Introduction to Advanced Data Science (DSC 205)

Lyon College Spring 2025

R Basics Review: 10 Problems with Solutions

This review covers essential R topics, including arithmetic, vectors, matrices, data frames, factors, and lists. Solve each problem by writing R code. Add explanations when asked.

Problem 1: Arithmetic Operations

Write R expressions to perform the following calculations:

1. Add 15 and 27.
2. Divide 144 by 12.
3. Calculate 8 squared.
4. Find the remainder when 25 is divided by 4.

```
15 + 27 # + operator on two numeric values
144/12 # / operator on two numeric values
8^2 # ^ operator on two numeric values
25%%4 # modulo operator %: 25 / 4 = 5 + 1
```

```
[1] 42
[1] 12
[1] 64
[1] 1
```

Problem 2: Vector Operations

Create a numeric vector named **scores** with the values 85, 90, 78, 92, and 88. Perform the following operations:

1. Calculate the mean of the **scores** vector.
2. Add 5 to each element of the vector.
3. Select the third element from the vector.
4. Check if any element in the vector is greater than 95.

```
scores <- c(85,90,78,92)
scores
scores + 5
scores[3]
any(scores > 95)
```

```
[1] 85 90 78 92
[1] 90 95 83 97
[1] 78
[1] FALSE
```

Problem 3: Matrix Operations

Create a 3x3 matrix named **m** using the numbers 1 to 9, filled by row. Perform the following operations:

1. Extract the element in the second row, third column.
2. Calculate the sum of each row.
3. Transpose the matrix.

```
m <- matrix(1:9,nrow=3,byrow=TRUE)
m
m[2,3] # second row, third column
rowSums(m) # sum each row individually
t(m) # transpose matrix
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[1] 6
[1]  6 15 24
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

An alternative way of transposing `m` using two loops:

```
t_m <- matrix(nrow=3,ncol=3)
for (i in 1:3)
  for (j in 1:3)
    t_m[j,i] = m[i,j]
t_m
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

An alternative way of row-wise summation with a `for` loop and `sum`:

```
for (i in 1:3) print(sum(m[i,]))
```

```
[1] 6
[1] 15
[1] 24
```

An alternative way of row-wise summation with `rowsum`:

```
rowsum(t(m),group=rep(1,3))
```

```
  [,1] [,2] [,3]
1    6   15   24
```

An alternative way of row-wise summation with `tapply`: We want the sum for each `group` of values in one row.

```
group <- c(rep(1,3),rep(2,3),rep(3,3))
group
tapply(t(m),group,sum)
```

```
[1] 1 1 1 2 2 2 3 3 3
 1  2  3
 6 15 24
```

Problem 4: Creating Data Frames

1. Create a data frame named **students** with the following data:

Name	Age	Score
Alice	20	85
Bob	22	90
Charlie	19	78

2. Select the **Score** column and print it.
3. Add a new column named **Pass** that is TRUE if the score is 80 or above, and FALSE otherwise.

```
## 1. Create data frame
students <- data.frame(
  "Name" = c("Alice","Bob","Charlie"),
  "Age" = c(20,22,19),
  "Score" = c(85,90,78))
students

## 2. Select Score column
students$Score

## 3. Create new Pass column for Score greater or equal 80
students$Pass <- students$Score >= 80
students
```

```
     Name Age Score
1   Alice  20    85
2     Bob  22    90
3 Charlie  19    78
[1] 85 90 78
     Name Age Score  Pass
1   Alice  20    85  TRUE
2     Bob  22    90  TRUE
3 Charlie  19    78 FALSE
```

Problem 5: Factor Levels

Create a factor named **grades** with the values "A", "B", "A", "C", and "B". Perform the following operations:

1. Display the levels of the factor.
2. Change the levels to "Excellent", "Good", and "Average" corresponding to "A", "B", and "C", respectively.
3. Count how many students received each grade.

```
grades <- factor(c("A","B","A","C","B"))
levels(grades)
levels(grades) <- c("Excellent","Good","Average")
levels(grades)
table(grades)
```

```
[1] "A" "B" "C"
[1] "Excellent" "Good"      "Average"
grades
Excellent      Good   Average
        2         2         1
```

