ASSiGNMENT 1

```
A <- 23.4
B <- 45
C <- 678
cat("Values of A, B, and C:\n")
print(A)
print(B)
print(C)

cat("Entire variables:\n")
ls()

rm(C)
cat("List of variables after removing C:\n")
ls()

firstname <- "MyName"
lastname <- "MySurname"
cat("First Name:", firstname, "\nLast Name:", lastname, "\n")

binary_var <- 1
cat("Binary variable:", binary_var, "\n")

cat("Operations on A and B:\n")
cat("Addition (A + B):", A + B, "\n")
cat("Subtraction (A - B):", A - B, "\n")
cat("Multiplication (A * B):", A * B, "\n")
cat("Division (A / B):", A / B, "\n")

cat("Using various mathematical functions:\n")
cat("Exp(2):", exp(2), "\n")
```

```
cat("Log(10):", log(10), "\n")

cat("Log10(10):", log10(10), "\n")

cat("Log2(8):", log2(8), "\n")

cat("Pi:", pi, "\n")

cat("Square root of 16:", sqrt(16), "\n")


cat("Solving expressions:\n")

cat("1. 23 + (4.5 * 2.3) / 10 =", 23 + (4.5 * 2.3) / 10, "\n")

cat("2. 456 / 12 - log(90) =", 456 / 12 - log(90), "\n")

cat("3. Exp(5) + 12 / (5 ^ 6) =", exp(5) + 12 / (5 ^ 6), "\n")

cat("4. √45 * 12 / 3 =", sqrt(45) * 12 / 3, "\n")
```

ASSIGNMENT 2

USING SWIRL

ASSIGNMENT 3

```
# 1. Create an array A with elements (12, 13, 14, 15, 16) and display them

A <- c(12, 13, 14, 15, 16)

A


# 2. Find the sum of all the elements of A

sum(A)


# 3. Find the product of all the elements of A

prod(A)


# 4. Find the maximum and minimum element of A

max(A)

min(A)
```

```
# 5. Find the range of array A
range(A)

# 6. Find the mean, variance, standard deviation, and median of A
mean(A)
var(A)
sd(A)
median(A)

# 7. Sort the elements of A in increasing and decreasing order
B <- sort(A)      # Increasing order
C <- sort(A, decreasing = TRUE) # Decreasing order
B
C

# 8. Create a 3x4 matrix of natural numbers
matrix_3x4 <- matrix(1:12, nrow = 3, byrow = TRUE)
matrix_3x4

# 9. Create MxN matrix by combining A, B, and C row-wise (RW) and column-wise (CW)
RW <- rbind(A, B, C)
CW <- cbind(A, B, C)
RW
CW

# 10. Find the 2nd and 3rd row elements of RW
RW[2:3, ]

# 11. Find the 1st and 4th column of CW
CW[, c(1, 4)]
```

12. Sub-matrices with elements [2, 3] and [2, 4] in RW and CW

RW[2, 3]

RW[2, 4]

CW[2, 3]

CW[2, 4]

ASSIGNMENT 4

1. Vector Creation

seq(1.3, 4.9, by = 0.3)

rep(1:4, times = 5)

seq(14, 0, by = -2)

rep(c(5, 12, 13, 20), each = 2)

2. Loading and Exploring Data Structure

data(iris)

class(iris)

dim(iris)

is.factor(iris$Species)

levels(iris$Species)

3. Use the "iris" Dataset

aggregate(iris[, 1:2], by = list(Species = iris$Species), FUN = mean) # Mean Sepal.Width, Sepal.Length by Species

aggregate(iris[, 1:2], by = list(Species = iris$Species), FUN = sd)   # SD Sepal.Width, Sepal.Length by Species

iris.class <- iris

iris.class$Calyx.Width <- ifelse(iris.class$Sepal.Length < 5, "short", "long")

head(iris.class)

