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
#     - format code ctr + shift + a
#     - test all code to make sure its still working
#     - use sumaryfunction for dataframes where applicable
```

Solutions

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.

```r
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.

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

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

```
## [1] 924.0 786.5 682.0 550.0 495.0 341.0 275.0
```

(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.

```r
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.

```r
adjust_sales(randomVector, 17)
```

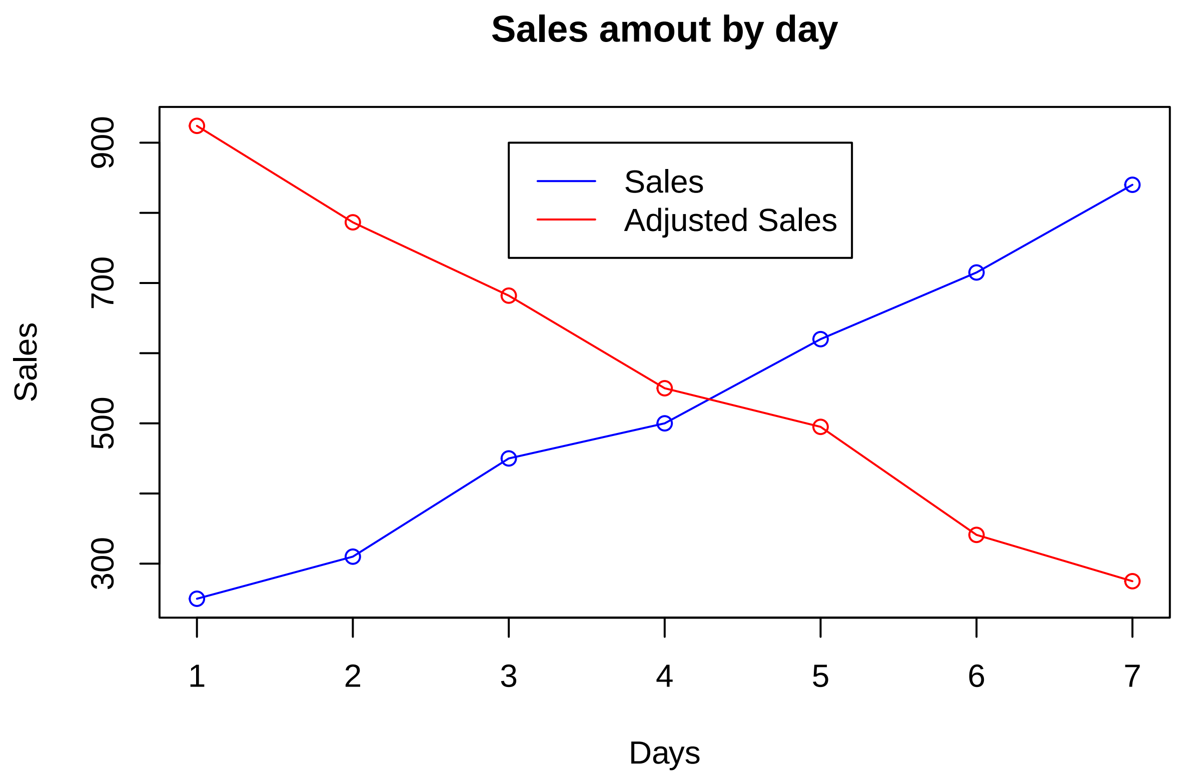
```
##  [1] 1047940.92  906373.26   67432.95   28482.48     627.12     624.78
##  [7]     143.91      28.08      25.74       3.51
```

(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.

```r
plot(
  sales,
  type = "o",
  col = "blue",
  ylim = range(c(sales, salesADJ)),
  main = "Sales amout by day",
  xlab = "Days",
  ylab = "Sales"
)

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

legend(
  3, 900,
  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

```
students = data.frame(
  Name = c("Alice", "Bob", "Charlie", "David", "Eva"),
  Age = c(23, 22, 24, 21, 23),
  Score = c(85, 92, 78, 88, 90)
)

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.

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

students
```

```
##      Name Age Score Passed
## 1   Alice  23    85   TRUE
## 2     Bob  22    92   TRUE
## 3 Charlie  24    78  FALSE
## 4   David  21    88   TRUE
## 5     Eva  23    90   TRUE
```

