M.Sc. Degree

Unique ID: E7321008

Continuous Assessment - 1

CSC 540: Data Science with R

Average weight loss — Part-A

1). Declaring a variable 'before' and assigning it to a vector containing values of weight before diet.

- Declaring a variable 'after' and assigning it to a vector containing values of weight after diet.
- Calculating weight after the diet.

```
before ← c(78,72,78,79,105)
after ← c(67,65,79,70,93)
weight_loss ← (before - after)
```

```
#  78  72  78  79  105
# -67 -65 -79 -70  -93
```

```
weight_loss ← (11  7  -1  9  12)
```

```
sum = 0    count = 0
for (i in weight_loss) {
    sum = sum + i
    count = count + 1
}
```

# i = 11
i = 7
i = -1
i = 9
i = 12

Logic:

count = 0 + 1 = 1 — sum = 0 + 11 = 11

count = 1 + 1 = 2 — sum = 11 + 7 = 18

count = 2 + 1 = 3 — sum = 18 + (-1) = 17

count = 3 + 1 = 4 — sum = 17 + 9 = 26

count = 4 + 1 = **5** — sum = 26 + 12 = **38**

Count = 5 ; Sum = 38; average ← Sum/count ⇒ 38/5 = **7.6**

R 4.1.1 · D:/R_savefiles/

```
> # 1) To find Average weight loss
> # Unique ID : E7321008
>
> before
[1]  78  72  78  79 105
> after
[1] 67 65 79 70 93
> weight_loss
[1] 11  7 -1  9 12
> average_weight_loss
[1] "The Average weight loss is = 7.6"
> 
```

2) Employee Data Frame :

- Declaring a variable 'employee_id' and assigning it to a vector of values from 101 to 105.
- Declaring a variable 'employee_name' and assigning it to a vector of values ~~from~~ of employee names.
- Declaring a variable 'designation' and assigning it to a vector of values containing the designation of employees.
- Declaring a variable 'salary' and assigning it to a vector of values containing salaries of 5 employes.
- Creating a data frame with the above data and assigning it to a variable emp_details.
- Finding the maximum salary in the emp_details using max() function.
- To find the person with maximum salary, we use the following logic:

```
for (i in 1:length(emp_details$salary))
    if (emp_details$salary[i] == max_salary)
        print(emp_details$employee_name[i])
```

salary ← (60000, 55000, 25000, 50000, 52000)
position → 1, 2, 3, 4, 5

i = 60000 in position: 1

if (60000 == 60000) : True

employee_name[i] ⇒ "James"

∴ employee_name ← c("James", "Harvey", "Shini", "Jim", "Oliver")

position → 1, 2, 3, 4, 5

- To find the average of employee_salary:

Logic:

```
Sum = 0      # to find total sum of 'salary'
count = 0    # to find the no: of values in 'salary'
for (i in emp_details$salary)
    Sum = Sum + i
    count = count + 1
```

---

count = 0+1 = 1	Sum = 0+60000 = 60000	i = 60000
count = 1+1 = 2	Sum = 60000+55000 = 115000	i = 55000
count = 2+1 = 3	Sum = 115000+25000 = 140000	i = 25000
count = 3+1 = 4	Sum = 140000+50000 = 190000	i = 50000
count = 4+1 = **5**	Sum = 190000+52000 = **242000**	i = 52000
count = 5	Sum = 242000	

$$\text{average\_salary} = \text{Sum}/\text{count} = \frac{242000}{5} = 48400$$

```
> #Employee Data Frame
> # Unique ID : E7321008
> emp_details
  employee_id employee_name          designation salary
1         101          James                   HR  60000
2         102          Harry              Manager  55000
3         103          Shini Technical Assistant  25000
4         104            Jim   Senior Programmer  50000
5         105         Oliver            Team Lead  52000
> maximum_salary
[1] "Maximum Salary is :  60000"
> person_maximum_salary
[1] "Person with maximum salary :  James"
> average_salary
[1] "The average salary of the employees:  48400"
>
```