4. Explore Dataset - mtcars

```
str(mtcars)
names(mtcars)

mtcars[mtcars$cyl >= 5, ]
head(mtcars, 10)
mtcars[grep("Honda", rownames(mtcars)), ]
```

ASSIGNMENT 4.1

```
# Q1. Create Data Frame (DF)
DF <- data.frame(
  PatientID = c(1, 2, 3, 4),
  AdmDate = as.Date(c("2009-10-15", "2009-11-01", "2009-10-21", "2009-10-28")),
  Age = c(25, 34, 28, 52),
  Diabetes = c("Type1", "Type2", "Type1", "Type1"),
  Status = c("Poor", "Improved", "Excellent", "Poor")
)
DF

# Q2a. Extract PatientID and Age in Subset 1
Subset1 <- DF[, c("PatientID", "Age")]
Subset1

# Q2b. Identify Type1 patients
Type1Patients <- DF[DF$Diabetes == "Type1", ]
Type1Patients

# Q2c. Count the patients with Poor status
PoorCount <- sum(DF$Status == "Poor")
PoorCount

# Q2d. Print the summary of the DF
```

```
summary(DF)

# Q2e. Find the average age of patients having Diabetes
AvgAgeDiabetes <- mean(DF$Age)
AvgAgeDiabetes

# Q2f. Input more patient data from Keyboard
MoreData <- read.table(text = "5 11/12/2009 45 Type2 Fair
                        6 12/01/2009 38 Type1 Poor",
                  col.names = names(DF))
DF <- rbind(DF, MoreData)
DF

# Q3. Create a list named MyList
a <- c(12, 14, 16, 20)
matrix_2D <- matrix(1:10, nrow = 5)
s <- c("First", "Second", "Third")
MyList <- list(Title = "My First List", Criteria = list(Age = a, Matrix = matrix_2D, Scores = s))
MyList
MyList$Criteria
MyList$Criteria$Age
```

ASSIGNMENT 5

```
# Q1. Read the CSV file and print the first 10 records
library(dplyr)
url <- "https://raw.githubusercontent.com/fivethirtyeight/data/master/daily-show-guests/daily_show_guests.csv"
daily_show <- read.csv(url)
head(daily_show, 10)

# Q2. Rename the columns
daily_show <- daily_show %>%
```

```
  rename(YEAR = year,
         GoogleKnowlege_Occupation = job,
         Show = date,
         Group = category,
         Raw_Guest_List = guest_name)

# Q3. Create a report having YEAR, Show, and Raw_Guest_List
report <- daily_show %>%
  select(YEAR, Show, Raw_Guest_List)
report

# Q4. Use select to print all records except YEAR
no_year <- daily_show %>%
  select(-YEAR)
no_year

# Q5. Extract the list of people who are "actor" only and name is "ABC"
actors_ABC <- daily_show %>%
  filter(GoogleKnowlege_Occupation == "actor", Raw_Guest_List == "ABC")
actors_ABC

# Q6. Arrange the records in order of date
ordered_daily_show <- daily_show %>%
  arrange(Show)
ordered_daily_show

# Q7. Add a column "Experience"
daily_show <- daily_show %>%
  mutate(Experience = "To be filled")
daily_show
```

