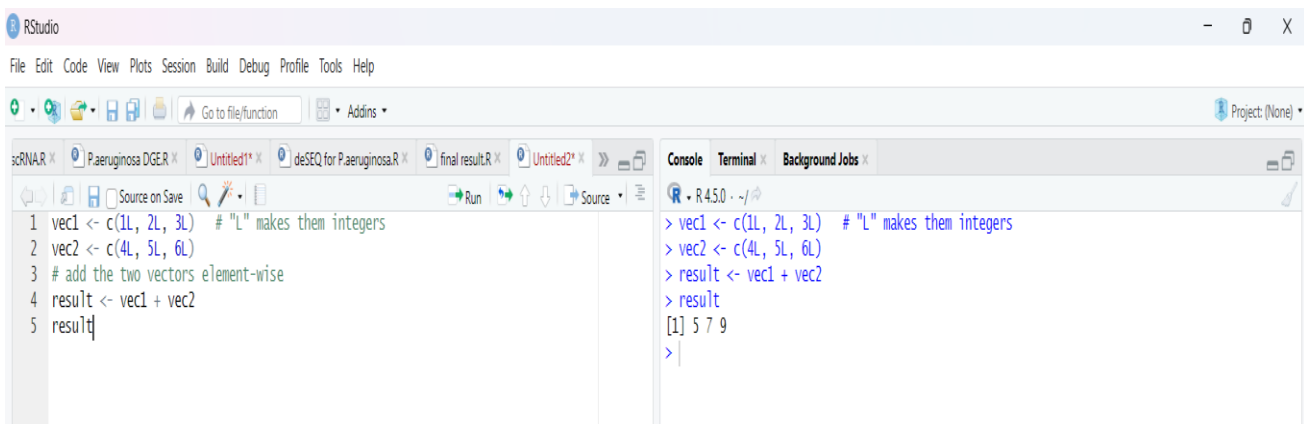


FUNDAMENTALS OF R Programming

1. How to add two vectors of integer's type and length 3?



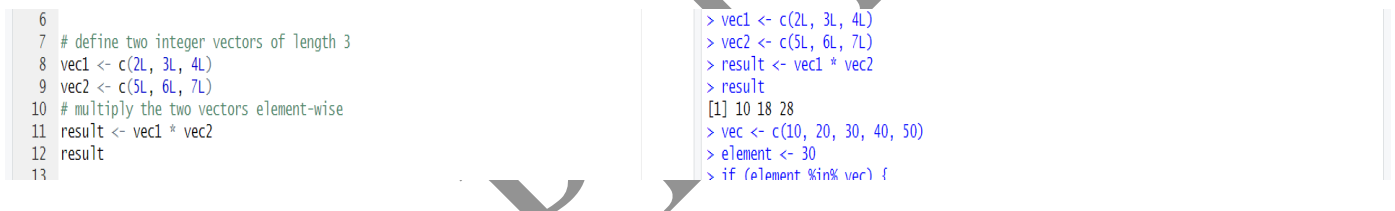
The screenshot shows the RStudio interface. The source editor on the left contains the following R code:

```
1 vec1 <- c(1L, 2L, 3L) # "L" makes them integers
2 vec2 <- c(4L, 5L, 6L)
3 # add the two vectors element-wise
4 result <- vec1 + vec2
5 result
```

The console on the right shows the execution of this code:

```
> vec1 <- c(1L, 2L, 3L) # "L" makes them integers
> vec2 <- c(4L, 5L, 6L)
> result <- vec1 + vec2
> result
[1] 5 7 9
>
```

2. How to multiply two vectors of integer's type and length 3?



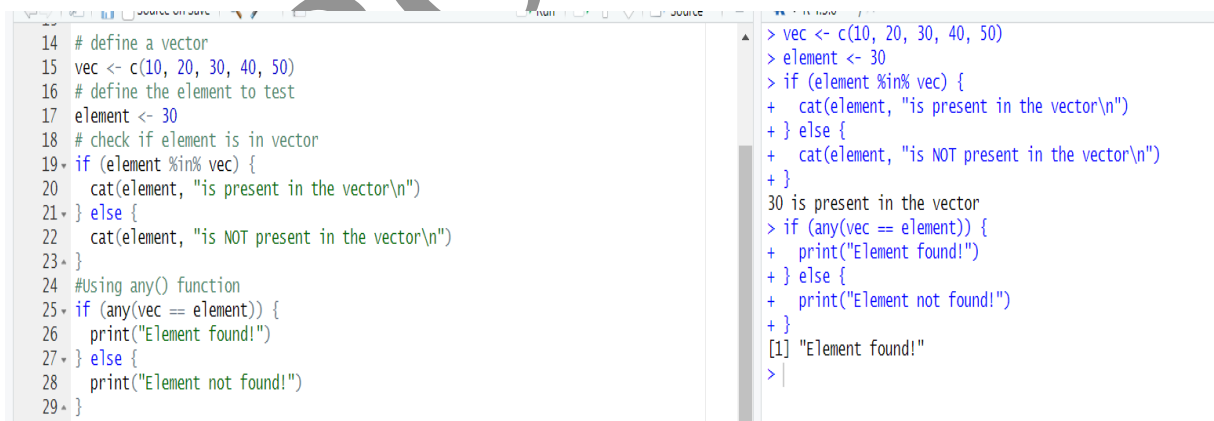
The screenshot shows the RStudio interface. The source editor on the left contains the following R code:

```
6
7 # define two integer vectors of length 3
8 vec1 <- c(2L, 3L, 4L)
9 vec2 <- c(5L, 6L, 7L)
10 # multiply the two vectors element-wise
11 result <- vec1 * vec2
12 result
13
```