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

```r
ageData <- list(
  AgeMean = mean(students$Age),
  AgeMedian = median(students$Age),
  AgeSD = sd(students$Age)
)

scoreData <- list(
  ScoreMean = mean(students$Score),
  ScoreMedian = median(students$Score),
  ScoreSD = sd(students$Score)
)

ageData
```

```
## $AgeMean
## [1] 22.6
##
## $AgeMedian
## [1] 23
##
## $AgeSD
## [1] 1.140175
```

```r
scoreData
```

```
## $ScoreMean
## [1] 86.6
##
## $ScoreMedian
## [1] 88
##
## $ScoreSD
## [1] 5.458938
```

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

```
maxScore = max(students$Score)

studentsWMS <- students[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.

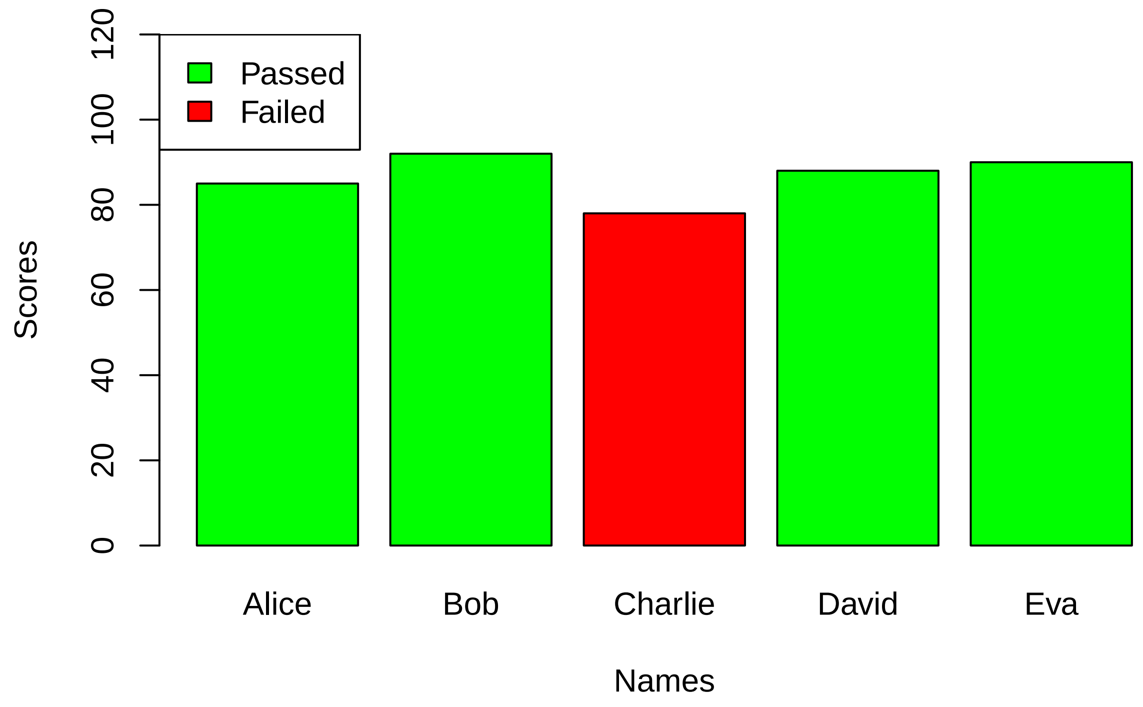
```
passed_students <- students[students$Passed, ]

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.

```
barplot(
  students$Score,
  names.arg = students$Name,
  ylim = c(0, 120),
  ylab = "Scores",
  xlab = "Names",
  col = ifelse(students$Passed, "Green", "Red"),
)
legend("topleft",
       legend = c("Passed", "Failed"),
       fill = c("Green", "Red"),)
```



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

From the presented data, we can conclude that most people passed, scoring well over the required 80%. we can also see that most people scored higher than the mean, with Alice and Charlie being the exceptions. The bar chart does not show any correlation between gender and the likelihood of passing. We can also see that most people are within one standard deviation form the mean, with Charlie and Bob being the exceptions.