ASSIGNMENT 6

```r
# Q1. Create Dataset

library(dplyr)

set.seed(123)

data <- data.frame(

  Country = rep(c("USA", "India", "China", "France", "Germany", "Brazil", "Canada", "Russia", "Japan",
"Australia"), 2),

  Continent = rep(c("North America", "Asia", "Europe", "South America", "Oceania"), each = 4),

  Year = rep(2000:2009, each = 2),

  LifeExp = runif(20, 60, 85),

  Pop = sample(5e6:1e9, 20),

  gdpPerc = runif(20, 1000, 50000)

)


# Q1.1 Unique countries per continent

data %>%

  group_by(Continent) %>%

  summarise(UniqueCountries = n_distinct(Country))


# Q1.2 Lowest GDP per capita in Europe in a given year

data %>%

  filter(Continent == "Europe") %>%

  arrange(gdpPerc) %>%

  slice(1)


# Q1.3 Average life expectancy per continent in a given year

data %>%

  group_by(Continent, Year) %>%

  summarise(AvgLifeExp = mean(LifeExp, na.rm = TRUE))


# Q1.4 Top 5 countries by total GDP
```

```
data %>%
  group_by(Country) %>%
  summarise(TotalGDP = sum(gdpPerc * Pop, na.rm = TRUE)) %>%
  arrange(desc(TotalGDP)) %>%
  slice(1:5)

# Q1.5 Countries and years with life expectancies of at least 80 years
data %>%
  filter(LifeExp >= 80) %>%
  select(Country, Year)

# Q1.6 Top 10 countries by correlation between life expectancy and GDP per capita
data %>%
  group_by(Country) %>%
  summarise(Correlation = cor(LifeExp, gdpPerc, use = "complete.obs")) %>%
  arrange(desc(abs(Correlation))) %>%
  slice(1:10)

# Q1.7 Highest average population by continent (excluding Asia) and year
data %>%
  filter(Continent != "Asia") %>%
  group_by(Continent, Year) %>%
  summarise(AvgPopulation = mean(Pop, na.rm = TRUE)) %>%
  arrange(desc(AvgPopulation)) %>%
  slice(1)

# Q1.8 Three countries with the most consistent population estimates
data %>%
  group_by(Country) %>%
  summarise(PopStdDev = sd(Pop, na.rm = TRUE)) %>%
  arrange(PopStdDev) %>%
  slice(1:3)
```

```
# Q1.9 Observations where population decreased and life expectancy increased
data %>%
  arrange(Country, Year) %>%
  group_by(Country) %>%
  filter(Pop < lag(Pop) & LifeExp > lag(LifeExp)) %>%
  na.omit()

# Q2.1 Create DataSet.csv
med_data <- data.frame(
  MedID = 1:10,
  Med_Name = c("MedA", "MedB", "MedC", "MedD", "MedE", "MedF", "MedG", "MedH", "MedI",
"MedJ"),
  Company = c("Comp1", "Comp2", "Comp1", "Comp3", "Comp2", "Comp4", "Comp1", "Comp5",
"Comp4", "Comp2"),
  Manf_year = c(2015, 2016, 2017, 2018, 2019, 2020, 2015, 2021, 2022, 2023),
  Exp_date = as.Date(c("2025-01-01", "2026-01-01", "2027-01-01", "2028-01-01", "2029-01-01",
                       "2030-01-01", "2025-01-01", "2031-01-01", "2032-01-01", "2033-01-01")),
  Quantity_in_stock = c(100, 150, 120, 80, 200, 90, 110, 300, 250, 180),
  Sales = c(500, 400, 350, 600, 450, 700, 500, 800, 750, 850)
)
write.csv(med_data, "DataSet.csv", row.names = FALSE)

# Q2.2 First 4 records
med_data <- read.csv("DataSet.csv")
head(med_data, 4)

# Q2.3 Last 4 records
tail(med_data, 4)

# Q2.4 Correlation between Quantity_in_stock and Exp_date
cor(med_data$Quantity_in_stock, as.numeric(as.Date(med_data$Exp_date)))
```

```
# Q2.5 Bar graph for Sales vs Manufacturing year

library(ggplot2)

ggplot(med_data, aes(x = as.factor(Manf_year), y = Sales)) +

  geom_bar(stat = "identity") +

  xlab("Manufacturing Year") +

  ylab("Sales")


# Q2.6 Companies with more than one type of medicine

med_data %>%

  group_by(Company) %>%

  filter(n() > 1) %>%

  distinct(Company)


# Q2.7 Types of Medicine available

unique(med_data$Med_Name)


# Q2.8 Medicines expiring shown by box plots

ggplot(med_data, aes(x = Med_Name, y = as.numeric(as.Date(Exp_date)))) +

  geom_boxplot() +

  xlab("Medicine Name") +

  ylab("Expiration Date")


# Q2.9 Average stock in the store

mean(med_data$Quantity_in_stock)


# Q2.10 Regression line between Manufacturing year and Sales

ggplot(med_data, aes(x = Manf_year, y = Sales)) +

  geom_point() +

  geom_smooth(method = "lm", se = FALSE) +

  xlab("Manufacturing Year") +

  ylab("Sales")
```