The console on the right shows the execution of this code:

```
> vec1 <- c(2L, 3L, 4L)
> vec2 <- c(5L, 6L, 7L)
> result <- vec1 * vec2
> result
[1] 10 18 28
> vec <- c(10, 20, 30, 40, 50)
> element <- 30
> if (element %in% vec) {
```

3. Write an R program to test whether a given vector contains a specified element?



The screenshot shows the RStudio interface. The source editor on the left contains the following R code:

```
14 # define a vector
15 vec <- c(10, 20, 30, 40, 50)
16 # define the element to test
17 element <- 30
18 # check if element is in vector
19 if (element %in% vec) {
20   cat(element, "is present in the vector\n")
21 } else {
22   cat(element, "is NOT present in the vector\n")
23 }
24 #Using any() function
25 if (any(vec == element)) {
26   print("Element found!")
27 } else {
28   print("Element not found!")
29 }
```

The console on the right shows the execution of this code:

```
> vec <- c(10, 20, 30, 40, 50)
> element <- 30
> if (element %in% vec) {
+   cat(element, "is present in the vector\n")
+ } else {
+   cat(element, "is NOT present in the vector\n")
+ }
30 is present in the vector
> if (any(vec == element)) {
+   print("Element found!")
+ } else {
+   print("Element not found!")
+ }
[1] "Element found!"
>
```

4. Create a list containing a vector, a matrix and a list and add element at the end of the list?

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

Environment History Connections Tutorial

```
31 # create a vector
32 vec <- c(1, 2, 3)
33 # create a matrix
34 mat <- matrix(1:6, nrow = 2)
35 # create a sub-list
36 sub_list <- list(name = "R Programming", level = "Beginner")
37 # combine them into one list
38 my_list <- list(vec, mat, sub_list)
39 my_list
40 # add a new element at the end using append()
41 my_list <- append(my_list, "New Element")
42 my_list
43 # add multiple elements at the end using append()
44 my_list <- append(my_list, list("New Element 1", 99, c("A", "B", "C")))
45 my_list
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
```

Console

```
R - R 4.5.0 ~ /
> my_list <- list(vec, mat, sub_list)
> my_list
[[1]]
[1] 1 2 3

[[2]]
[1,] [1,] [2,] [3,]
[1,] 1 3 5
[2,] 2 4 6

[[3]]
[[3]]$name
[1] "R Programming"

[[3]]$level
[1] "Beginner"

> my_list <- append(my_list, "New Element")
> my_list
[[1]]
[1] 1 2 3

[[2]]
[1,] [1,] [2,] [3,]
[1,] 1 3 5
[2,] 2 4 6

[[3]]
[[3]]$name
[1] "R Programming"

[[3]]$level
[1] "Beginner"

[[4]]
[1] "New Element"

>
```

Files Plots Packages Help Viewer Presentation

30°C Mostly cloudy

Quick search

20-08-2025 17:35

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

Environment History Connections Tutorial

```
31 # create a vector
32 vec <- c(1, 2, 3)
33 # create a matrix
34 mat <- matrix(1:6, nrow = 2)
35 # create a sub-list
36 sub_list <- list(name = "R Programming", level = "Beginner")
37 # combine them into one list
38 my_list <- list(vec, mat, sub_list)
39 my_list
40 # add a new element at the end using append()
41 my_list <- append(my_list, "New Element")
42 my_list
43 # add multiple elements at the end using append()
44 my_list <- append(my_list, list("New Element 1", 99, c("A", "B", "C")))
45 my_list
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
```

