

# Assignment 1

Ariz Kazani

2024-05-31

## Assignment 1

Name: Ariz Kazani

Student ID: 101311311

---

## Notes

TODO: add information about assignment and libraries used - make sure to double check and double read each question

---

## Question 1. Function Creation and Vector Operations

(a) Create a vector named sales that contains the following sales figures for a week: 250, 310, 450, 500, 620, 715, and 840

```
# vector that contains sales figures
sales <- c(250, 310, 450, 500, 620, 715, 840)
```

(b) Write a function named sales\_summary that takes a vector as input and returns the sum and mean of the vector. Test your function using the sales vector.

```
sales_summary <- function(vector) {
  Sum <- sum(vector)
  Mean <- mean(vector)
  returnVar <- list(Sum = Sum, Mean = Mean)
  return(returnVar)
}

sales_summary(sales)
```

```
## $Sum
## [1] 3685
##
## $Mean
## [1] 526.4286
```

(c) Write a function named `adjust_sales` that takes a vector and a percentage as inputs, adjusts each entry in the vector by the given percentage, and returns the adjusted vector in descending order. Test your function with the sales vector and a 10% increase.

```
adjust_sales <- function(vector, percentage) {
  newVector <- vector * ((percentage + 100 )/100)
  sort(newVector, decreasing=TRUE)
  return(newVector)
}

salesADJ <- adjust_sales(sales, 10)
```

(d) Create another test for the `sales_summary` function with a random vector of 10 elements. Print the result to check if your function works correctly with different inputs.

```
randomVector = c(22, 3, 24, 536, 774678, 895676, 57635, 24344, 123, 534)

sales_summary(randomVector)

## $Sum
## [1] 1753575
##
## $Mean
## [1] 175357.5
```

(e) Similarly, test the `adjust_sales` function with a random vector of 10 elements and a random percentage between 5% and 20%. Print the adjusted vector to ensure your function works correctly.

```
adjust_sales(randomVector, 17)

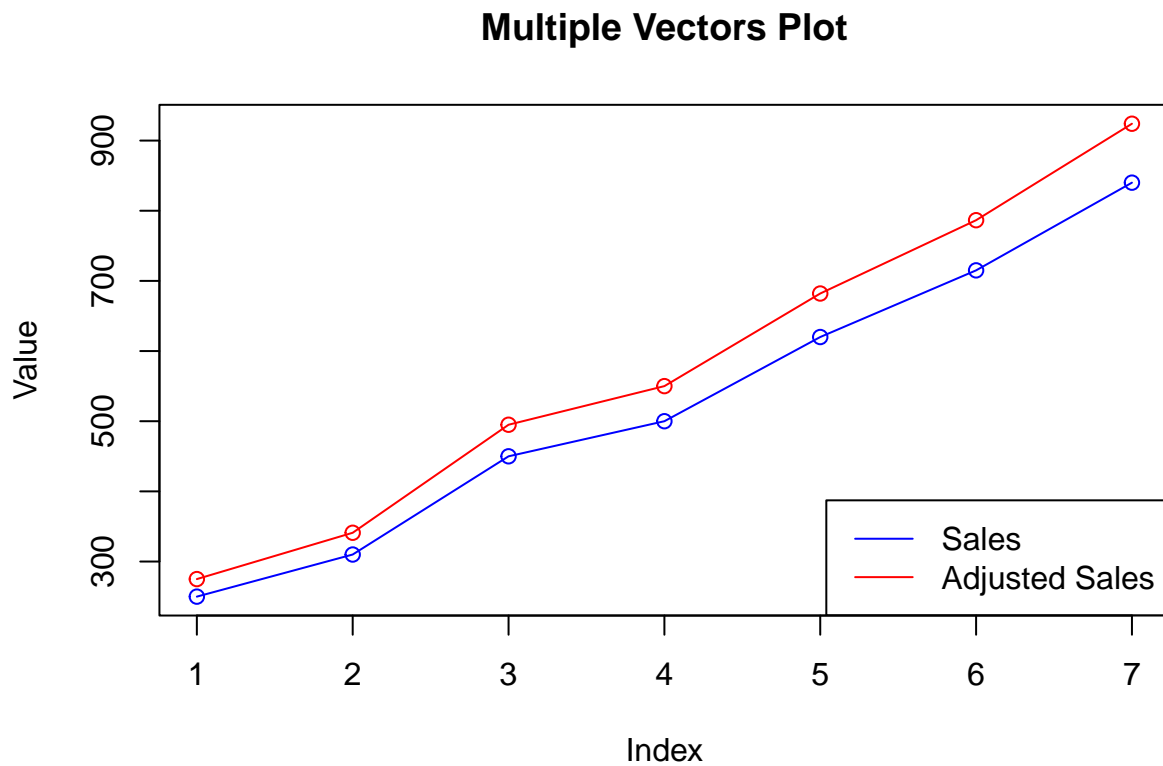
## [1] 25.74 3.51 28.08 627.12 906373.26 1047940.92
## [7] 67432.95 28482.48 143.91 624.78
```