---

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

```r
employees <- data.frame(
  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
```

```
##   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[df$Department == name, ]
    returnDF = data.frame(
      DepartmentName = name,
      MeanSalary = mean(dfWithName$Salary),
      MedianSalary = median(dfWithName$Salary)
    )
    return(summary(returnDF))
  }

  departments <- unique(dataFrame$Department)
  df = data.frame()
  for (dep in departments) {
    df <- rbind(df, getMeanMedian(dataFrame, dep))
  }
  return(df)
}

department_summary(employees)
```

```
##    Var1           Var2               Freq
## 1       DepartmentName Length:1
## 2       DepartmentName Class :character
## 3       DepartmentName Mode  :character
## 4       DepartmentName               <NA>
## 5       DepartmentName               <NA>
## 6       DepartmentName               <NA>
## 7           MeanSalary    Min.   :60000
## 8           MeanSalary    1st Qu.:60000
## 9           MeanSalary    Median :60000
## 10          MeanSalary    Mean   :60000
## 11          MeanSalary    3rd Qu.:60000
## 12          MeanSalary    Max.   :60000
## 13        MedianSalary    Min.   :60000
## 14        MedianSalary    1st Qu.:60000
## 15        MedianSalary    Median :60000
## 16        MedianSalary    Mean   :60000
## 17        MedianSalary    3rd Qu.:60000
## 18        MedianSalary    Max.   :60000
## 19      DepartmentName Length:1
## 20      DepartmentName Class :character
## 21      DepartmentName Mode  :character
## 22      DepartmentName               <NA>
## 23      DepartmentName               <NA>
## 24      DepartmentName               <NA>
## 25          MeanSalary    Min.   :65000
## 26          MeanSalary    1st Qu.:65000
## 27          MeanSalary    Median :65000
## 28          MeanSalary    Mean   :65000
## 29          MeanSalary    3rd Qu.:65000
## 30          MeanSalary    Max.   :65000
## 31        MedianSalary    Min.   :65000
## 32        MedianSalary    1st Qu.:65000
## 33        MedianSalary    Median :65000
## 34        MedianSalary    Mean   :65000
## 35        MedianSalary    3rd Qu.:65000
## 36        MedianSalary    Max.   :65000
## 37      DepartmentName Length:1
## 38      DepartmentName Class :character
## 39      DepartmentName Mode  :character
## 40      DepartmentName               <NA>
## 41      DepartmentName               <NA>
## 42      DepartmentName               <NA>
## 43          MeanSalary    Min.   :70000
## 44          MeanSalary    1st Qu.:70000
## 45          MeanSalary    Median :70000
## 46          MeanSalary    Mean   :70000
## 47          MeanSalary    3rd Qu.:70000
## 48          MeanSalary    Max.   :70000
## 49        MedianSalary    Min.   :70000
## 50        MedianSalary    1st Qu.:70000
## 51        MedianSalary    Median :70000
## 52        MedianSalary    Mean   :70000
## 53        MedianSalary    3rd Qu.:70000
## 54        MedianSalary    Max.   :70000
## 55      DepartmentName Length:1
## 56      DepartmentName Class :character
## 57      DepartmentName Mode  :character
## 58      DepartmentName               <NA>
## 59      DepartmentName               <NA>
## 60      DepartmentName               <NA>
## 61          MeanSalary    Min.   :72000
## 62          MeanSalary    1st Qu.:72000
## 63          MeanSalary    Median :72000
## 64          MeanSalary    Mean   :72000
## 65          MeanSalary    3rd Qu.:72000
## 66          MeanSalary    Max.   :72000
## 67        MedianSalary    Min.   :72000
## 68        MedianSalary    1st Qu.:72000
## 69        MedianSalary    Median :72000
## 70        MedianSalary    Mean   :72000
## 71        MedianSalary    3rd Qu.:72000
## 72        MedianSalary    Max.   :72000
## 73      DepartmentName Length:1
## 74      DepartmentName Class :character
## 75      DepartmentName Mode  :character
## 76      DepartmentName                <NA>
## 77      DepartmentName                <NA>
## 78      DepartmentName                <NA>
## 79          MeanSalary    Min.   :68000
## 80          MeanSalary    1st Qu.:68000
## 81          MeanSalary    Median :68000
## 82          MeanSalary    Mean   :68000
## 83          MeanSalary    3rd Qu.:68000
## 84          MeanSalary    Max.   :68000
## 85        MedianSalary    Min.   :68000
## 86        MedianSalary    1st Qu.:68000
## 87        MedianSalary    Median :68000
## 88        MedianSalary    Mean   :68000
## 89        MedianSalary    3rd Qu.:68000
## 90        MedianSalary    Max.   :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.