ASSIGNMENT 7

```
# Q1. Create data matrix MARKS
set.seed(123)
MARKS <- matrix(sample(50:100, 60, replace = TRUE), nrow = 20, ncol = 3)
colnames(MARKS) <- c("SUB1", "SUB2", "SUB3")

# Q1a. Total marks of each student
TotalMarks <- apply(MARKS, 1, sum)

# Q1b. Append total to MARKS dataset
MARKS <- cbind(MARKS, Total = TotalMarks)

# Q1c. Function for standard error
st.err <- function(x) sd(x) / sqrt(length(x))
apply(MARKS[, 1:3], 2, st.err)

# Q1d. Add 0.25 bonus marks
MARKS[, 1:3] <- apply(MARKS[, 1:3], 2, function(x) x + 0.25)

# Q2. Create vectors V1, V2, and V3
V1 <- MARKS[, "SUB1"]
V2 <- MARKS[, "SUB2"]
V3 <- MARKS[, "SUB3"]
lapply(list(V1, V2, V3), sum)

# Q3. Create vector TOTAL_SUM using sapply()
TOTAL_SUM <- sapply(list(V1, V2, V3), sum)

# Q4. Compute squares of values in V1, V2, and V3 using sapply()
sapply(list(V1, V2, V3), function(x) x^2)
```

```r
# Q5. Add index field I and compute mean() and sd() of SUB1 using tapply()

I <- rep(1:4, each = 5)

tapply(MARKS[, "SUB1"], I, mean)

tapply(MARKS[, "SUB1"], I, sd)


# Q6. Create function f(x, y) and use mapply()

f <- function(x, y) x / y

mapply(f, V1, V2)


# Q7. Practice apply functions on "Seatbelts" dataset

data("Seatbelts")

apply(Seatbelts, 2, mean)

apply(Seatbelts, 2, sd)

sapply(Seatbelts, sum)

lapply(Seatbelts, max)

tapply(Seatbelts[, "DriversKilled"], Seatbelts[, "law"], mean)
```

ASSIGNMENT 8

```r
# Load necessary libraries

library(ggplot2)

library(dplyr)


# Read the data

url <-
"https://raw.githubusercontent.com/biocorecrg/CRG_RIntroduction/master/ex12_normalized_intensities.csv"

project1 <- read.csv(url, header = TRUE, row.names = 1)


# Q1.1: Simple scatter plot

p1 <- ggplot(project1, aes(x = sampleB, y = sampleH)) +
```

```
  geom_point() +
  labs(title = "Scatter Plot: sampleB vs sampleH", x = "sampleB", y = "sampleH")
print(p1)

# Q1.2: Add expr_limits column
project1 <- project1 %>%
  mutate(expr_limits = case_when(
    sampleB > 13 & sampleH > 13 ~ "high",
    sampleB < 6 & sampleH < 6 ~ "low",
    TRUE ~ "normal"
  ))

# Q1.3: Color scatter plot by expr_limits
p <- ggplot(project1, aes(x = sampleB, y = sampleH, color = expr_limits)) +
  geom_point() +
  labs(title = "Scatter Plot with expr_limits", x = "sampleB", y = "sampleH") +
  theme_minimal()
print(p)

# Q1.4: Boxplot of expression for all samples
project1_long <- project1 %>%
  pivot_longer(cols = starts_with("sample"), names_to = "sample", values_to = "expression")
p2 <- ggplot(project1_long, aes(x = sample, y = expression)) +
  geom_boxplot() +
  labs(title = "Boxplot of Gene Expression for All Samples", x = "Sample", y = "Expression")
print(p2)

# Q1.5: Sub-boxplots for low, normal, and high expressions
p3 <- ggplot(project1_long, aes(x = sample, y = expression, fill = expr_limits)) +
  geom_boxplot() +
  labs(title = "Sub-Boxplots for Gene Expression Levels", x = "Sample", y = "Expression") +
  theme_minimal()
```

```
print(p3)

# Q1.6: Bar plot of low/normal/high counts
expr_count <- project1 %>%
  count(expr_limits)
p4 <- ggplot(expr_count, aes(x = expr_limits, y = n, fill = expr_limits)) +
  geom_bar(stat = "identity") +
  labs(title = "Count of Gene Expression Categories", x = "Expression Limits", y = "Count") +
  theme_minimal()
print(p4)
```

