Assignment 1

- **Student Name:** Umut Demirhan
- **Student Number:** 46739106
- **Lecturer's Name:** Connor Smith
- **Unit Code:** STAT8310
- **Unit Name:** Statistical Theory
- **Assignment Title:** Assignment 1
- **Due Date:** 19/May/2023
- **Date of Submission:** 19/May/2023**.**

Contents

Question 1

Part (a) Plotting to understand the relationship among the variables.

```
# Loading necessary packages
library(tidytuesdayR)
library(dplyr)
library(ggplot2)
library(corrplot)
```

- First, we will load the data from the CSV file using the **read.csv()** function and store it in a data frame called **DB,** excluding 'X' as it is not a variable.

```
# Load the dataset
starbucks <- read.csv("C:/Users/demir/OneDrive/Desktop/Assignment/starbucks.csv")
# Exclude column 'X'
DB <- select(starbucks, -X)
```

- Next, we can inspect the first few rows of the data frame using the **head()** function to get a sense of the variables and their values.

```
head(DB)
   calories total_fat_g cholesterol_mg sodium_mg total_carbs_g   sugar_g caffeine_mg
1 -1.637189   -1.020169     -0.8481098 -1.446475     -1.621355 -1.558173   0.4883517
2 -1.629925   -1.020169     -0.8481098 -1.392764     -1.621355 -1.558173   1.2949157
3 -1.622661   -1.020169     -0.8481098 -1.392764     -1.621355 -1.558173   2.1526900
4 -1.622661   -1.020169     -0.8481098 -1.392764     -1.621355 -1.558173   3.1768982
5 -1.637189   -1.020169     -0.8481098 -1.446475     -1.621355 -1.558173  -0.9839475
6 -1.629925   -1.020169     -0.8481098 -1.392764     -1.621355 -1.558173  -0.9199345
```

- Displaying the structure of the dataset.

```
str(DB)
```

```
'data.frame':	1147 obs. of  7 variables:
 $ calories      : num  -1.64 -1.63 -1.62 -1.62 -1.64 ...
 $ total_fat_g   : num  -1.02 -1.02 -1.02 -1.02 -1.02 ...
 $ cholesterol_mg: num  -0.848 -0.848 -0.848 -0.848 -0.848 ...
 $ sodium_mg     : num  -1.45 -1.39 -1.39 -1.39 -1.45 ...
 $ total_carbs_g : num  -1.62 -1.62 -1.62 -1.62 -1.62 ...
 $ sugar_g       : num  -1.56 -1.56 -1.56 -1.56 -1.56 ...
 $ caffeine_mg   : num  0.488 1.295 2.153 3.177 -0.984 ...
```

- Descriptive Statistics: You can use the summary() function in R to get a summary of your data.

```
# Get descriptive statistics for each variable
summary(DB)
```

```
    calories          total_fat_g      cholesterol_mg      sodium_mg       
 Min.   :-1.65898   Min.   :-1.0369   Min.   :-0.8481   Min.   :-1.50019  
 1st Qu.:-0.71470   1st Qu.:-0.8693   1st Qu.:-0.8481   1st Qu.:-0.74823  
 Median :-0.06097   Median :-0.2827   Median :-0.5699   Median :-0.04999  
 Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000  
 3rd Qu.: 0.66540   3rd Qu.: 0.6392   3rd Qu.: 0.8211   3rd Qu.: 0.64825  
 Max.   : 2.98979   Max.   : 3.6563   Max.   : 3.3250   Max.   : 2.47442  
 total_carbs_g         sugar_g          caffeine_mg     
 Min.   :-1.62136   Min.   :-1.55817   Min.   :-1.1760  
 1st Qu.:-0.76160   1st Qu.:-0.75671   1st Qu.:-0.7919  
 Median :-0.03081   Median :-0.04429   Median :-0.2158  
 Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000  
 3rd Qu.: 0.65700   3rd Qu.: 0.62360   3rd Qu.: 0.7444  
 Max.   : 2.50547   Max.   : 2.40463   Max.   : 4.9052  
```

- Calculating the correlation coefficients using the **cor()** function to get a numerical measure of the strength and direction of these relationships.