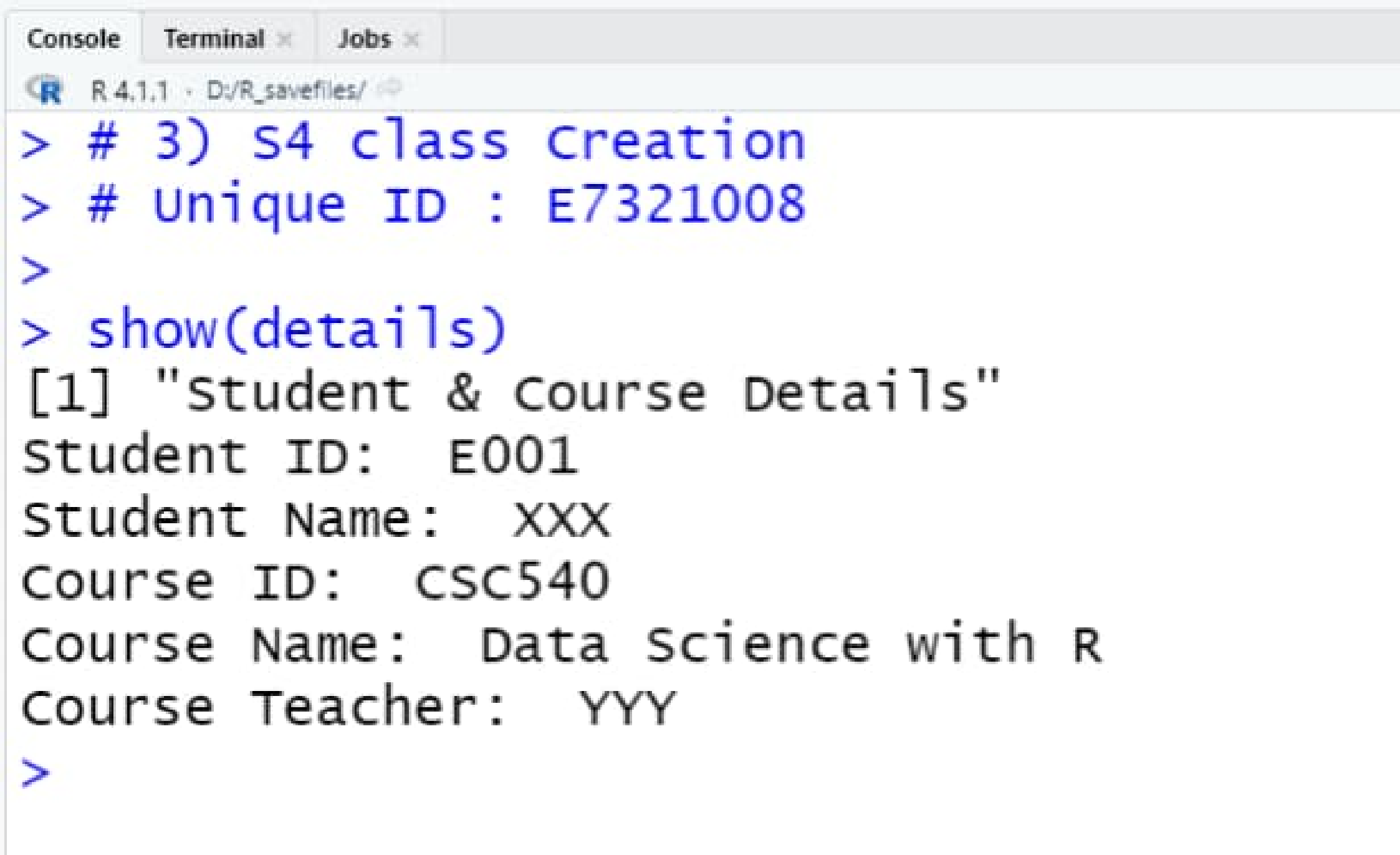
3) S4 class creation

- Creating an S4 class setclass() named 'course' and creating a list of variables, assigning to slots.
- Creating another S4 class setclass() named 'student' and creating a list of variables, assigning to slots.
- Creating a new class inheriting from the class 'course' and 'student'.
- Assigning the new class to variable 'details'.

- ~~Def~~ Defining a generic function setmethod("show", "student", function(object){

- setmethod("show", "student",
  function(object){
  cat("student ID", object@student_id)
  cat("Studentname", object@ student_name)
  cat("CourseID", object@ course_id)
  cat("Course Name", object@ course_name)
  cat("Course Teacher", object@ course_teacher)

- Accessing the slot values in student_id, student_name, course_id, course_name, course_teacher.
- Now using the variable inside the function show() to display accessed slot values. ie. show(details).


Console
Terminal
Jobs
R 4.1.1 · D:/R_savefiles/
> # 3) S4 class Creation
> # Unique ID : E7321008
>
> show(details)
[1] "Student & Course Details"
Student ID: E001
Student Name: XXX
Course ID: CSC540
Course Name: Data Science with R
Course Teacher: YYY
>

4) b) Printing a sequence

- Creating a sequence of numbers from 1 to 6.

  sequence_num ← seq(1:6)

- Declaring a variable 'num' for incrementation. num = 0

```
for (i in sequence_num){
  num = num + i
  print(num)
}
```

Output:

1
3
6
10
15
21

num = 0+1 = 1	i = 1
num = 1+2 = 3	i = 2
num = 3+3 = 6	i = 3
num = 6+4 = 10	i = 4
num = 10+5 = 15	i = 5
num = 15+6 = 21	i = 6

4) a) Printing numbers divisible by 3 from a sequence of number.

