NPTEL PMRF Live sessions on the course "Datascience For Engineers"

Teaching Assistant: Parvathy Neelakandan

July 29, 2025

0.1 NPTEL DSFE - Week 1 - Introduction to R

[1]:
```
a = 10
b = a * 10

num1 = 10
num2 = 15
sum = num1 + num2
print(c(num1, num2, sum))

print(c(a,b))
```

```
[1] 10 15 25
[1]  10 100
```

0.1.1 Setting and getting working directory

[2]:
```
#print(getwd())
```

[3]:
```
#setwd("Path to the folder I want to set the working directory")

setwd(getwd())
```

0.1.2 Saving and loading session data

[4]:
```
#This is the function call to save the workspace -> save.image()
save.image(file= "session.RData")

#Function to load workspace
load(file = "session.RData")

#Listing all vars
ls() #This will list all variables previously saved
rm(a) #remove a specific object
ls()
```

1. 'a' 2. 'b' 3. 'num1' 4. 'num2' 5. 'sum'

1. 'b' 2. 'num1' 3. 'num2' 4. 'sum'

[6]:
```
print(is.integer(x_int))
print(is.character(x_string))
print(is.logical(x_int))
```

```
[1] TRUE
[1] TRUE
[1] FALSE
```

0.1.4 Vectors

[7]:
```
num_vec = c(1, 2, 3, 4)
names_vec = c("Tom Cruise", "JUlia ROberts", "Tom Hanks")
bool_vec = c("True", "False")

named_vec = c("MI" = "Tom Cruise", "Nottinghill" = "Julia Roberts",␣
 ↪"DaVinci"="Tom Hanks")
names(bool_vec) = c("T", "F")

num_vec1 = num_vec + 1 #element-wise operation
num_vec2 = num_vec * 2

cat(num_vec1, "\n", num_vec2, "\n", num_vec1+num_vec2)

num_vec[2]
num_vec[c(3,4)]

num_vec[-3]
num_vec[c(-2, -4)]

num_vec2[num_vec2>6]

named_vec["DaVinci"]
```

```
2 3 4 5
 2 4 6 8
 4 7 10 13
```

2

1. 3 2. 4

1. 1 2. 2 3. 4

1. 1 2. 3

8

DaVinci: 'Tom Hanks'

[8]:
```
x = c(120, 240, 500)
```

```
x>50
x==240
x!=120

x>120 & x<500
```

1. TRUE 2. TRUE 3. TRUE

1. FALSE 2. TRUE 3. FALSE

1. FALSE 2. TRUE 3. TRUE

1. FALSE 2. TRUE 3. FALSE

[9]:
```
which(x>120)
which.max(x)
which.min(x)
```

1. 2 2. 3

3

1

0.1.5 Lists in R

[10]:
```
student_RollNum = c(1, 2, 3, 4)
student_Name = c("Tom", "Edward", "Paul")
student_list = list("roll_num" = student_RollNum, "name" = student_Name)
print(student_list)
print(student_list[[1]])
cat(student_list[[2]][2], "\n")
```

```
$roll_num
[1] 1 2 3 4

$name
[1] "Tom"    "Edward" "Paul"

[1] 1 2 3 4
Edward
```

[11]:
```
#Add new list
print("Add a new list")
student_list["Total_num_students"] = 4
print(student_list)
```

```
[1] "Add a new list"
$roll_num
[1] 1 2 3 4

$name
```

```
[1] "Tom"    "Edward" "Paul"

$Total_num_students
[1] 4
```

[12]:
```
print(student_list[['name']])
print(length(student_list)) #get the length
print(names(student_list))
print(unlist(student_list)) #flatten
print(length(unlist(student_list)))

student_list["Rank"] = list(c(1, 2, 3, 4))
print(student_list)
```

```
[1] "Tom"    "Edward" "Paul"
[1] 3
[1] "roll_num"           "name"               "Total_num_students"
         roll_num1           roll_num2           roll_num3           roll_num4
               "1"                 "2"                 "3"                 "4"
             name1               name2               name3 Total_num_students
             "Tom"             "Edward"              "Paul"                 "4"
[1] 8
$roll_num
[1] 1 2 3 4

$name
[1] "Tom"    "Edward" "Paul"

$Total_num_students
[1] 4

$Rank
[1] 1 2 3 4
```