Console

```
R - R 4.5.0 ~ /
[1] "R Programming"

[[3]]$level
[1] "Beginner"

[[4]]
[1] "New Element"

> my_list <- append(my_list, list("New Element 1", 99, c("A", "B", "C")))
> my_list
[[1]]
[1] 1 2 3

[[2]]
[1,] [1,] [2,] [3,]
[1,] 1 3 5
[2,] 2 4 6

[[3]]
[[3]]$name
[1] "R Programming"

[[3]]$level
[1] "Beginner"

[[4]]
[1] "New Element"

[[5]]
[1] "New Element 1"

[[6]]
[1] 99

[[7]]
[1] "A" "B" "C"

>
```

Files Plots Packages Help Viewer Presentation

30°C Mostly cloudy

Quick search

20-08-2025 17:36

5. Create three lists and combine the entire list into one list?

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Project: (None)

# create three lists
list1 <- list(1, 2, 3)
list2 <- list("A", "B", "C")
list3 <- list(TRUE, FALSE, TRUE)
# combine all three lists into one
combined_list <- c(list1, list2, list3)
combined_list
```

```
R 4.5.0 ~ /
> list1 <- list(1, 2, 3)
> list2 <- list("A", "B", "C")
> list3 <- list(TRUE, FALSE, TRUE)
> combined_list <- c(list1, list2, list3)
> combined_list
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

[[4]]
[1] "A"

[[5]]
[1] "B"

[[6]]
[1] "C"

[[7]]
[1] TRUE

[[8]]
[1] FALSE

[[9]]
[1] TRUE

> |
```

6. Find out the levels of factors of a given below vector? Y=C(1, 2, 3, 3, 4, NA, 3, 2, 4, 5, NA, 5)

```
# given vector
Y <- c(1, 2, 3, 3, 4, NA, 3, 2, 4, 5, NA, 5)
# convert to factor
Y_factor <- factor(Y)
# find levels of the factor
print(levels(Y_factor))
```

```
R 4.5.0 ~ /
> Y <- c(1, 2, 3, 3, 4, NA, 3, 2, 4, 5, NA, 5)
> Y_factor <- factor(Y)
> print(levels(Y_factor))
[1] "1" "2" "3" "4" "5"
> |
```

7. Write an R program to create an ordered factor from data consisting of the names of months?

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Project: (None)

# create a vector of month names
months_data <- c("March", "January", "April", "November", "January", "September")
# create an ordered factor with correct month order
ordered_months <- factor(
  months_data,
  levels = month.name, # built-in vector of full month names in correct order
  ordered = TRUE        # makes it an ordered factor
)
print(ordered_months)
# check the levels
print(levels(ordered_months))
```

```
R 4.5.0 ~ /
> months_data <- c("March", "January", "April", "November", "January", "September")
> ordered_months <- factor(
+   months_data,
+   levels = month.name, # built-in vector of full month names in correct order
+   ordered = TRUE      # makes it an ordered factor
+ )
> print(ordered_months)
[1] March January April November January September
12 Levels: January < February < March < April < May < June < July < ... < December
> print(levels(ordered_months))
[1] "January" "February" "March" "April" "May" "June"
[7] "July" "August" "September" "October" "November" "December"
> |
```

8. Create a data frame by taking following variables & find out the structure of data.

Name = "RX","TX","VX" "FX"