- Declaring a sequence of numbers from 1 to 50 incremented by 2. num ← seq(1, 50, 2)
- Declaring variable 'count' to count the total numbers divisible by 3. count ← 0

```
for (i in numbers) {
    if (i %% 3 == 0) {
        print (i)
        count = count + 1
    }
}
```

Logic:

i = 1	
1 %% 3 == 0 False	~~count = 0 + 1 = 1~~
i = 3	
3 %% 3 == 0 True	count = 0 + 1 = 1
i = 7	
7 %% 3 == 0 False	
i = 9	
9 %% 3 == 0 True	count = 1 + 1 = 2
i = 11	
11 %% 3 == 0 False	
i = 13	
13 %% 3 == 0 False	
i = 15	
15 %% 3 == 0 True	count = 2 + 1 = 3
i = 17	
17 %% 3 == 0 False	
i = 19	
19 %% 3 == 0 False	
i = 21	
21 %% 3 == 0 True	count = 3 + 1 = 4
.	.
.	.
.	.
45 %% 3 == 0 True	count = 7 + 1 = 8
47 %% 3 == 0 False	
49 %% 3 == 0 False	

Output:

3
9
15
21
27
33
39
45

Total numbers divisible by 3 : 8

- If the numbers between 1 to 50 incremented by 2 are divisible by 3 then we have to print those numbers & find the total count of those numbers.

4) c) Average height

height ← (5.5, 4.5, 6, NA, 7)

avg ← sum (na.omit (height)/length (height) # 23/5 ⇒ 4.6

na.omit(height) ⇒ omits 'NA' value & takes the rest of values.

length (height) ⇒ gives the length of height.

```
> # 4) a) Printing numbers divisible by 3 from a sequence of numbers
> # Unique ID : E7321008
>
> source("D:/SRET/R_Prog/Assessment_1/Sequence_divisible_by_3.R")
[1] "Numbers divisible by 3 :"
[1] 3
[1] 9
[1] 15
[1] 21
[1] 27
[1] 33
[1] 39
[1] 45
[1] "Total numbers Divisible by 3:  8"
>
```

```
> # 4) b) Printing a sequence
> # Unique ID : E7321008
>
> source("D:/SRET/R_Prog/Assessment_1/Sequence_numbers.R")
[1] 1
[1] 3
[1] 6
[1] 10
[1] 15
[1] 21
>
```

Console Terminal Jobs

R 4.1.1 · D:/R_savefiles/

```
> #4) c) Finding the average height of 5 persons
> # Unique ID : E7321008
> average_height
[1] "Average Height of the persons :  4.6"
>
```

5) Kaggle Data Set

- Downloading the employee_dataset from kaggle.
- Importing the 'employee_data'.
- Creating a data frame of 'employee_data' and assigning it to a variable 'df'.
- Finding the maximum age of employees in the data frame using max() function & assigning it to variable 'max_age'.
- Finding the ID of person with maximum salary. using the which() function inside the 'id' variable, which locates position of maximum salary into the variable 'id'.
- Now the position of the 'id' will be executed.
- Finding the average of salary using mean() function.
- Using which() function as mentioned above, to find the number of employees in healthy eating habit of 8.
- length (which (employee_data $ healthy_eating == 8))

  When the value in healthy_eating is equal to 8 the which() function, it selects that position & since we use length() function it locates and takes the position of all healthy_eating values which is equal to 8. & gives the total number.
- str(df) is used to display number of observations & variables.

```
> # 5) Kaggle Data
> # Unique ID : E7321008
>
> no_of_obs_var
[1] "Type str(df) to get no: of observations & variables "
> str(df)
'data.frame':   1000 obs. of  7 variables:
 $ ...1             : num  0 1 2 3 4 5 6 7 8 9 ...
 $ id               : num  0 1 2 3 4 5 6 7 8 9 ...
 $ groups           : chr  "A" "A" "A" "O" ...
 $ age              : num  36 55 61 29 34 42 53 41 47 31 ...
 $ healthy_eating   : num  5 3 8 3 6 5 4 8 5 4 ...
 $ active_lifestyle: num  5 5 1 6 2 3 6 6 6 8 ...
 $ salary           : num  2297 1134 4969 902 3574 ...
> maximum_age
[1] "Maximum Age of Employees :  64"
> maximum_salary_id
[1] "Maximum Salary ID :  607"
> average_salary
[1] "Average Salary Of Employees :  2227.461"
> Healthy_eating
[1] 73
> healthy_eating
[1] "Employees with healthy eating habit of 8 :  73"
>
```