```r
top_earner <- function(dataFrame) {
  getMax <- function(df, dep) {
    peopleByDep <- df[df$Department == dep, ]
    maxSal <- max(peopleByDep$Salary)
    topEarner <- peopleByDep[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).

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

employees
```

```
##   EmployeeID  Name Department Salary Experience AdjustedSalary
## 1        101  John      Sales  60000          3          63600
## 2        102  Jane         HR  65000          7          74100
## 3        103   Doe         IT  70000          5          77000
## 4        104 Smith    Finance  72000         10          86400
## 5        105 Emily  Marketing  68000          4          73440
```

(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.

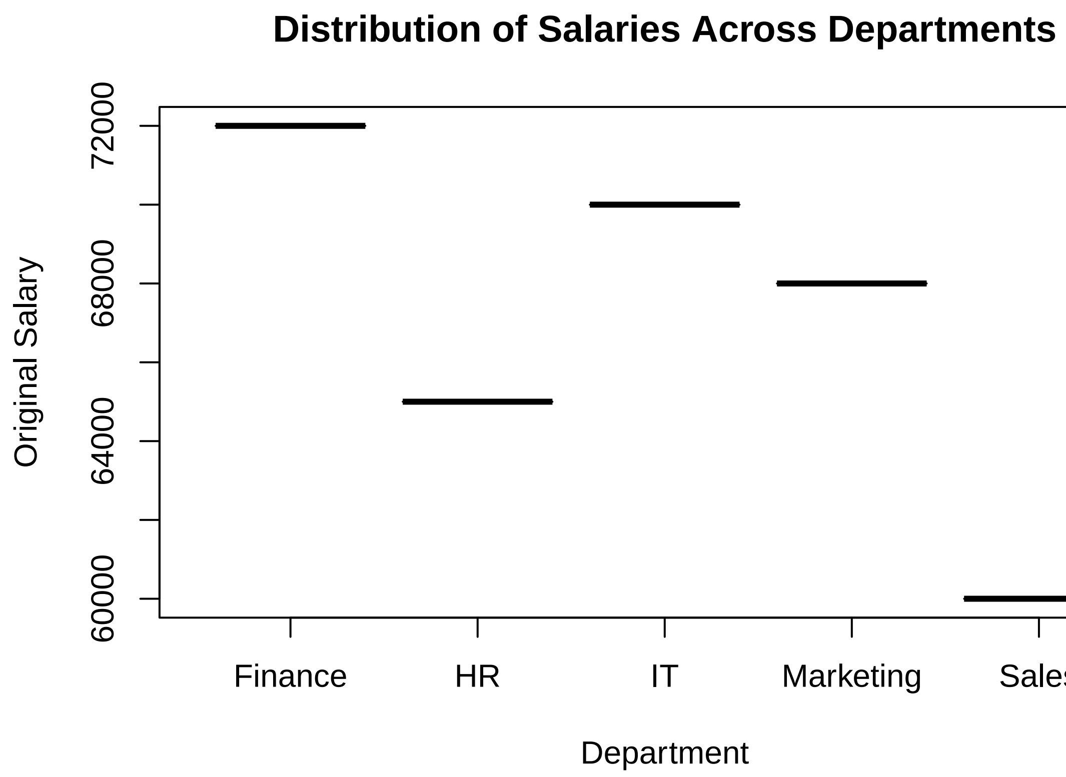
```r
high_earners <- employees[employees$AdjustedSalary > 70000, ]