V1 = 10,30,40,90

V2 = 0.5, 0.8,1.5,1.9

Logi=TRUE, FALSE, TRUE, TRUE

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
.R x P.aeruginosa DGE.R x Untitled1* x deSEQ for P.aeruginosa.R x final result.R x R Assignment.R* x
75 # create data frame
76 df <- data.frame(
77   name=c("RX", "TX", "VX", "FX"),
78   V1=c(10, 30, 40, 90),
79   V2=c(0.5, 0.8, 1.5, 1.9),
80   Logi=c(TRUE, FALSE, TRUE, TRUE)
81 )
82 df
83 # find structure of data frame
84 str(df)
85
86
87
88
89
90
91
92
93
```

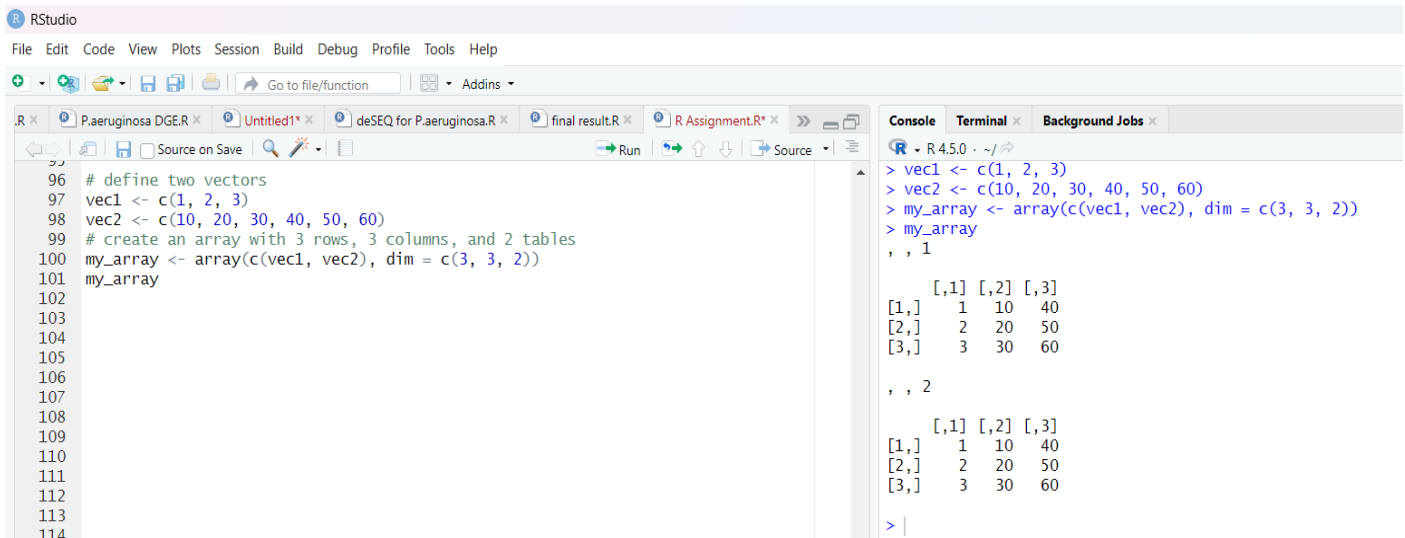
```
> df <- data.frame(
+   name=c("RX", "TX", "VX", "FX"),
+   V1=c(10, 30, 40, 90),
+   V2=c(0.5, 0.8, 1.5, 1.9),
+   Logi=c(TRUE, FALSE, TRUE, TRUE)
+ )
> df
  name V1  V2 Logi
1  RX 10 0.5  TRUE
2  TX 30 0.8 FALSE
3  VX 40 1.5  TRUE
4  FX 90 1.9  TRUE
> str(df)
'data.frame':   4 obs. of  4 variables:
 $ name: chr  "RX" "TX" "VX" "FX"
 $ V1  : num  10 30 40 90
 $ V2  : num   0.5 0.8 1.5 1.9
 $ Logi: logi   TRUE FALSE TRUE TRUE
> |
```

9. Write an R program to find the maximum and the minimum value of a given vector, nums = c(10, 20, 30, 40, 50, 60)

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
.R x P.aeruginosa DGE.R x Untitled1* x deSEQ for P.aeruginosa.R x final result.R x R Assignment.R* x
86 # given vector
87 nums <- c(10, 20, 30, 40, 50, 60)
88 # find maximum
89 max_value <- max(nums)
90 # find minimum
91 min_value <- min(nums)
92 # print results
93 cat("Maximum value:", max_value, "\n")
94 cat("Minimum value:", min_value, "\n")
95 |
96
```

```
> nums <- c(10, 20, 30, 40, 50, 60)
> max_value <- max(nums)
> min_value <- min(nums)
> cat("Maximum value:", max_value, "\n")
Maximum value: 60
> cat("Minimum value:", min_value, "\n")
Minimum value: 10
> |
```

10. How to create an array with three columns, three rows, and two "tables", taking two vectors as input to the array.



The screenshot shows the RStudio interface. The script editor on the left contains the following code:

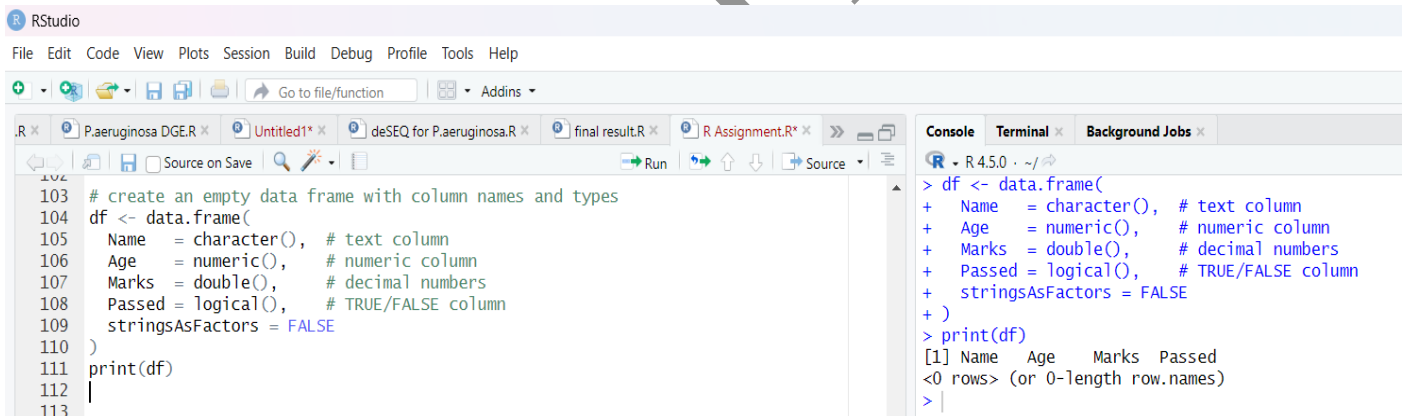
```
96 # define two vectors
97 vec1 <- c(1, 2, 3)
98 vec2 <- c(10, 20, 30, 40, 50, 60)
99 # create an array with 3 rows, 3 columns, and 2 tables
100 my_array <- array(c(vec1, vec2), dim = c(3, 3, 2))
101 my_array
```

The console on the right shows the output of the code:

```
> vec1 <- c(1, 2, 3)
> vec2 <- c(10, 20, 30, 40, 50, 60)
> my_array <- array(c(vec1, vec2), dim = c(3, 3, 2))
> my_array
, , 1
      [,1] [,2] [,3]
[1,]    1   10   40
[2,]    2   20   50
[3,]    3   30   60

, , 2
      [,1] [,2] [,3]
[1,]    1   10   40
[2,]    2   20   50
[3,]    3   30   60
> |
```

11. How to create an empty data frame.



The screenshot shows the RStudio interface. The script editor on the left contains the following code:

```
103 # create an empty data frame with column names and types
104 df <- data.frame(
105   Name = character(), # text column
106   Age = numeric(), # numeric column
107   Marks = double(), # decimal numbers
108   Passed = logical(), # TRUE/FALSE column
109   stringsAsFactors = FALSE
110 )
111 print(df)
112 |
113
```

The console on the right shows the output of the code:

```
> df <- data.frame(
+   Name = character(), # text column
+   Age = numeric(), # numeric column
+   Marks = double(), # decimal numbers
+   Passed = logical(), # TRUE/FALSE column
+   stringsAsFactors = FALSE
+ )
> print(df)
[1] Name Age Marks Passed
<0 rows> (or 0-length row.names)
> |
```

12. How to get the structure of following data frame.

```
exam_data = data.frame (
name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
)
```

```

113 # create the data frame
114 exam_data <- data.frame(
115   name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
116            'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
117   score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
118   attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
119   qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
120 )
121
122 # get structure of data frame
123 str(exam_data)
124
125
126
127

```

```

> exam_data <- data.frame(
+   name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
+            'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
+   score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
+   attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
+   qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
+ )
> str(exam_data)
'data.frame':   10 obs. of  4 variables:
 $ name      : chr  "Anastasia" "Dima" "Katherine" "James" ...
 $ score     : num  12.5  9 16.5 12 9 20 14.5 13.5 8 19
 $ attempts  : num   1  3  2  3  2  3  1  1  2  1
 $ qualify   : chr   "yes" "no" "yes" "no" "no" "yes" "yes" "no" "no" "yes"

```

13. How to get the statistical summary of the data of a following data frame?

```

exam_data = data.frame (
name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
score = c (12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
attempts = c (1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
)

```

```

125 # create the data frame
126 exam_data <- data.frame(
127   name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
128            'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
129   score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
130   attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
131   qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
132 )
133
134 # get statistical summary
135 summary(exam_data)
136
137
138
139
140
141

```

```

> exam_data <- data.frame(
+   name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
+            'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
+   score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
+   attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
+   qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
+ )
> summary(exam_data)
      name      score      attempts      qualify
Length:10      Min.   : 8.00      Min.   :1.00      Length:10
Class :character 1st Qu.: 9.75      1st Qu.:1.00      Class :character
Mode :character  Median:13.00      Median:2.00      Mode :character
                  Mean :13.40      Mean :1.90
                  3rd Qu.:16.00      3rd Qu.:2.75
                  Max.  :20.00      Max.  :3.00

```

14. How to extract specific column from a given data frame without using indexing method?

```

exam_data = data.frame(
name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),

```

```
score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
)
```

The screenshot shows the RStudio interface with the following code in the script editor:

```
# create the data frame
exam_data <- data.frame(
  name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
           'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
  score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
  attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
  qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
)
exam_data
#using the $ operator
exam_data$score
#using subset() function
subset(exam_data, select = score)
#using dplyr
library(dplyr)
exam_data %>% select(score)
```

