}

- Multiple R-squared: 0.956, Adjusted R-squared: 0.9557
- F-statistic: 4299 on 1 and 198 DF, p-value = 2.2×10^{-16}

$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x = 0.13919 + 2.72670x$ and p-value = 2.2×10^{-16}.

From the predicted line or the best fitting line with with intercept $\hat{\beta}_0 = 0.13919$ and slope $\hat{\beta}_1 = 2.72670$.

That is means for every one unit increase in the predictor variable x, the estimate value of the outcome variable y will increase approximately by 2.7270 units.

On the other hand I have noticed that the output of the linear regression model when I applied the lm() function shows that the coefficient of determination $R^2 = 0.956$ is close to 1, it should be interpreted as the proportion of the total response variation explained by the regression line and a very low p-value = 2.2×10^{-16} means the model fit the data well and estimate coefficients $\hat{\beta}_0$ and $\hat{\beta}_1$ are statistically significant.

We can see clearly from the R code output that both method yield the same predicted coefficients (intercept and slope) shows that the linear regression model is well-specified and the optimization algorithm of Newton_Raphson converge to the same result. By comparing the two method approach, I can state the following: Newton_Raphson algorithm is an interactive optimization algorithm that seeks to minimize the sum of the squared residuals by updating the coefficients iterative until convergence. While OLS is defined in terms of residuals - it is the line which minimizes the squared residual.