high_earners
```

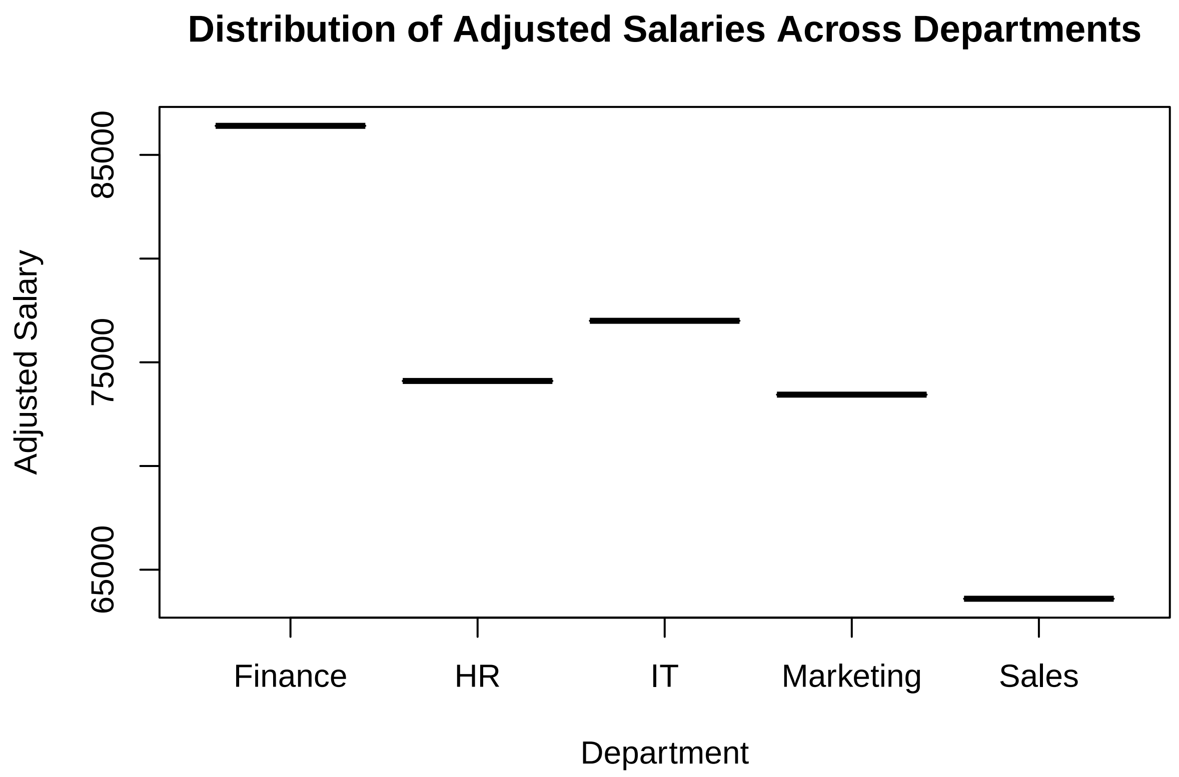
```
##   EmployeeID  Name Department Salary Experience AdjustedSalary
## 2        102  Jane         HR  65000          7          74100
## 3        103   Doe         IT  70000          5          77000
## 4        104 Smith    Finance  72000         10          86400
## 5        105 Emily  Marketing  68000          4          73440
```

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

```r
boxplot(
  employees$Salary ~ employees$Department,
  main = "Distribution of Salaries Across Departments",
  xlab = "Department",
  ylab = "Original Salary",
)
```



```r
boxplot(
  employees$AdjustedSalary ~ employees$Department,
  main = "Distribution of Adjusted Salaries Across Departments",
  xlab = "Department",
  ylab = "Adjusted Salary",
)
```



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

From the statistics we can see that the mean and median are exactly the same because there is only one person in each department. From the top-earners we can see that most people are earning above 70000 with the exception being sales. This is likely the case as sales has the lowest starting salary and the person working sales has been there the least amount of time. From the box plots we can see that the mean of the salary between all departments has increased, with the biggest jump happening from finance. We can also see, with the adjustment HR's mean salary goes from 4th to 3rd.

---

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

```r
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)
)
```

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

```r
calculate_revenue <- function(dataFrame) {
  dataFrame$Revenue <- dataFrame$Price * dataFrame$QuantitySold
  return(dataFrame)
}

products <- calculate_revenue(products)

products
```

```
##   ProductID ProductName    Category Price QuantitySold Revenue
## 1       201      Laptop Electronics  1200          150  180000
## 2       202  Smartphone Electronics   800          200  160000
## 3       203      Tablet Electronics   600          300  180000
## 4       204  Headphones Accessories   200          400   80000
## 5       205  Smartwatch Electronics   350          250   87500
```

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

```r
maxRevenue <- max(products$Revenue)
products[products$Revenue == maxRevenue, ]
```

```
##   ProductID ProductName    Category Price QuantitySold Revenue
## 1       201      Laptop Electronics  1200          150  180000
## 3       203      Tablet Electronics   600          300  180000
```

(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.