0.1.6 Dataframes

[13]:
```
student_df = data.frame(
    "Name" = c("Tom", "Edward", "Paul"),
    "RollNum" = c(1, 2, 3))
student_df
```

Name	RollNum
Tom	1
Edward	2
Paul	3

[14]:
```
head(student_df)
tail(student_df)
nrow(student_df)
ncol(student_df)
dim(student_df)
summary(student_df)
```

Name	RollNum
Tom	1
Edward	2
Paul	3

Name	RollNum
Tom	1
Edward	2
Paul	3

3

2

1. 3 2. 2

```
     Name       RollNum
 Edward:1   Min.   :1.0
 Paul   :1   1st Qu.:1.5
 Tom    :1   Median :2.0
             Mean   :2.0
             3rd Qu.:2.5
             Max.   :3.0
```

[15]:
```
student_df[, "RollNum"]
student_df[1, 2]
student_df[,1:2]
```

1. 1 2. 2 3. 3

1

Name	RollNum
Tom	1
Edward	2
Paul	3

[16]:
```
student_df["Rank"] = c(2, 1, 3)
student_df["Home"] = c("US", "Canada", "Australia")
student_df
```

Name	RollNum	Rank	Home
Tom	1	2	US
Edward	2	1	Canada
Paul	3	3	Australia

[17]:
```
df2 = subset(student_df, Name == "Paul" | Home == "US")
df2
```

	Name	RollNum	Rank	Home
1	Tom	1	2	US
3	Paul	3	3	Australia

[18]:
```
student_df["Home"] = NULL
student_df
```

Name	RollNum	Rank
Tom	1	2
Edward	2	1
Paul	3	3

[19]:
```
student_df[[2]][1] = 4
student_df
```

Name	RollNum	Rank
Tom	4	2
Edward	2	1
Paul	3	3

0.1.7 Adding extra rows and columns

[20]:
```
student_df = rbind(student_df, data.frame(Name = "Alice", RollNum = 1, Rank = 4))
student_df
```

Name	RollNum	Rank
Tom	4	2
Edward	2	1
Paul	3	3
Alice	1	4

[21]:
```
student_df = cbind(student_df, Home = c("US", "Canada", "Australia",␣
↪"Netherlands"))
student_df
```

Name	RollNum	Rank	Home
Tom	4	2	US
Edward	2	1	Canada
Paul	3	3	Australia
Alice	1	4	Netherlands

0.1.8 Deleting rows and columns

[22]:
```
student_df2 = student_df[-3, -3]
student_df2

student_df3 = student_df[student_df["Name"] == "Paul",]
student_df3
```

	Name	RollNum	Home
1	Tom	4	US
2	Edward	2	Canada
4	Alice	1	Netherlands

	Name	RollNum	Rank	Home
3	Paul	3	3	Australia

[23]:
```
student_df3 = rbind(student_df3, data.frame(Name = "Sam", RollNum = 3, Rank=3,␣
↪Home="Australia" ))
student_df3
```

	Name	RollNum	Rank	Home
3	Paul	3	3	Australia
1	Sam	3	3	Australia

0.1.9 Recast

Melt -> When each measurement type needs different treatment

[24]:
```
scores = data.frame(
  Math = c(85, 95, 72, 88),
  Science = c(78, 92, 85, 90))
student_df4 = cbind(student_df, scores)
student_df4
```

Name	RollNum	Rank	Home	Math	Science
Tom	4	2	US	85	78
Edward	2	1	Canada	95	92
Paul	3	3	Australia	72	85
Alice	1	4	Netherlands	88	90

[25]:
```
library(reshape2)

m_df4 =  melt(student_df4,
                  id.vars = c("Name", "RollNum", "Rank", "Home"),
                  measure.vars = c("Math", "Science"),
                  variable.name = "Subject",
                  value.name = "Score")
m_df4
```

Name	RollNum	Rank	Home	Subject	Score
Tom	4	2	US	Math	85
Edward	2	1	Canada	Math	95
Paul	3	3	Australia	Math	72
Alice	1	4	Netherlands	Math	88
Tom	4	2	US	Science	78
Edward	2	1	Canada	Science	92
Paul	3	3	Australia	Science	85
Alice	1	4	Netherlands	Science	90

[26]:
```
d_df4 = dcast(m_df4, Name + RollNum + Rank + Home ~ Subject, value.var = "Score")
d_df4
```

Name	RollNum	Rank	Home	Math	Science
Edward	2	1	Canada	95	92
Paul	3	3	Australia	72	85
Tom	4	2	US	85	78
Alice	1	4	Netherlands	88	90

[27]:
```
d_df5 = dcast(m_df4, Subject ~ Name, value.var = "Score")
d_df5
```

Subject	Edward	Paul	Tom	Alice
Math	95	72	85	88
Science	92	85	78	90

recast() = melt() + dcast() in one function

What's the benefit of doing this?