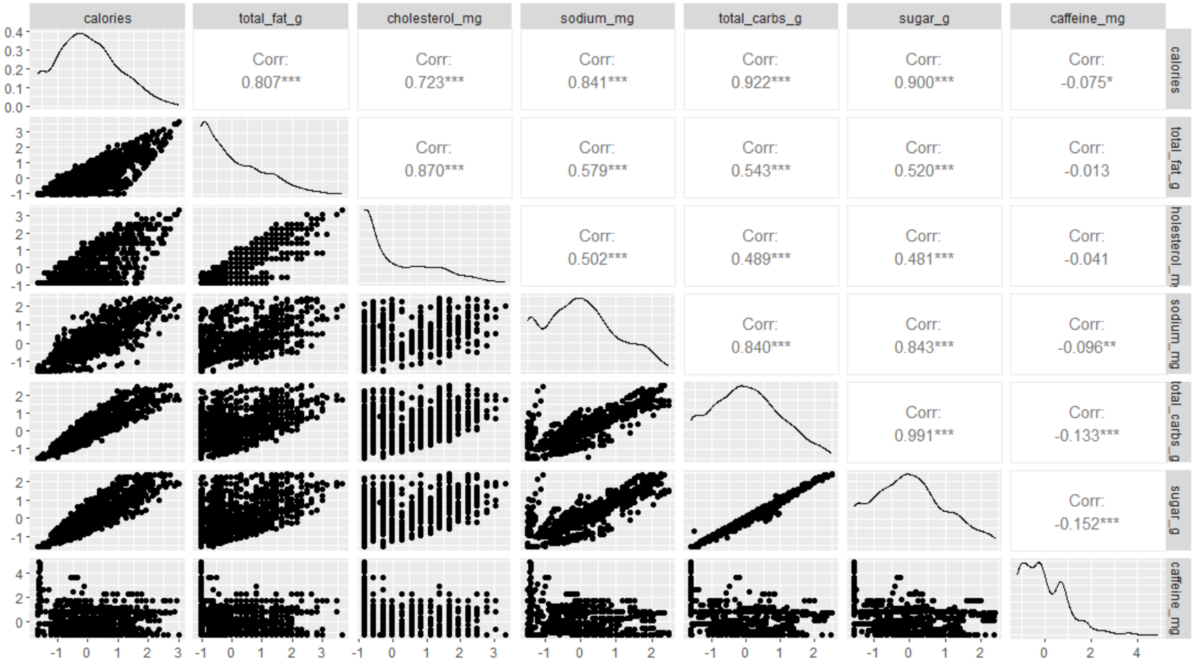
```
cor_matrix <- cor(DB)
cor_matrix
```

```
                 calories total_fat_g cholesterol_mg   sodium_mg
calories        1.00000000  0.80731381     0.72298837  0.84080546
total_fat_g     0.80731381  1.00000000     0.87040512  0.57938004
cholesterol_mg  0.72298837  0.87040512     1.00000000  0.50222940
sodium_mg       0.84080546  0.57938004     0.50222940  1.00000000
total_carbs_g   0.92159269  0.54302297     0.48856249  0.83953671
sugar_g         0.90028087  0.51958432     0.48118993  0.84318545
caffeine_mg    -0.07455882 -0.01307082    -0.04112263 -0.09575978
               total_carbs_g    sugar_g caffeine_mg
calories           0.9215927  0.9002809 -0.07455882
total_fat_g        0.5430230  0.5195843 -0.01307082
cholesterol_mg     0.4885625  0.4811899 -0.04112263
sodium_mg          0.8395367  0.8431855 -0.09575978
total_carbs_g      1.0000000  0.9912846 -0.13310151
sugar_g            0.9912846  1.0000000 -0.15198468
caffeine_mg       -0.1331015 -0.1519847  1.00000000
```

The correlation coefficients range from -1 to 1, where -1 indicates a perfect negative correlation, 0 indicates no correlation, and 1 indicates a perfect positive correlation. Overall, these results suggest that there are complex relationships between different nutrition levels, and that a high calorie item is likely to be high in other nutrients as well, except caffeine.