```r
category_revenue <- function(dataFrame) {
  dataByProduct <- function(dataFrame, category) {
    toteRev <- sum(dataFrame[dataFrame$Category == category, 'Revenue'])
    returnDf <- data.frame(Category = category, TotalRevenue = toteRev)
    return(returnDf)
  }
  catagories <- unique(dataFrame$Category)
  df <- data.frame()
  for (cat in catagories) {
    df <- rbind(df, dataByProduct(dataFrame, cat))
  }
  return(df)
}

category_revenue(products)
```

```
##      Category TotalRevenue
## 1 Electronics       607500
## 2 Accessories        80000
```

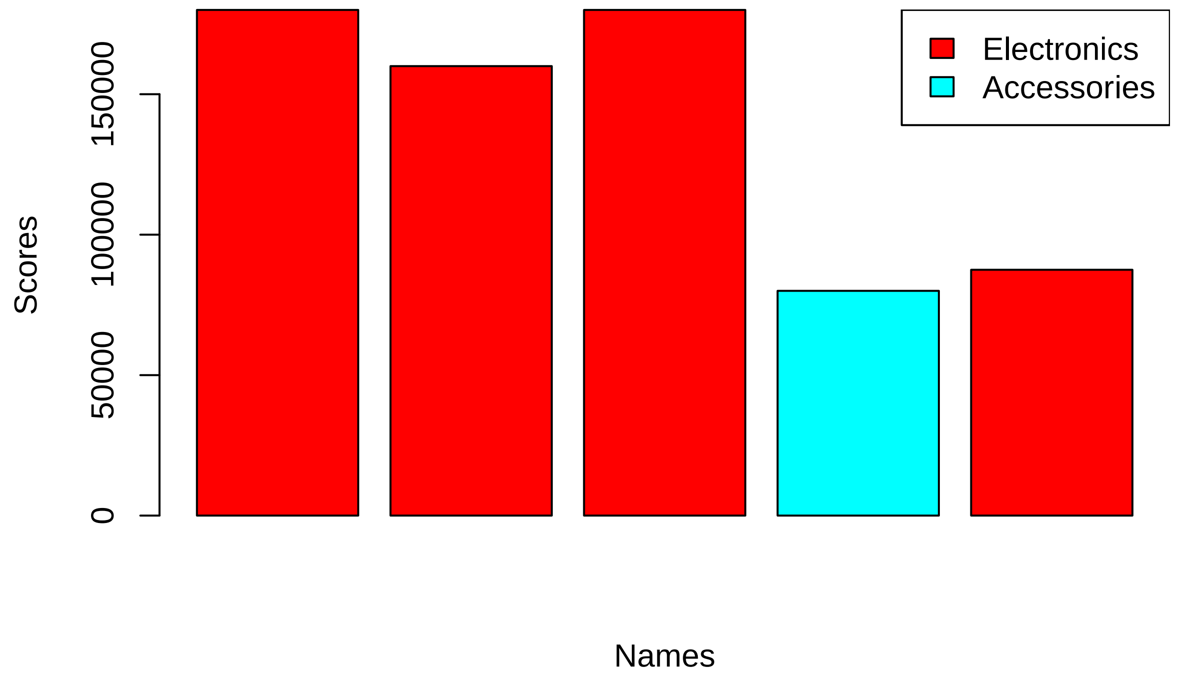
(e) Create a bar chart to display the total revenue for each product, and use different colors for each category.

```r
# Create a list with categories as names and colors as values
categories <- unique(products$Category)
numCat <- length(categories)
colours <- rainbow(numCat)

colourList <- list()
for (i in 1:numCat) {
  colourList[[categories[i]]] <- colours[i]
}

barplot(
  products$Revenue,
  names.arg = products$Name,
  ylab = "Scores",
  xlab = "Names",
  col = sapply(products$Category, function(col) colourList[[col]])
)

legend("topright",
       legend = categories,
       fill = colours)
```



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

```r
# 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)

```r
students <- data.frame(
  Name = c("Alice", "Bob", "Charlie", "David", "Eva"),
  Age = c(23, 22, 24, 21, 23),
  Score = c(85, 92, 78, 88, 90))
```

```r
# Extracting ages of students who scored above 80
high_scorers_ages <- students[students$Score > 80][, "Age"]
print(high_scorers_ages)
```

Explanation: `students[students$Score > 80]` is not correctly specifying rows and columns, to fix this we need to add a comma after specifying the columns like this `students[students$Score > 80,]`. Although it is not syntactically wrong we can also combine `students[students$Score > 80, ]` and `[, "Age"]` to look more clean `students[students$Score > 80, "Age"]`. **Corrected Code:**

```r
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)
```

```
## [1] 23 22 21 23
```