0.1.10 Mutate function

[28]:
```
library(dplyr)
student_df6 = mutate(student_df4, bon_mark = Math + 1)
student_df6
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

Name	RollNum	Rank	Home	Math	Science	bon_mark
Tom	4	2	US	85	78	86
Edward	2	1	Canada	95	92	96
Paul	3	3	Australia	72	85	73
Alice	1	4	Netherlands	88	90	89

0.1.11 Join operation

[29]:
```
df1 = data.frame(ID = c(1, 2, 3),
                 Name = c("Alice", "Sam", "Ed"))

df2 = data.frame(ID = c(2, 3, 4),
                 Score = c(85, 90, 75))
```

[30]:
```
df1
```

ID	Name
1	Alice
2	Sam
3	Ed

[31]:
```
df2
```

ID	Score
2	85
3	90
4	75

[32]:
```
inner_join(df1, df2, by = "ID")
```

ID	Name	Score
2	Sam	85
3	Ed	90

[33]:
```
left_join(df1, df2, by = "ID")
```

ID	Name	Score
1	Alice	NA
2	Sam	85
3	Ed	90

[34]:
```
right_join(df1, df2, by = "ID")
```

ID	Name	Score
2	Sam	85
3	Ed	90
4	NA	75

0.1.12 Hierarchy of operations

[35]:
```
A1 = 5 + 3 * (2^4 / 4^2) - 6

A1
```

2

[36]:
```
A2 = 12 - 4 * (3^2 / 3) + 1
A2
```

1

[37]:
```
A3 = 10 + 2 * (4^3 / 2^2) - 5

A3
```

37

[38]:
```
A4 = 8 - 3 * (5^2 / 5) + 7
A4
```

0

0.1.13 Matrices

[39]:
```
m = matrix(1:9, nrow = 3, ncol = 3)
print(m)
```

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

[40]:
```
m[1,2]
```

4

[41]:
```
m[ ,2]
```

1. 4 2. 5 3. 6

[42]:
```
m[3, ]
```

1. 3 2. 6 3. 9

[43]:
```
m[1, 1] = 100
print(m)
```

```
     [,1] [,2] [,3]
[1,]  100    4    7
[2,]    2    5    8
[3,]    3    6    9
```

[44]:
```
m2 = m[-2, ]
print(m2)
```

```
     [,1] [,2] [,3]
[1,]  100    4    7
[2,]    3    6    9
```

[45]:
```
m3 = m[ , -1]
print(m3)
```

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

[46]:
```
dim(m)
nrow(m)
ncol(m)
```

1. 3 2. 3

3

3

[47]:
```
sum(m)
```

144

[48]:
```
prod(m)
```

36288000

[49]:
```
rowSums(m)
```

1. 111 2. 15 3. 18

[50]:
```
colSums(m)
```

1. 105 2. 15 3. 24

$$B = \begin{bmatrix} 2 & 5 & 8 \\ 3 & 6 & 9 \\ 1 & 4 & 7 \end{bmatrix}$$

- Change the element **9** to **10**
- Access the **first row and second column**
- List all the elements in the **third column** and **second row**
- Find the **sum** of all elements in `B`
- Multiply all elements using `prod()`

Using colon operator to create row matrix

[51]:
```
1:10
```

1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8 9. 9 10. 10

[52]:
```
m = matrix(1:6, nrow = 2)

m2 = rbind(m, c(7, 8, 9))
m2
```

1	3	5
2	4	6
7	8	9

[53]:
```
m3 = cbind(m2, c(10, 11, 12))
m3
```

1	3	5	10
2	4	6	11
7	8	9	12

[54]:
```
m_4 = rbind(m[1, ], c(-1, -2, -3), m[2, ])
m_4
```

1	3	5
-1	-2	-3
2	4	6

0.1.14 Matrix operations

[55]:
```
A = matrix(c(1, 2, 3, 4), nrow = 2)
B = matrix(c(5, 6, 7, 8), nrow = 2)
```

[56]:
```
A + B
```

6	10
8	12

[57]:
```
A - B
```

-4	-4
-4	-4

[58]:
```
A * B
```

5	21
12	32

[59]:
```
A / B
```

0.2000000 0.4285714
0.3333333 0.5000000

[60]:
```
A %*% B
```

23 31
34 46

0.1.15 Functions

[61]:
```
area_square = function(side) {
  return(side^2)
}
```

How do you call this?

[62]:
```
area_square(4)
area_square(7)
```

16

49

[63]:
```
rect_area_peri = function(length, width) {
  area = length * width
  perimeter = 2 * (length + width)