(f) Plot the original sales vector and the adjusted sales vector (from Part 3) on the same graph using different colors. Label the axes and add a legend.

```
# TODO: fix this when the lecture comes out
plot(sales, type="o", col="blue", ylim=range(c(sales, salesADJ)), main="Multiple Vectors Plot", xlab="Index")

lines(salesADJ, type="o", col="red")

legend("bottomright", legend=c("Sales", "Adjusted Sales"), col=c("blue", "red"), lty=1)
```



## Question 2. Dataframe Operations and Descriptive Statistics

(a) Create a dataframe named `students` with the following data: Name: “Alice”, “Bob”, “Charlie”, “David”, “Eva” Age: 23, 22, 24, 21, 23 Score: 85, 92, 78, 88, 90

```
names <- c("Alice", "Bob", "Charlie", "David", "Eva")
ages <- c(23, 22, 24, 21, 23)
scores <- c(85, 92, 78, 88, 90)

students = data.frame(Name = names, Age = ages, Score = scores)

students
```

```
##      Name Age Score
```

```
## 1   Alice  23   85
## 2    Bob  22   92
## 3 Charlie 24   78
## 4   David 21   88
## 5    Eva  23   90
```

(b) Add a new column to the students dataframe named **Passed** with a value of **TRUE** if the Score is 80 or above, and **FALSE** otherwise.

```
students$Passed <- students$Score >= 80
```

```
df
```

```
## function (x, df1, df2, ncp, log = FALSE)
## {
##   if (missing(ncp))
##     .Call(C_df, x, df1, df2, log)
##   else .Call(C_dnf, x, df1, df2, ncp, log)
## }
## <bytecode: 0x0000018e3b82fca0>
## <environment: namespace:stats>
```

(c) Calculate the mean, median, and standard deviation of the Age and Score columns in the students dataframe.

```
meanAge = mean(students$Age)
medianAge = median(students$Age)
stdAge = sd(students$Age)
meanScore = mean(students$Score)
medianScore = median(students$Score)
stdScore = sd(students$Score)
```

```
meanAge
```

```
## [1] 22.6
```

```
stdAge
```

```
## [1] 1.140175
```

```
medianAge
```

```
## [1] 23
```

```
meanScore
```

```
## [1] 86.6
```

```
medianScore
```

```
## [1] 88
```

```
stdScore
```

```
## [1] 5.458938
```

(d) Identify the student(s) with the highest score and display their details.

```
maxScore = max(students$Score)
```

```
studentsWMS <- students[which(students$Score == maxScore),]
```

```
studentsWMS
```

```
##   Name Age Score Passed
```

```
## 2   Bob  22    92   TRUE
```

(e) Filter the dataframe to show only the students who passed and save it as a new dataframe named `passed_students`.

```
passed_students <- students[which(students$Passed),]
```

```
passed_students
```

```
##   Name Age Score Passed
```

```
## 1 Alice  23    85   TRUE
```

```
## 2   Bob  22    92   TRUE
```

```
## 4 David  21    88   TRUE
```

```
## 5   Eva  23    90   TRUE
```

(f) Create a bar chart showing the scores of all students. Use different colors for those who passed and those who did not.

```
# TODO: fill this out
```

(g) Write a short summary (3-5 sentences) interpreting the statistical results and the bar chart created in the previous steps.

```
# TODO: fill this out
```