- Pair Plots: Pair plots are a fantastic way to visualize the distributions of each variable and the relationships between pairs of variables. The pairs() function in R can be used to create these.

```
# Create pair plots to visualize relationships between all variables
pairs(DB)
# Load GGally package
library(GGally)
# Create a pair plot
ggpairs(DB)
```



Exploring relationships

From the correlation plot plots, I can see that there are some correlations between the variables:

The correlation plot matrix shows that there are positive correlations between calories and total fat, total carbs and sugar. Additionally, there is a positive correlation between total fat and cholesterol, as well as sodium, and a positive correlation between total carbs and sodium. Sugar and total carbs have a positive correlation, while there is no strong correlation between sugar and sodium or caffeine and calories. The highest correlation is between Total carbs and sugar as 0.991. These relationships suggest that high calorie items are likely to be high in other nutrients. Caffeine, on the other hand, seems like uncorrelated with the other variables.

Part (b) Checking whether the data has already been centred and standardized Providing the supporting R output.

```
# Calculate the mean and standard deviation of each column in DB
DB_mean <- apply(DB, 2, mean)
DB_sd <- apply(DB, 2, sd)
# Display the mean and standard deviation of each column in DB
data.frame(mean = DB_mean, sd = DB_sd)
```

```
                        mean sd
calories       -4.713415e-17  1
total_fat_g     1.887117e-17  1
cholesterol_mg -5.479299e-16  1
sodium_mg       3.597547e-17  1
total_carbs_g  -7.194017e-17  1
sugar_g         3.509718e-16  1
caffeine_mg    -4.600359e-16  1
```

The standard deviations of variables are one, and the means can be considered zero, this indicates that the data has been standardized, and centred.

Part (c) Performing a principal component analysis on the variables. Providing the proportion of the variance explained by the principal components and the cumulative variance percentage.

```
# Perform PCA on the centered data
pca <- prcomp(DB)

# Calculate proportion of variance explained by each principal component
pca_prop_var <- round(100 * pca$sdev^2 / sum(pca$sdev^2), 2)

pca_prop_var
```

```
[1] 66.59 15.48 12.73  3.01  1.88  0.26  0.06
```

```
# Calculate cumulative variance percentage
cum_var <- cumsum(pca_prop_var)
cum_var
```

```
[1]  66.59  82.07  94.80  97.81  99.69  99.95 100.01
```

```
# Combine proportion of variance and cumulative variance percentage into a data frame
Principal_Component = paste0("PC", 1:length(pca_prop_var))
df_pca_results <- data.frame(Principal_Component = Principal_Component,
                             Proportion_of_Variance = pca_prop_var,
                             Cumulative_Variance_Percentage = cum_var)
# Display data frame of PCA results
df_pca_results
```

```
  Principal_Component Proportion_of_Variance Cumulative_Variance_Percentag
                  PC1                  66.59                         66.59
                  PC2                  15.48                         82.07
                  PC3                  12.73                         94.80
                  PC4                   3.01                         97.81
                  PC5                   1.88                         99.69
                  PC6                   0.26                         99.95
                  PC7                   0.06                        100.01
```

These outputs allow us to understand the importance of each principal component in capturing the variability in the data set.