Problem 6: Lists

1. Create a list named **student1** that contains the following elements:
    - Name: "Alice"
    - Age: 20
    - Scores: A numeric vector with values 85, 88, 90
2. Access the **Name** element of the list and print it.
3. Add a new element to the list named **Graduated** with the value FALSE.

```
student1 <- list(Name="Alice",Age=20,Scores=c(85,88,90))
student1[["Name"]]
student1["Graduated"] <- FALSE
str(student1)
```

```
[1] "Alice"
List of 4
 $ Name     : chr "Alice"
 $ Age      : num 20
 $ Scores   : num [1:3] 85 88 90
 $ Graduated: logi FALSE
```

Problem 7: Subsetting Vectors

Given the vector **temperatures** = c(72, 75, 78, 80, 77, 73, 68), perform the following operations:

1. Extract the first three values and print them.
2. Find (and print) the temperatures greater than 75.
3. Replace the last value with 70 without using its index.

```
temperatures = c(72, 75, 78, 80, 77, 73, 68)
temperatures
temperatures[1:3]
temperatures[temperatures > 75]
```

```
temperatures[length(temperatures)] <- 70
temperatures
```

```
[1] 72 75 78 80 77 73 68
[1] 72 75 78
[1] 78 80 77
[1] 72 75 78 80 77 73 70
```

Problem 8: Matrix Calculations

1. Create a 2x3 matrix named **sales** with the following data:

Q1	Q2	Q3
150	200	250
180	220	270

2. Calculate the total sales for each quarter.
3. Find the maximum value in the matrix.

```
## Create matrix
sales <- matrix(
  c(Q1=c(150,180),
    Q2=c(200,220),
    Q3=250,270),
  nrow=2) # `byrow=FALSE` is default for `matrix`
sales

## Calculate total sales for each quarter
colSums(sales)

## Find the maximum value in the matrix
max(sales)
```

```
     [,1] [,2] [,3]
[1,]  150  200  250
[2,]  180  220  270
[1] 330 420 520
[1] 270
```

Problem 9: Data Frame Manipulation

Using the **students** data frame from Problem 4, perform the following operations:

1. Display the structure of the data frame.
2. Filter the data frame to show only students who passed.
3. Sort the data frame by the **Score** column in descending order.

R

```
str(students)   # show data frame
which(students$Pass==TRUE) # filter students who passed (index)
students[which(students$Pass==TRUE),]
```

```
order(students$Score,decreasing=TRUE) # sort rows by Score (desc)
students[order(students$Score,decreasing=TRUE),]
```

```
'data.frame':   3 obs. of  4 variables:
 $ Name : chr  "Alice" "Bob" "Charlie"
 $ Age  : num  20 22 19
 $ Score: num  85 90 78
 $ Pass : logi  TRUE TRUE FALSE
[1] 1 2
   Name Age Score Pass
1 Alice  20    85 TRUE
2   Bob  22    90 TRUE
[1] 2 1 3
     Name Age Score  Pass
2     Bob  22    90  TRUE
1   Alice  20    85  TRUE
3 Charlie  19    78 FALSE
```

Problem 10: Working with Factors and Lists Together

Create a list named **classroom** that contains:

- A factor named **Subjects** with levels "Math", "Science", and "History".
- A numeric vector named **Attendance** with values 25, 28, and 23.
- A character vector named **Teachers** with values "Mr. Smith", "Ms. Lee", "Mrs. Johnson".

Perform the following operations:

1. Print the list structure.
2. Access the **Subjects** element of the list and print only its elements.
3. Add a new element named **Location** with the value "Room 101".

```
## create list
classroom <- list(
  Subjects=factor(c("Math","Science","History")),
  Attendance=c(25,28,23),
  Teachers=c("Mr. Smith", "Ms. Lee", "Mrs. Johnson"))
str(classroom)

## Show `Subjects`
levels(classroom[["Subjects"]])

## Add `Location`
classroom$Location <- c("Room 101")
str(classroom)
```

```
List of 3
 $ Subjects  : Factor w/ 3 levels "History","Math",..: 2 3 1
 $ Attendance: num [1:3] 25 28 23
 $ Teachers  : chr [1:3] "Mr. Smith" "Ms. Lee" "Mrs. Johnson"
[1] "History" "Math"    "Science"
List of 4
 $ Subjects  : Factor w/ 3 levels "History","Math",..: 2 3 1
 $ Attendance: num [1:3] 25 28 23
 $ Teachers  : chr [1:3] "Mr. Smith" "Ms. Lee" "Mrs. Johnson"
 $ Location  : chr "Room 101"
```