### 3. Advanced Data Manipulation and Visualization

(a) Create a dataframe named `employees` with the following data:

- EmployeeID: 101, 102, 103, 104, 105
- Name: "John", "Jane", "Doe", "Smith", "Emily"
- Department: "Sales", "HR", "IT", "Finance", "Marketing"
- Salary: 60000, 65000, 70000, 72000, 68000
- Experience: 3, 7, 5, 10, 4

```
employeeID <- c(101, 102, 103, 104, 105)
name <- c("John", "Jane", "Doe", "Smith", "Emily")
department <- c("Sales", "HR", "IT", "Finance", "Marketing")
salary <- c(60000, 65000, 70000, 72000, 68000)
experience <- c(3, 7, 5, 10, 4)

employees <- data.frame(EmployeeID = employeeID, Name = name, Department = department,
                        Salary = salary, Experience = experience)

employees
```

```
##   EmployeeID Name Department Salary Experience
## 1         101 John      Sales  60000          3
## 2         102 Jane       HR   65000          7
## 3         103 Doe        IT   70000          5
## 4         104 Smith    Finance  72000         10
## 5         105 Emily  Marketing  68000          4
```

(b) Calculate the mean and median salary for each department. Write a function named `department_summary` that returns a summary dataframe containing the department name, mean salary, and median salary.

```
department_summary <- function(dataFrame) {
  getMeanMedian <- function(df, name) {
    dfWithName <- df[which(df$Department == name), ]
    returnDF = data.frame(DepartmentName = name, MeanSalary = mean(dfWithName$Salary), MedianSalary = median(dfWithName$Salary))
    return(returnDF)
  }
  departments <- unique(dataFrame$Department)
  df = data.frame()
  for (dep in departments) {
    df <- rbind(df, getMeanMedian(dataFrame, dep))
  }
  return(df)
}

department_summary(employees)
```

```
##   DepartmentName MeanSalary MedianSalary
## 1         Sales      60000      60000
## 2          HR      65000      65000
```

```
## 3          IT      70000      70000
## 4      Finance      72000      72000
## 5      Marketing      68000      68000
```

(c) Identify and display details of the employee with the highest salary in each department. Write a function named `top_earner` to achieve this.

```
top_earner <- function(dataFrame) {
  getMax <- function(df, dep) {
    peopleByDep <- df[which(df$Department == dep),]
    maxSal <- max(peopleByDep$Salary)
    topEarner <- peopleByDep[which(peopleByDep$Salary == maxSal), ]
    return(topEarner)
  }
  departments <- unique(dataFrame$Department)
  df = data.frame()
  for (dep in departments) {
    df <- rbind(df, getMax(dataFrame, dep))
  }
  return(df)
}

top_earner(employees)
```

```
##   EmployeeID  Name Department Salary Experience
## 1         101  John      Sales  60000          3
## 2         102  Jane       HR   65000          7
## 3         103   Doe       IT   70000          5
## 4         104 Smith    Finance  72000         10
## 5         105 Emily  Marketing  68000          4
```

(d) Add a new column to the employees dataframe named `AdjustedSalary`, which is the Salary adjusted for experience (increase by 2% for each year of experience).

```
employees$AdjustedSalary <- employees$Salary * ((100 + employees$Experience * 2)/100)
```

(e) Filter the dataframe to show only employees with an adjusted salary above 70,000 and save it as a new dataframe named `high_earners`.

```
high_earners <- employees[which(employees$AdjustedSalary > 70000),]
```

(f) Create a boxplot to compare the distribution of original salaries and adjusted salaries across different departments. Add appropriate labels and a title.

```
# TODO: fill this out
```

(g) Write a short analysis (4-6 sentences) interpreting the results from the summary statistics, top earners, and the boxplot.