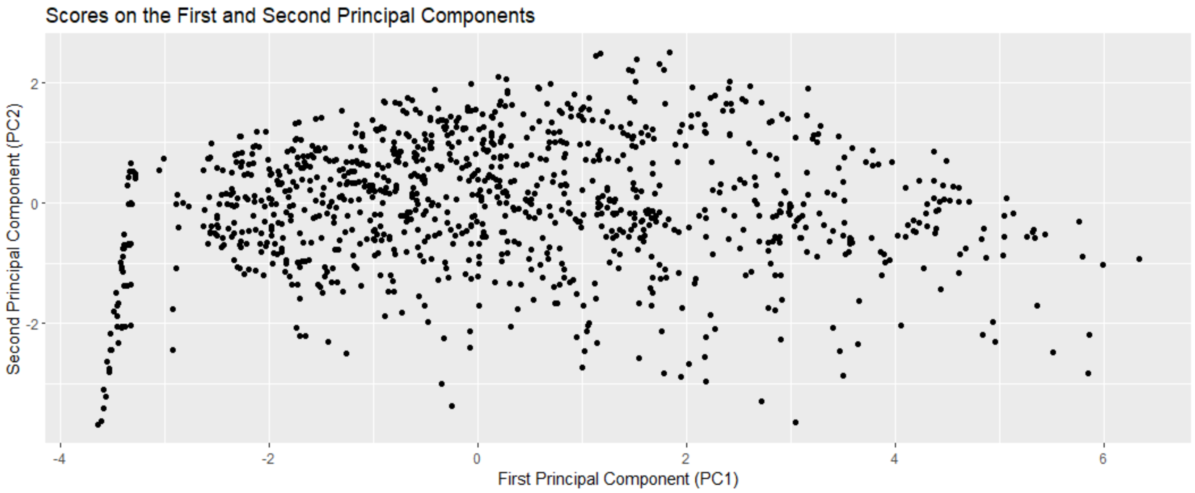
Part (d) Identifying the number of principal components that are needed to explain at least 95% of the variance in the starbucks data.

```
# Identify principal components that explain at least 95% of the variance
pc_95 <- which(cum_var >= 95)[1]
pc_subset <- 1:pc_95
pc_subset
[1] 1 2 3 4
```

The resulting pc_subset vector indicates the principal components that explain at least 95% of the variance in the starbucks data. So, the first 4 components cover 97.81% of the variance in dataset.

Part (e) Create a plot comparing the first and second principal components. Comment on this plot.

```
# Perform PCA
pca_results <- prcomp(DB, scale. = FALSE)

# Create a data frame for the first two principal components
pca_df <- data.frame(pca_results$x[,1:2])

ggplot(pca_df, aes(x = PC1, y = PC2)) +
  geom_point() +
  xlab("First Principal Component (PC1)") +
  ylab("Second Principal Component (PC2)") +
  ggtitle("Scores on the First and Second Principal Components")
```



Commments on the Plot:

Looking at the scatter plot of the first and second principal components, I notice a broad dispersion of data points. This suggests a wide range of variability captured by these components, but without obvious clustering. This might mean that the underlying structure of the data is complex and not easily separable by these two components alone. The variance captured by each axis would give more insights, and it would be interesting to see how the original variables contribute to these components.

Question 2

Part (a) R to simulate both X and Y

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

# Define parameters
n <- 200
m <- 300
mu <- 5
sigma_sq <- 12
sigma <- sqrt(sigma_sq)

# Simulate independent random samples X and Y
X <- rnorm(n, mean = mu, sd = sigma)
Y <- rnorm(m, mean = mu, sd = sigma)
```

Part (b) Using your simulated data, calculate the value of each estimator; T1 and T2.

```
# Calculate sample means
X_bar <- mean(X)
Y_bar <- mean(Y)

