SCS2211 - LABORATORY II
R Lab Practical Sheet - 08

Instructions

- **Do the Activities given in the practical sheet and take screenshots of the output**
- **Create a report using the screenshots.**
- **Report must be in PDF format.**
- **Report name should be <Index number>.pdf (Eg: 2000000.pdf)**
- **Any form of plagiarism or collusion is not allowed**

Simple Calculations

R allows basic arithmetic operations and provides built-in functions for mathematical calculations.

Example:

- 3 * 4 gives 12, and sqrt(16) returns 4.
- exp(x) computes e^x .For example, exp(1) returns Euler's number (~2.718).
- log(x, base) calculates the logarithm of x to the specified base. For example, log(8, base=2) gives 3.

Activity 1:

Start R and run the following commands.

a. >2+2
b. >exp(-2)#call the exponential function and pass the value -2
c. > log(100, base=10) #call the log function. Notice that this function takes 2 arguments, the value and the base. (log10 100 = 2)
d. > runif(10) #generate 10 random numbers in the range [0,1]. Run this command twice and compare the results

Working with Variables

Variables are used to store data, which can be manipulated or reused later. Assigning a value uses = or <-.

Example:

- a = 10; b = 5; a * b results in 50.
- Variables can hold different types, like numbers or strings: name = "John Doe".

Activity 2:

Assign values to variables and perform operations:

a. x = 2
b. x + x
c. y = x + 3
d. Print y
e. Assign a string to a variable: s = "this is a char str"
f. Print s

Creating and Manipulating Vectors

Vectors store sequences of elements. Use c() to combine elements into a vector.

Example:

- my_vector = c(5, 10, 15) creates a numeric vector.
- You can apply functions to entire vectors: sum(my_vector) returns 30.

```
> weight = c(60,70,86,97,45,67)
> weight
[1] 60 70 86 97 45 67
```

- Plot the values of weight vector,

  > plot(weight)
- Create a vector of regularly spaced numbers,

  > seq(0,1, length = 11)

```
> seq(0,1, length = 11)
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

- Also try
  - > seq (4, 10, 0.5)
  - > seq (length=10)
  - You would have noticed that the same function (e.g. seq) can be used in many ways. And the output depends on how we call the function.
  - help(seq) #This will open the help file for the function seq(). It will explain typical usages and the arguments.
    The c() function can be used to combine vectors as well as scalars

```
> x=seq(10)
> c(x,1:10,100)
 [1]   1   2   3   4   5   6   7   8   9  10   1   2   3   4   5   6   7   8   9  10 100
```

- Common arithmetic operations (including +, -, *, /, ^) and mathematical functions (e.g. sin(),cos(), log()) work element-wise on vectors, producing another vector

```
> a=c(1,2,3,4,5,6,7)
> a^2
[1]  1  4  9 16 25 36 49
```

- You can access elements of a vector using square brackets [] with the index of the element you want to retrieve. R uses 1-based indexing, meaning the first element is at index 1.

  ```
  fruits <- c("apple", "banana", "cherry", "date", "fig")
  ```

  - Single Element: Use a single index to access one element.

    ```
    first_fruit <- fruits[1]
    print(first_fruit)
    ```

  - Multiple Elements: Use a vector of indices to access multiple elements.

    ```
    selected_fruits <- fruits[c(1, 4)]
    print(selected_fruits)
    ```

  - Negative Index: Use negative indices to exclude elements. For example, fruits[-2] will exclude the second element.

    ```
    numbers <- c(10, 20, 30, 40, 50)
    # Exclude the second element
    numbers_exclude_second <- numbers[-2]
    print(numbers_exclude_second)  # Output: 10 30 40 50
    ```

- Generate random numbers and plot them.
  - `random_numbers <- runif(10, min = 1, max = 100)`
  - `print(random_numbers)`

Activity 03:

a. Create a vector named temperatures with values 30, 32, 31, 29, 28. Access and print the second and fourth elements.
b. Create two vectors named sales_Q1 and sales_Q2 with values 100, 150, 200 and 120, 180, 240, respectively. Calculate the total sales for each quarter and the difference between the two quarters.
c. Create a vector named grades with values 85, 72, 90, 65, 88. Find and print all grades greater than 80.

d. Generate a sequence named time_intervals from 0 to 24 with increments of 3. Plot this sequence.
e. Create a vector named prices with values 20, 30, 40, 50. Change the third value to 35 and print the modified vector.

Arithmetic Operations on Vectors

Operations on vectors are element-wise, meaning each element is operated on individually.

Example:

- v1 = c(1, 2, 3); v2 = c(4, 5, 6); v1 + v2 results in c(5, 7, 9).
- Element-wise multiplication: v1 * v2 results in c(4, 10, 18).

Activity 04:

a. Add two vectors x and y, and print the result.
   x <- c(5, 10, 15, 20)
   y <- c(1, 2, 3, 4)
b. Subtract vector y from vector x, and print the result.
c. Multiply two vectors a and b, and print the result.
   a <- c(2, 4, 6)
   b <- c(1, 3, 5)
d. Multiply vector c by a scalar value 10, and print the result.
e. Divide vector p by vector q, and print the result.
   p <- c(100, 200, 300)
   q <- c(2, 4, 5)
f. Apply the modulo operation(%%) to two vectors m and n to get the remainder of each division, and print the result.
   m <- c(10, 20, 30)
   n <- c(3, 5, 7)
g. Raise vector v to the power of 2, and print the result.
   v <- c(1, 2, 3, 4)

Summary Statistics

- Many functions summarize a data vector by producing a scalar from a vector.