---

## 4. Exploring Dataframes with Multiple Operations

(a) Create a dataframe named `products` with the following data:

- ProductID: 201, 202, 203, 204, 205
- ProductName: "Laptop", "Smartphone", "Tablet", "Headphones", "Smartwatch"
- Category: "Electronics", "Electronics", "Electronics", "Accessories", "Electronics"
- Price: 1200, 800, 600, 200, 350
- QuantitySold: 150, 200, 300, 400, 250

```
# TODO: fill this out
```

(b) Calculate the total revenue for each product ( $\text{Price} * \text{QuantitySold}$ ). Write a function named `calculate_revenue` that adds a new column `Revenue` to the `products` dataframe.

```
# TODO: fill this out
```

(c) Identify the product with the highest revenue and display its details.

```
# TODO: fill this out
```

(d) Group the products by `Category` and calculate the total revenue for each category. Write a function named `category_revenue` that returns a summary dataframe with `Category` and `TotalRevenue`.

```
# TODO: fill this out
```

(e) Create a bar chart to display the total revenue for each product, and use different colors for each category.

```
# TODO: fill this out
```

(f) Generate a scatter plot of `Price` versus `QuantitySold` with different colors for each category. Add a trend line to the plot.

```
# TODO: fill this out
```



(g) Write a detailed report (5-7 sentences) analyzing the

---

## 5. Debugging Subsetting and Indexing Issues

Explain the issues with the code and provide the correct working code. Output the code to show that you have it corrected.

(a)

```
students <- data.frame(
  Name = c("Alice", "Bob", "Charlie", "David", "Eva"),
  Age = c(23, 22, 24, 21, 23),
  Score = c(85, 92, 78, 88, 90))
# Extracting ages of students who scored above 80
high_scorers_ages <- students[students$Score > 80][, "Age"]
print(high_scorers_ages)
```

(b)

```
employee_list <- list(
  Name = "John",
  Age = 30,
  Department = "HR",
  Salary = 50000
)
# Accessing the salary of the employee
salary <- employee_list["Salaries"]
print(salary)
```

(c)

```
sales_data <- array(1:27, dim = c(3, 3, 3))
# Extracting the value in the second row, second column of the first matrix
value <- sales_data[3, 3, 0]
print(value)
```

(d)

```
products <- data.frame(
  ProductID = c(201, 202, 203, 204, 205),
  ProductName = c("Laptop", "Smartphone", "Tablet", "Headphones", "Smartwatch"),
  Category = c("Electronics", "Electronics", "Electronics", "Accessories", "Electronics"),
  Price = c(1200, 800, 600, 200, 350),
```

```
QuantitySold = c(150, 200, 300, 400, 250)
)
# Extracting products with a price above 500
expensive_products <- products[products$Price >= "500", ]
print(expensive_products)
```

---

## 6. Analysis of the “trees” Dataset

This dataset has three variables (Girth, Height, Volume) on 31 felled black cherry trees.

(a)

- Load the “trees” dataset and check the structure with `str()`.
- Use `apply()` to return the mean values for the three variables (Girth, Height, Volume) and output these values.
- Determine the number of trees with Volume greater than the mean Volume.

(b)

- Convert each Girth (diameter) to a radius  $r$ .
- Calculate the cross-sectional area of each tree using  $3.14 \times r^2$ .
- Calculate and output the interquartile range (IQR) of the areas.

(c)

- Create a histogram of the areas calculated in part (b).
- Title and label the axes.

(d)

- Identify the tree with the largest area.
  - Output its row number and the three measurements (Girth, Height, Volume) on one line
- 

## 7. Comprehensive Data Analysis and Function Creation

(a)

- Load the `mtcars` dataset.
- Filter the dataset to include only cars with 6 or more cylinders and horsepower greater than 150. Save this filtered dataset as `filtered_cars`.

(b)

- Create a function named `efficiency_score` that calculates an efficiency score for each car based on the formula:  $EfficiencyScore = \frac{mpg}{(hp \times wt)}$
- Apply this function to the `filtered_cars` dataset and add the resulting scores as a new column named `Efficiency`.

(c)

- Identify rows where the `Efficiency` score is less than the 1st percentile or greater than the 99th percentile of all `Efficiency` scores.
- Replace these outlier values with the mean `Efficiency` score of the remaining cars.

(d)

- Create a scatter plot of `hp` versus `Efficiency`, with points colored by the number of cylinders (`cyl`).
- Add a trend line to the scatter plot.
- Write a detailed analysis (6-8 sentences) interpreting the relationship between horsepower and efficiency, considering the number of cylinders.