# Compute estimators
T1 <- (n * X_bar + m * Y_bar) / (n + m)
T2 <- 0.5 * (X_bar + Y_bar)

# Print results
cat("X_bar:", X_bar, "\n")
cat("Y_bar:", Y_bar, "\n")
cat("T1:", T1, "\n")
cat("T2:", T2, "\n")
```

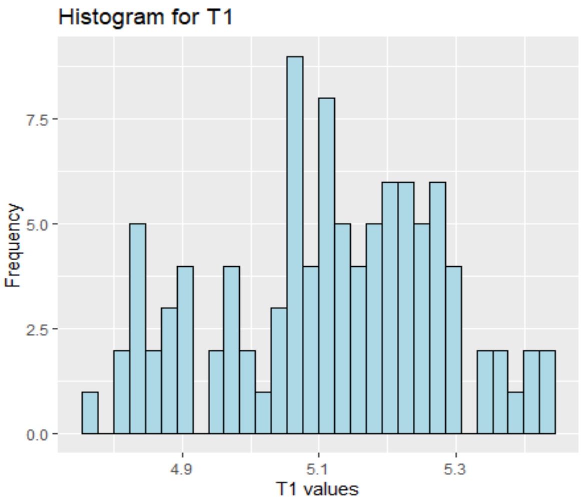
Outcome:

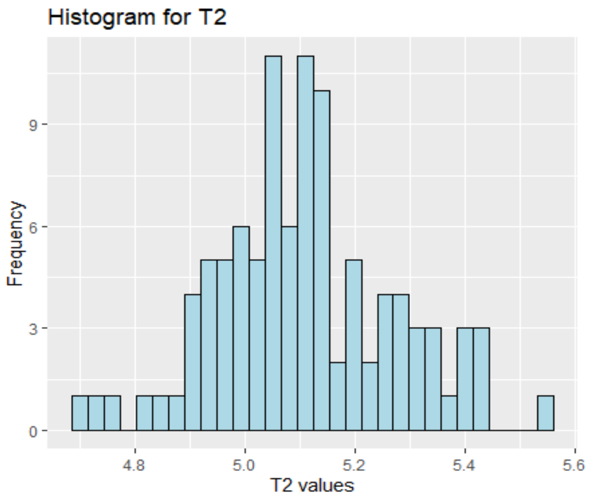
```
X_bar: 4.970311
Y_bar: 5.219501
T1: 5.119825
T2: 5.094906
```

Part (c) Using nonparametric bootstrapping (sampling with replacement), generate b = 100 bootstrapped values for each estimator. Construct histograms for each of the estimators.

```
# Required library for ggplot2
library(ggplot2)

# Set the number of bootstraps
b <- 100

# Initialize vectors to store bootstrap estimates
T1_boot <- numeric(b)
T2_boot <- numeric(b)

# Perform bootstrapping
for (i in 1:b) {
  # Sample with replacement
  X_boot <- sample(X, size = n, replace = TRUE)
  Y_boot <- sample(Y, size = m, replace = TRUE)

  # Calculate sample means
  X_bar_boot <- mean(X_boot)
  Y_bar_boot <- mean(Y_boot)

  # Compute estimators
  T1_boot[i] <- (n * X_bar_boot + m * Y_bar_boot) / (n + m)
  T2_boot[i] <- 0.5 * (X_bar_boot + Y_bar_boot)
}
# Create data frame for plotting
boot_estimates <- data.frame(T1 = T1_boot, T2 = T2_boot)

# Construct histograms
ggplot(boot_estimates, aes(x = T1)) +
  geom_histogram(color = "black", fill = "lightblue", bins = 30) +
  labs(title = "Histogram for T1",
       x = "T1 values",
       y = "Frequency")

ggplot(boot_estimates, aes(x = T2)) +
  geom_histogram(color = "black", fill = "lightblue", bins = 30) +
  labs(title = "Histogram for T2",
       x = "T2 values",
       y = "Frequency")
```

Outcome:





Part (d) Using the bootstrapped estimators, comment on each estimator. Which estimator would you prefer? Provide reasoning.

Both estimators are quite close to each other and to the true population mean (μ = 5). This is expected because both estimators are unbiased for the population mean.

However, the values of T1 and T2 are slightly different from each other. The distribution of T1 appears to be slightly wider than the distribution of T2, meaning T1 has a bit more variability. This makes sense because T1 is a weighted average of X and Y with weights proportional to the sample sizes, so it tends to be influenced more by larger samples and thus exhibits more variability.

On the other hand, T2 is a simple average of the sample means, so it doesn't give more weight to the larger sample and therefore has less variability.

Overall, in this case, I would prefer estimator T2 because it has less variability as indicated by the narrower distribution of bootstrap estimates.

Part (e) Construct a 95% confidence interval for both estimators using the bootstrapped values and the Empirical Distribution Function (ECDF)

```
# Function to calculate ECDF-based confidence intervals
ecdf_conf_interval <- function(boot_values, alpha = 0.05) {
  lower_quantile <- quantile(boot_values, probs = alpha / 2)
  upper_quantile <- quantile(boot_values, probs = 1 - alpha / 2)
  return(c(lower_quantile, upper_quantile))
}