```
> sum(a)
[1] 28
> length(a)
[1] 7
```

- Simple summary statistics (mean, median, standard deviation, variance, and quantiles) can be computed from numeric vectors using appropriately named functions
    - mean(data)
    - median(data)
    - sd(data)
    - var(data)
    - quantile(data)
- Use summary(data) for a comprehensive summary.

```
> x=rnorm(100)
> x
  [1]  0.563435076 -2.082429002  1.209769033 -0.187011523 -1.465571655  0.5
 [21]  1.710293526 -1.204407353 -0.185805150  1.170403935 -0.496001207  2.0
 [41] -1.256443743 -0.016674572  0.664074923  1.401153620  1.202129365  0.8
 [61]  0.709347947 -0.811596385  0.507797175  0.374891053  0.390989165  0.6
 [81] -1.289735598 -1.112787165 -0.139899752  0.584979667  2.083280116  1.4
> mean(x)
[1] 0.1119943
> sd(x)
[1] 1.034006
> var(x)
[1] 1.069167
> median(x)
[1] -0.0003030917
```

- Quantiles can be computed using the quantile() function.
- IQR() computes the Interquartile range (midspread or middle fifty).

```
> quantile(x)
          0%           25%           50%           75%          100%
-2.6507749756 -0.5060044006 -0.0003030917  0.7336578229  2.3648533090
> IQR(x)
[1] 1.239662
```

- The five-number summary (minimum, maximum, and quartiles) is given by fivenum(). A slightly extended summary is given by summary().

```
> fivenum(x)
[1] -2.6507749756 -0.5109344443 -0.0003030917  0.7361951337  2.3648533090
> summary(x)
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-2.6507750 -0.5060044 -0.0003031  0.1119943  0.7336578  2.3648533
```

Activity 05

a. You have the following numeric vector:
   data <- c(5, 10, 15, 20, 25, 30, 35)
   Compute the mean of the data vector and print the result.

b. You have the following numeric vector:
   data <- c(18, 22, 30, 40, 50)
   Compute the median of the data vector and print the result.
c. You have the following numeric vector:
   data <- c(5, 7, 10, 15, 20)
   Compute the standard deviation of the data vector and print the result.
d. You have the following numeric vector:
   data <- c(12, 18, 25, 30, 36)
   Compute the variance of the data vector and print the result.
e. You have the following numeric vector:
   data <- c(3, 5, 7, 9, 11, 13, 15)
   Compute the quantiles of the data vector and print the result.
f. You have the following numeric vector:
   data <- c(100, 200, 300, 400, 500)
   Use the summary function to obtain a comprehensive summary of the data vector. Print the result.
g. You have the following numeric vector:
   data <- c(2, 4, 6, 8, 10, 12)
   Compute the interquartile range (IQR) of the data vector and print the result.
h. You have the following numeric vector:
   data <- c(10, 20, 30, 40, 50, 60, 70)
   Compute the five-number summary of the data vector using the fivenum() function and print the result.
i. You have the following numeric vector:
   data <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
   Use the summary function to obtain an extended summary of the data vector. Print the result.
j. You have the following numeric vector:
   data <- c(7, 8, 9, 10, 12, 14, 16)
   Compute the quantiles of the data vector for the 25th, 50th, and 75th percentiles using the quantile() function.

File Handling

R can read and write files, making it easy to import/export data for analysis.

Example:

- To read a CSV file: data = read.csv("file.csv").

- To write data to a CSV file: write.csv(data, "output.csv").
- Check the working directory: getwd()#gives the current working directory
- Set the working directory: setwd("/path/to/directory")#sets the working directory or do this using the menu option File ->Change dir