  return(list(area = area, perimeter = perimeter))
}
```

[64]:
```
result = rect_area_peri(5, 3)
print(result)
```

```
$area
[1] 15

$perimeter
[1] 16
```

[65]:
```
result['area']
```

$area = 15

[66]:
```
function(total_mark) total_mark + 10
```

r function (total_mark) total_mark + 10

[67]:
```
sapply(70:75, function(total_mark) total_mark + 10)
```

1. 80 2. 81 3. 82 4. 83 5. 84 6. 85

[68]:
```
m = matrix(1:9, nrow = 3)
print(m)
apply(m, 1, sum)
apply(m, 2, sum)
```

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

1. 12 2. 15 3. 18

1. 6 2. 15 3. 24

[69]:
```
my_list = list(a = 1:3, b = 4:6)

lapply(my_list, sum)
```

$a 6

$b 15

[70]:
```
mapply(sum, c(1, 2, 3), c(10, 20, 30))
```

1. 11 2. 22 3. 33

[71]:
```
id = c(1, 1, 2, 2, 3, 3)
values = c(5, 10, 2, 6, 3, 7)

tapply(values, id, sum)
```

1 15 **2** 8 **3** 10

0.1.16 Control structures

[72]:
```
x = 5

if (x > 0) {
  print("Positive")
} else {
  print("Non-positive")
}
```

```
[1] "Positive"
```

[73]:
```
for (i in 1:3)
{
  print(paste("Value of i is", i))
}
```

```
[1] "Value of i is 1"
[1] "Value of i is 2"
[1] "Value of i is 3"
```

[74]:
```r
i = 1
while (i <= 3) {
  print(paste("i is", i))
  i = i + 1
}
```

```
[1] "i is 1"
[1] "i is 2"
[1] "i is 3"
```

[75]:
```r
seq(1, 5)
seq(2, 10, by = 2)
```

1. 1 2. 2 3. 3 4. 4 5. 5

1. 2 2. 4 3. 6 4. 8 5. 10

0.1.17 Simple plots in R

[76]:
```r
x = 1:5
y = c(2, 4, 6, 8, 10)

plot(x, y, type = "l", col = "blue", main = "Line Plot")
```

[77]:
```r
x = c(1, 2, 3, 4, 5)
y = c(2, 1, 4, 3, 5)

plot(x, y, type = "p", col = "red", main = "Scatter Plot")
```

0.1.18 Practice Questions

1. Create a 3x3 matrix with values from 1 to 9.
2. Change the middle element (5) to 100.
3. Get the sum of each row using `apply()`.
4. Add a new row at the top: `c(10, 20, 30)`.
5. Add a new column at position 2: `c(5, 6, 7, 8)`.
6. Use a `for` loop to print squares of numbers from 1 to 5.

7. Write an `if-else` statement to check whether a number is even or odd.

8. Use a `while` loop to count from 1 to 5.
9. Write a function that takes one input (side) and returns the area of a square.

10. Write a function that takes length and width, and returns both area and perimeter.

11. Use an **inline function** with `sapply()` to square numbers from 1 to 5.
12. Create a **line plot** for `x = 1:5` and `y = 2 * x`.

13. Create a **bar plot** for `values = c(10, 20, 15)` with names `A, B, C`.

14. Create a **scatter plot** for `x = c(1, 2, 3)` and `y = c(3, 2, 1)`.
15. Create a 3x3 matrix and use `apply()` to get column sums.

16. Create a list and use `lapply()` to compute the mean of each item.

17. Use `mapply()` to add two vectors element-wise.

18. Use `tapply()` to get the sum of values grouped by an ID vector.
19. Write a function that returns "positive", "negative", or "zero" based on input.

20. Generate a sequence from 5 to 25 by 5s using `seq()`.

21. Create two small matrices and use `lapply()` to get row sums from a list of matrices.