# Calculate 95% confidence intervals for T1 and T2
T1_conf_interval <- ecdf_conf_interval(T1_boot, alpha = 0.05)
T2_conf_interval <- ecdf_conf_interval(T2_boot, alpha = 0.05)

# Print results
cat("95% confidence interval for T1: (", T1_conf_interval[1], ",", T1_conf_interval[2], ")\n")
cat("95% confidence interval for T2: (", T2_conf_interval[1], ",", T2_conf_interval[2], ")\n")
```

Outcome:

```
95% confidence interval for T1: ( 4.827335 , 5.432221 )

95% confidence interval for T2: ( 4.786906 , 5.424964 )
```

Part (f) (Without using R) Provide the theoretical expectation for each estimator? Does this agree with your empirical results in part (d)?

Let's derive the theoretical expectation for each estimator $T1$ and $T2$.

The theoretical expectation of T1 can be derived as follows:

$$E[T1] = E[(\bar{n}X + \bar{m}Y)/(n + m)]$$

$$= (\bar{n}E[X] + \bar{m}E[Y])/(n + m)$$

$$= (\bar{n}\mu + \bar{m}\mu)/(n + m)$$

$= (\bar{n} + \bar{m})\mu/(n + m)$

$= \mu$

Therefore, the theoretical expectation of T1 is equal to μ, which is 5 in this case.

The theoretical expectation of T2 can be derived as follows:

$E[T2] = E[1/2(\bar{X} + \bar{Y})]$

$= 1/2(E[\bar{X}] + E[\bar{Y}])$

$= 1/2(\mu + \mu)$

$= \mu$

Again, the theoretical expectation of T2 is equal to μ, which is 5 in this case.

Yes, the theoretical expectations derived in part (f) agree with the empirical results obtained in part (d). Both indicate that the expected values of the estimators T1 and T2 are equal to the population mean μ, which is 5 in this case.

In conclusion, the theoretical expectations for both estimators $T1$ and $T2$ are equal to the population mean μ

Question 3

Part (a) Likelihood and log-likelihood function for θ:

Given a sequence of i.i.d. random variables Xi with the given distribution, the likelihood function is the product of the probability mass functions of each Xi:

L(θ | X) = Π_i $fX(x_i)$ = Π_i [θ^(1−x_i) * (1 − θ)^x_i] = θ^(Σ_i (1 - x_i)) * (1 - θ)^(Σ_i x_i)

The log-likelihood function is the natural logarithm of the likelihood function, which turns the product into a sum:

l(θ | X) = log(L(θ | X)) = Σ_i [(1 - x_i) log(θ) + x_i log(1 - θ)]

Part (b) Derive the maximum likelihood estimator (MLE) $\hat{\theta}$ MLE of θ:

To obtain the MLE, I need to take the derivative of the log-likelihood with respect to θ, set it equal to zero and solve for θ. This gives me the value of θ that maximizes the log-likelihood function:

- l'(θ | X) = Σ_i [(1 - x_i) / θ - x_i / (1 - θ)]
- 0 = Σ_i (1 - x_i)/θ - Σ_i x_i / (1 - θ)
- θ_mle = Σ_i x_i / n

Part (c) Calculate the Cramér-Rao lower bound (CRLB) for the variances of unbiased estimators of θ.

I(θ) = E[((d/dθ)log(f(x|θ)))^2]

= E[((1-x)/θ - x/(1-θ))^2]

= n/[(θ)(1 - θ)]

Therefore, the CRLB is:

CRLB(θ) = 1 / I(θ) = (θ)(1 - θ) / n

Part (d) Find the variance of the MLE, $\hat{\theta}$mle and compare to the CRLB.

- Var($\hat{\theta}$_mle) = Var(∑Xi/n) = θ(1-θ)/n
- The Fisher Information, CRLB = 1/I(θ).

So, Var($\hat{\theta}$_MLE) = CRLB = θ(1-θ)/n

This indicates that the sample mean is an efficient estimator of θ in a Bernoulli distribution, because it achieves the lower bound of the variance.