The console output shows the data frame structure and the results of the subset and select operations:

```
> exam_data
  name score attempts qualify
1 Anastasia 12.5      1    yes
2 Dima      9.0      3    no
3 Katherine 16.5      2    yes
4 James    12.0      3    no
5 Emily     9.0      2    no
6 Michael  20.0      3    yes
7 Matthew  14.5      1    yes
8 Laura    13.5      1    no
9 Kevin     8.0      2    no
10 Jonas   19.0      1    yes

> exam_data$score
[1] 12.5  9.0 16.5 12.0  9.0 20.0 14.5 13.5  8.0 19.0

> subset(exam_data, select = score)
  score
1 12.5
2  9.0
3 16.5
4 12.0
5  9.0
6 20.0
7 14.5
8 13.5
9  8.0
10 19.0

> library(dplyr)

Attaching package: 'dplyr'

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

The screenshot shows the RStudio interface with the following code in the script editor:

```
# create the data frame
exam_data <- data.frame(
  name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
           'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
  score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
  attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
  qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
)
exam_data
#using the $ operator
exam_data$score
#using subset() function
subset(exam_data, select = score)
#using dplyr
library(dplyr)
exam_data %>% select(score)
```

The console output shows the data frame structure and the results of the subset and select operations:

```
> exam_data$score
[1] 12.5  9.0 16.5 12.0  9.0 20.0 14.5 13.5  8.0 19.0

> subset(exam_data, select = score)
  score
1 12.5
2  9.0
3 16.5
4 12.0
5  9.0
6 20.0
7 14.5
8 13.5
9  8.0
10 19.0

> library(dplyr)

Attaching package: 'dplyr'

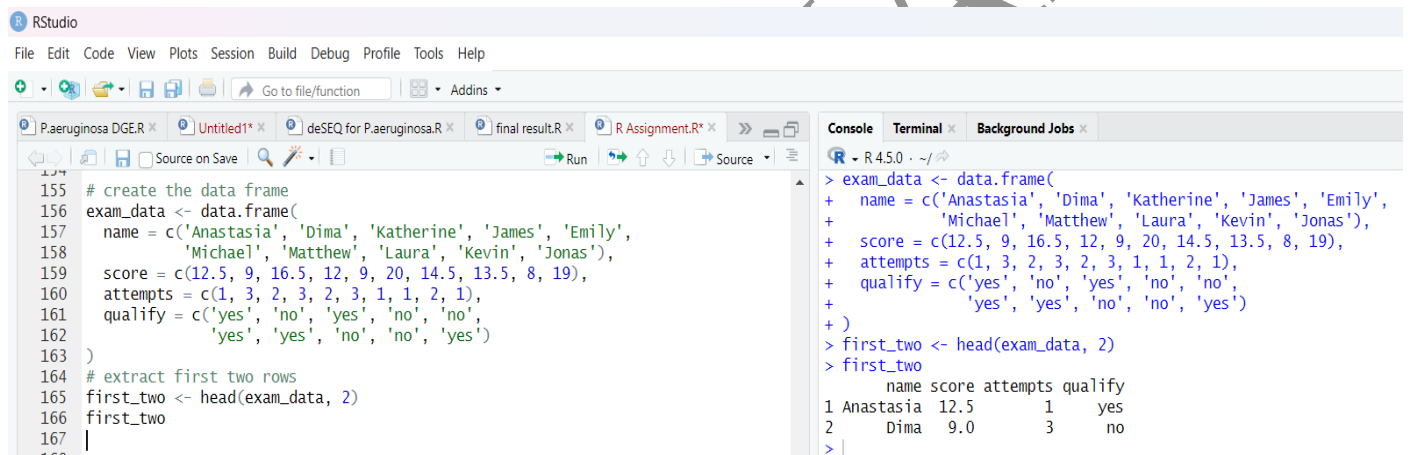
The following objects are masked from 'package:stats':
  filter, lag

The following objects are masked from 'package:base':
  intersect, setdiff, setequal, union

> exam_data %>% select(score)
  score
1 12.5
2  9.0
3 16.5
4 12.0
5  9.0
6 20.0
7 14.5
8 13.5
9  8.0
10 19.0
```

15. Write an R program to extract first two rows from a given data frame.

```
exam_data = data.frame(  
name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),  
score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),  
attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),  
qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')  
)
```



The screenshot shows the RStudio interface. The script editor on the left contains the R code for creating the data frame and extracting the first two rows. The console on the right shows the execution of the code, resulting in the first two rows of the data frame being displayed.