(b)

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

Explanation: The salary of the employee is not correctly being accessed. It was assigned as `Salary` but we are attempting to get `"Salaries"`. Instead we need to get `"Salary"`. **Corrected Code:**

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

```
## $Salary
## [1] 50000
```

(c)

```r
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)
```

Explanation: The first issue is that we aren't correctly selecting the row and column if we want the second row and second column we want to get the data from `[2, 2,]`. the second issue is that we are not correctly accessing the first matrix, since R is 1 indexed and not 0, to get the first matrix we need to use 1 to access it and not 0: `sales_data[2, 2, 1]`. **Corrected Code:**

```r
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[2, 2, 1]
print(value)
```

```
## [1] 5
```

(d)

```r
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)
```

Explanation: The first issue is that we are trying to compare string value with a numeric value. When this happens the numeric value gets converted into a string and gets compared lexicographically. This will not always give the desired result, for example `1200` gets converted into `"1200"` and then gets compared with `"500"`. Since `"1"` is smaller than `"5"` it will not add it. Another issue is that we want to get prices above 500 but we are also including 500 which is not what we want. **Corrected Code:**

```r
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)
```

```
##   ProductID ProductName    Category Price QuantitySold
## 1       201      Laptop Electronics  1200          150
## 2       202  Smartphone Electronics   800          200
## 3       203      Tablet Electronics   600          300
```

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.

```
data(trees)
str(trees)
```

```
## 'data.frame':    31 obs. of  3 variables:
##  $ Girth : num  8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
##  $ Height: num  70 65 63 72 81 83 66 75 80 75 ...
##  $ Volume: num  10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 ...
```

```
meanValues <- apply(trees, 2, mean)
meanValues
```

```
##    Girth   Height   Volume
## 13.24839 76.00000 30.17097
```

```
meanVolume <- meanValues["Volume"]
treesBiggerMean <- sum(trees$Volume > meanVolume)
treesBiggerMean
```

```
## [1] 12
```

(b)

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

```
trees$Girth <- trees$Girth / 2

crossSecArea <- (pi * trees$Girth ** 2)

IQRCSA <- IQR(crossSecArea)
IQRCSA
```

```
## [1] 87.1949
```

(c)

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

```
# TODO: sdfdmlsd
```

(d)

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

```
# TODO: sdfdmlsd
```

---

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.

```
data(mtcars)

filtered_cars <- mtcars[mtcars$cyl >= 6 & mtcars$hp > 150, ]

filtered_cars
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
```

(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.

```
efficiency_score <- function(dataFrame) {
  efs <- dataFrame$mpg / (dataFrame$hp * dataFrame$wt)
  return(efs)
}

filtered_cars$Efficiency <- efficiency_score(filtered_cars)
filtered_cars
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
##                      Efficiency
## Hornet Sportabout   0.031063123
## Duster 360          0.016349397
## Merc 450SE          0.022386022
## Merc 450SL          0.025767054
## Merc 450SLC         0.022339800
## Cadillac Fleetwood  0.009663182
## Lincoln Continental 0.008918159
## Chrysler Imperial   0.011957539
## Camaro Z28          0.014136905
## Pontiac Firebird    0.028534275
## Ford Pantera L      0.018879648
## Ferrari Dino        0.040639505
## Maserati Bora       0.012542330
```

(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.

```
percential <- quantile(filtered_cars$Efficiency, probs = c(0.01, 0.99))

outLiers <- which(filtered_cars$Efficiency < percential[1] | filtered_cars$Efficiency > percential[2])

meanRC <- mean(filtered_cars$Efficiency[-outLiers])
```

```
filtered_cars$Efficiency[outLiers] <- meanRC
filtered_cars
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
##                      Efficiency
## Hornet Sportabout   0.031063123
## Duster 360          0.016349397
## Merc 450SE          0.022386022
## Merc 450SL          0.025767054
## Merc 450SLC         0.022339800
## Cadillac Fleetwood  0.009663182
## Lincoln Continental 0.019419934
## Chrysler Imperial   0.011957539
## Camaro Z28          0.014136905
## Pontiac Firebird    0.028534275
## Ford Pantera L      0.018879648
## Ferrari Dino        0.019419934
## Maserati Bora       0.012542330
```

(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.

```
# TODO: sdfdmlsd
```