ASSIGNMENT 11

```
install.packages("shiny")
install.packages("ggplot2")
install.packages("dplyr")
library(shiny)
library(ggplot2)
library(dplyr)

# Load libraries
library(shiny)
library(ggplot2)
library(dplyr)

# Define UI
ui <- fluidPage(
  titlePanel("Data Science Foundation Dashboard"),

  # Sidebar layout with input and output definitions
  sidebarLayout(
    sidebarPanel(
```

```
    selectInput("dataset", "Select Dataset:",
                choices = c("Iris", "Mtcars")),
    uiOutput("plot_selector")
  ),

  mainPanel(
    tabsetPanel(
      tabPanel("Summary", verbatimTextOutput("summary")),
      tabPanel("Plot", plotOutput("plot")),
      tabPanel("Data", tableOutput("data"))
    )
  )
 )
)

# Define server logic
server <- function(input, output, session) {

  # Reactive dataset based on user input
  dataset <- reactive({
    if(input$dataset == "Iris") {
      iris
    } else {
      mtcars
    }
  })

  # Dynamic UI for plot selection based on dataset
  output$plot_selector <- renderUI({
    if(input$dataset == "Iris") {
      selectInput("plot_type", "Choose plot for Iris:",
                  choices = c("Sepal Length vs Sepal Width", "Petal Length vs Petal Width"))
```

```
  } else {
    selectInput("plot_type", "Choose plot for Mtcars:",
                choices = c("Miles per Gallon vs Horsepower", "Miles per Gallon vs Weight"))
  }
})

# Render summary for the selected dataset
output$summary <- renderPrint({
  summary(dataset())
})

# Render the plot based on user selection
output$plot <- renderPlot({
  if(input$dataset == "Iris") {
    if(input$plot_type == "Sepal Length vs Sepal Width") {
      ggplot(dataset(), aes(x = Sepal.Length, y = Sepal.Width)) +
        geom_point(aes(color = Species)) +
        labs(title = "Sepal Length vs Sepal Width")
    } else {
      ggplot(dataset(), aes(x = Petal.Length, y = Petal.Width)) +
        geom_point(aes(color = Species)) +
        labs(title = "Petal Length vs Petal Width")
    }
  } else {
    if(input$plot_type == "Miles per Gallon vs Horsepower") {
      ggplot(dataset(), aes(x = hp, y = mpg)) +
        geom_point() +
        labs(title = "Miles per Gallon vs Horsepower")
    } else {
      ggplot(dataset(), aes(x = wt, y = mpg)) +
        geom_point() +
        labs(title = "Miles per Gallon vs Weight")
```

```
    }
  }
 })

 # Render the dataset as a table
 output$data <- renderTable({
   dataset()
 })
}

# Run the app
shinyApp(ui = ui, server = server)
```