```
# create the data frame  
exam_data <- data.frame(  
  name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',  
            'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),  
  score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),  
  attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),  
  qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')  
)  
  
# extract first two rows  
first_two <- head(exam_data, 2)  
first_two
```

```
> exam_data <- data.frame(  
+   name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',  
+             'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),  
+   score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),  
+   attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),  
+   qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')  
+ )  
> first_two <- head(exam_data, 2)  
> first_two  
      name score attempts qualify  
1 Anastasia 12.5         1     yes  
2 Dima      9.0         3     no
```

16. How to extract 3rd and 5th rows with 1st and 3rd columns from a given data frame.

```
exam_data = data.frame(  
name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),  
score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),  
attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),  
qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')  
)
```



```

168 # create the data frame
169 exam_data <- data.frame(
170   name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
171            'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
172   score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
173   attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
174   qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
175 )
176
177 # extract 3rd and 5th rows, with 1st and 3rd columns
178 result <- exam_data[c(3, 5), c(1, 3)]
179 result
180
181

```

```

> exam_data <- data.frame(
+   name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
+            'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
+   score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
+   attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
+   qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
+ )
> result <- exam_data[c(3, 5), c(1, 3)]
> result
      name attempts
3 Katherine        2
5   Emily         2

```

17. Write a R program to add a new column in a given data frame.

```

exam_data = data.frame(
name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
)

```

```

181 # create the data frame
182 exam_data <- data.frame(
183   name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
184            'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
185   score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
186   attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
187   qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
188 )
189
190 # add a new column 'remarks'
191 exam_data$remarks <- c('Good', 'Poor', 'Excellent', 'Average', 'Poor',
192                       'Excellent', 'Good', 'Average', 'Poor', 'Excellent')
193 exam_data
194
195
196
197
198
199
200
201
202
203

```

```

> exam_data <- data.frame(
+   name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
+            'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
+   score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
+   attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
+   qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
+ )
> exam_data$remarks <- c('Good', 'Poor', 'Excellent', 'Average', 'Poor',
+                       'Excellent', 'Good', 'Average', 'Poor', 'Excellent')
> exam_data
      name score attempts qualify remarks
1 Anastasia 12.5         1    yes    Good
2   Dima      9.0         3     no    Poor
3 Katherine 16.5         2    yes Excellent
4   James  12.0         3     no   Average
5   Emily   9.0         2     no    Poor
6  Michael 20.0         3    yes Excellent
7  Matthew 14.5         1    yes    Good
8   Laura  13.5         1     no   Average
9   Kevin   8.0         2     no    Poor
10  Jonas  19.0         1    yes Excellent

```

18. How to add new row(s) to an existing data frame.

```

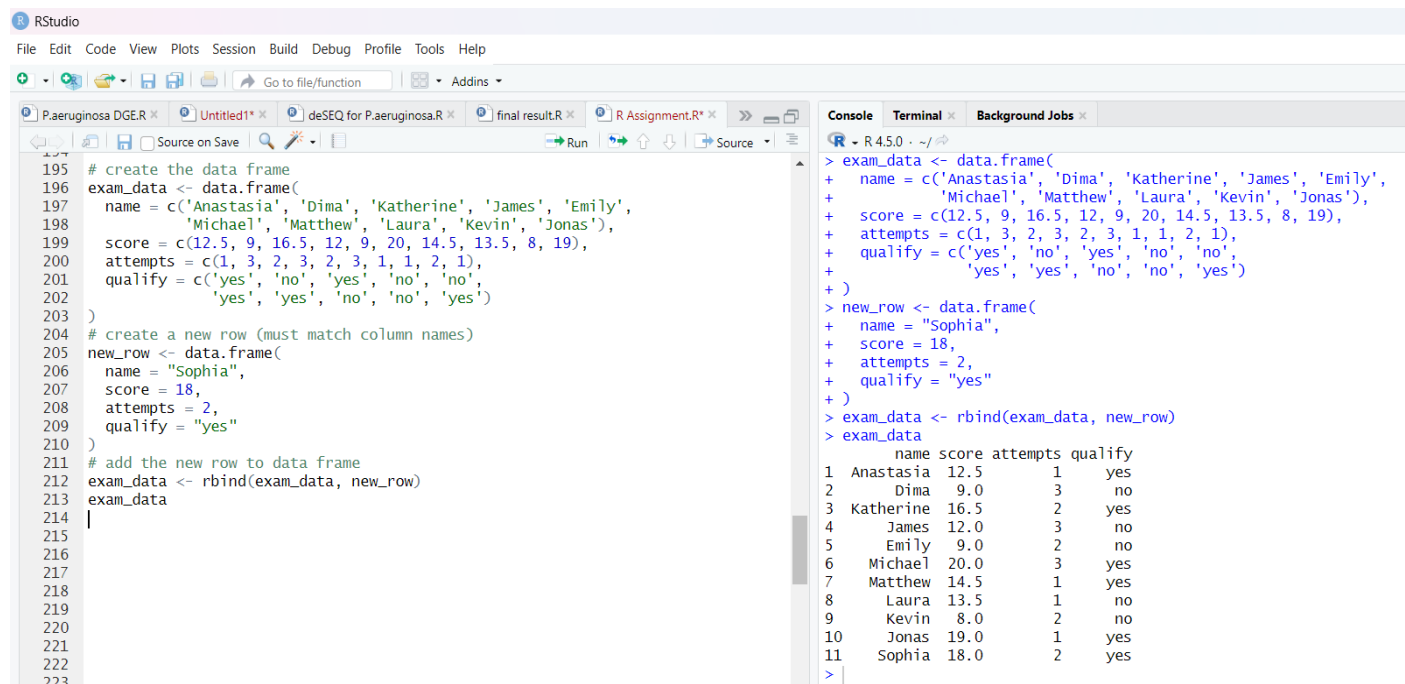
exam_data = data.frame(
name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),

```