```
> getwd()
[1] "/Users/kavinda"
> setwd("/Users/kavinda/Documents/UCSC/2021/2nd Sem/SCS2211 Lab 2/Practicals/Practical 1")
> getwd()
[1] "/Users/kavinda/Documents/UCSC/2021/2nd Sem/SCS2211 Lab 2/Practicals/Practical 1"
```

- List files: list.files()#lists the files in the current working directory
- Read the file: d = read.table("d.txt")#reads the data.
- Check summary statistics: summary(d)
- Create a scatter plot: plot(d)
- Extract the first column: col1 = d[1]
- Convert col1 to a vector: v1 = as.numeric(unlist(col1))
- Create histograms: hist(v1), hist(v1, 5), hist(v1, 100)

Activity 06

a. Download the file “d1.txt” from the LMS and save it in your current working directory. This is a file with two columns of data (V1 and V2) and 500 rows.
b. read the data in d1.txt file
c. Check the summary statistics d1.txt file
d. Draw a simple scatter plot
e. get the values of the first column (V1)

Scripting

Scripts help automate repetitive tasks by running multiple commands at once.

Example:

- A script might contain commands to load data, perform analysis, and generate plots.
- Run the script: source("analysis_script.R").
- Type the following in a file and save it as “myscript.txt”

myscript.txt

```
# A comment: this is a sample script.
y=c(12,15,28,17,18)
x=c(22,39,50,25,18)
mean(y)
mean(x)
plot(x,y)
```

- To run the script;> source("myscript.txt")

Activity 07

a. Open a text editor or RStudio and create a new file named myscript.txt.
b. Type the following commands into the file to load some example data, perform analysis, and generate a plot:

```
# Load necessary library
library(ggplot2)
# Create a sample dataset
data <- data.frame(
  Age = c(23, 45, 34, 25, 36, 50, 41),
  Height = c(167, 175, 160, 162, 180, 170, 165),
  Weight = c(55, 70, 60, 58, 75, 68, 62)
)
# Perform analysis: Calculate summary statistics
summary_stats <- summary(data)
print("Summary Statistics:")
print(summary_stats)
# Generate a scatter plot of Age vs Height
ggplot(data, aes(x = Age, y = Height)) +
  geom_point() +
  labs(title = "Scatter Plot of Age vs Height", x = "Age", y = "Height")
# Generate a histogram of Weight
ggplot(data, aes(x = Weight)) +
  geom_histogram(binwidth = 5, fill = "blue", color = "black") +
  labs(title = "Histogram of Weight", x = "Weight", y = "Frequency")
```

Data Types and Logical Comparisons

R supports various data types and logical comparisons for conditional checks.

Example:

- x = 10; y = 5; x > y returns TRUE.
- Logical operations: !(x > y) returns FALSE; (x > y) & (x == 10) returns TRUE.

There are many data types in R. The following note is a simple explanation of data types

available in R.

1. Numeric

These are generally positive and negative numbers with the decimal point

- 10
- -2
- 0.02
- 1.5e2 = 1.5 x 102 = 150
- 1e-7 =1 x 10-7 = 0.0000001
- can be also hexadecimal (starting with ‘0x’ or ‘0X’ followed by zero or more digits, and ‘a-f’ or ‘A-F’)
    - 0XF = 15
    - 0XFA = 15*16^1 + 10*16^0 = 250
    - Hexadecimal floating point constants are supported using C99 syntax, e.g.
    - 0x1.1p1

2. Integer

- created by using the qualifier L (e.g. : 123L)
- can be used with (non-complex) numbers given by hexadecimal or scientific notation
- Valid integer constants: try 1L, 0x10L, 1000000L, 1e6L
- However, if the value is not a valid integer, a warning is emitted and the numeric value created. (try 1.1L, 1e-3L, 0x1.1p-2)
- Try the following;
    - > typeof(2) #this is a “double” value
    - > typeof(2L) #this is an “integer” value

3. Logical

- either TRUE or FALSE

4. Complex

- A numeric constant immediately followed by i is regarded as an imaginary complex number. (e.g. 2i, 2+4.1i, 1e-2i)

5. String

- Delimited by a pair of single (‘’) or double (‘”’) quotes and can contain all other printable characters. Quotes and other special characters within strings are specified using escape sequences (e.g.: \n):

- Try the following;
    - > name = “Anne”
    - > name
    - > name = “Anne Mary”
    - > name

6. Special Types

   In addition, there are four special constants,

- NULL : used to indicate the empty object
- NA : for absent (“Not Available”) data values
- Inf : denotes infinity
- NaN: is not-a-number
- E.g..: Try the following; 1/0, 0/1, 0/0, -2/0