```
attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
```

```
qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
```

```
)
```



The screenshot shows the RStudio interface. The source editor on the left contains R code for creating a data frame and adding a new row. The console on the right shows the execution of the code, resulting in a data frame with 11 rows and 4 columns: name, score, attempts, and qualify.

```
# create the data frame
exam_data <- data.frame(
  name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
            'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
  score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
  attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
  qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
)

# create a new row (must match column names)
new_row <- data.frame(
  name = "Sophia",
  score = 18,
  attempts = 2,
  qualify = "yes"
)

# add the new row to data frame
exam_data <- rbind(exam_data, new_row)
```

```
> exam_data <- data.frame(
+   name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
+           'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
+   score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
+   attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
+   qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
+ )
> new_row <- data.frame(
+   name = "Sophia",
+   score = 18,
+   attempts = 2,
+   qualify = "yes"
+ )
> exam_data <- rbind(exam_data, new_row)
> exam_data
  name score attempts qualify
1 Anastasia 12.5      1     yes
2 Dima      9.0      3     no
3 Katherine 16.5      2     yes
4 James    12.0      3     no
5 Emily     9.0      2     no
6 Michael  20.0      3     yes
7 Matthew  14.5      1     yes
8 Laura    13.5      1     no
9 Kevin     8.0      2     no
10 Jonas   19.0      1     yes
11 Sophia   18.0      2     yes
```

19. How to drop 3rd column(s) by name from a given data frame.

```
exam_data = data.frame(
```

```
name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
```

```
score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
```

```
attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
```

```
qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
```

```
)
```

```

215 # create the data frame
216 exam_data <- data.frame(
217   name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
218             'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
219   score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
220   attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
221   qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
222 )
223 # drop 3rd column using negative index
224 exam_data <- exam_data[, -3]
225 # print updated data frame
226 exam_data
227 |
228
229
230
231
232
233
234
235
236

```

```

> exam_data <- data.frame(
+   name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
+             'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
+   score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
+   attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
+   qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
+ )
> exam_data <- exam_data[, -3]
> exam_data
  name score qualify
1 Anastasia 12.5    yes
2 Dima      9.0    no
3 Katherine 16.5    yes
4 James    12.0    no
5 Emily     9.0    no
6 Michael  20.0    yes
7 Matthew  14.5    yes
8 Laura    13.5    no
9 Kevin     8.0    no
10 Jonas   19.0    yes
>

```

20. How to drop 2nd & 4th row(s) from a given data frame.

```

exam_data = data.frame(
name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
)

```

```

229 # create the data frame
230 exam_data <- data.frame(
231   name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
232             'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
233   score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
234   attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
235   qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
236 )
237 # drop 2nd and 4th rows using negative indexing
238 exam_data <- exam_data[-c(2, 4), ]
239 exam_data
240 |
241
242
243
244
245
246
247
248

```

```

> exam_data <- data.frame(
+   name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
+             'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
+   score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
+   attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
+   qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
+ )
> exam_data <- exam_data[-c(2, 4), ]
> exam_data
  name score attempts qualify
1 Anastasia 12.5      1    yes
3 Katherine 16.5      2    yes
5 Emily      9.0      2     no
6 Michael  20.0      3    yes
7 Matthew  14.5      1    yes
8 Laura    13.5      1     no
9 Kevin     8.0      2     no
10 Jonas   19.0      1    yes
>

```

DEBOPRIYA2320