

Statistics Practical

1. Use the internal/own database and run the following operators also explain the output.

data(), dim(), names(), View(), str(), ls(), rm()

#data() returns a list of currently loaded datasets or loads a dataset.

#dim() is used to get or set the dimension of the specified matrix, array or data frame.

#name() returns names of the columns

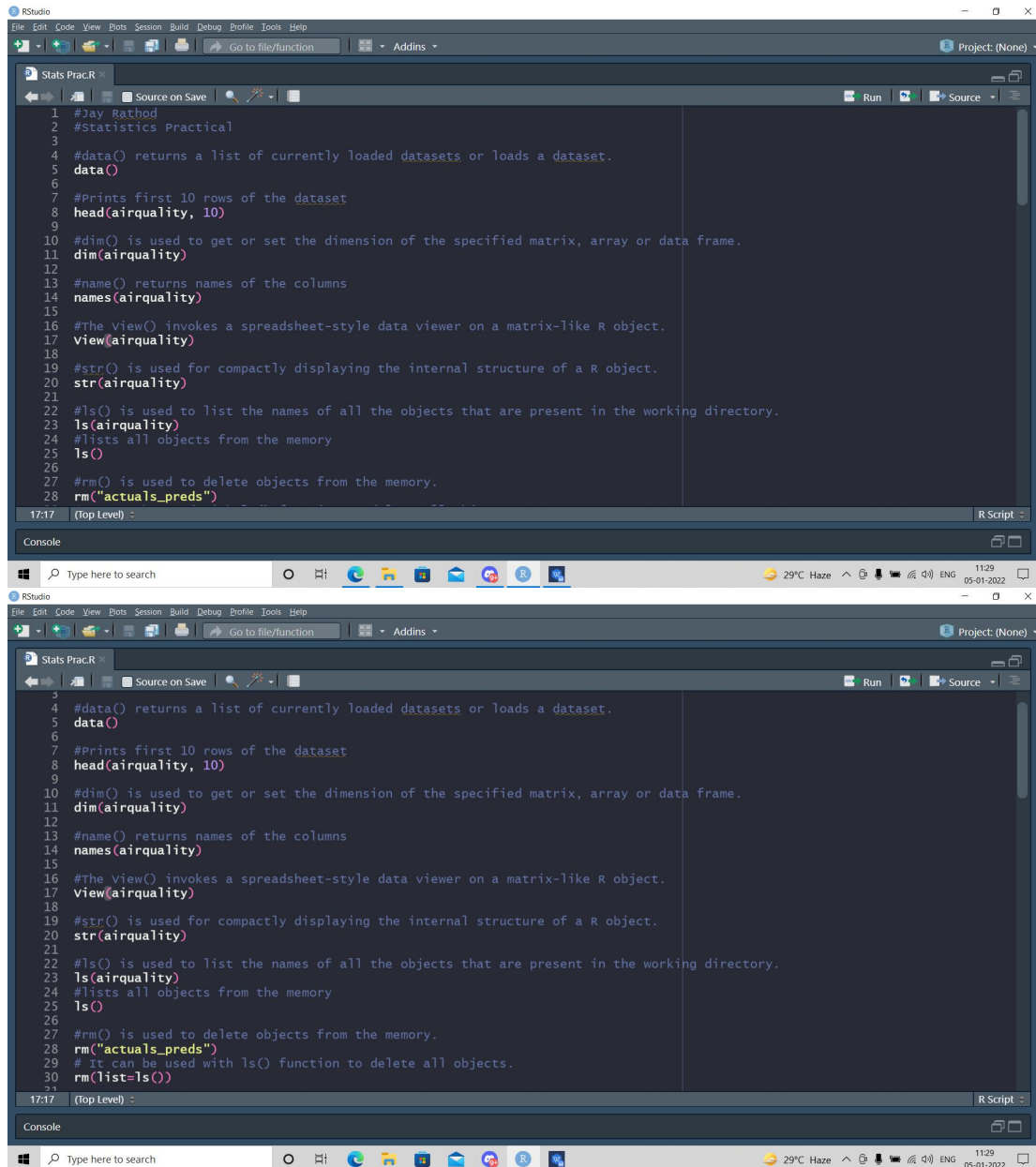
#The View() invokes a spreadsheet-style data viewer on a matrix-like R object.

#str() is used for compactly displaying the internal structure of a R object.

#ls() is used to list the names of all the objects that are present in the working directory.

#lists all objects from the memory

#rm() is used to delete objects from the memory.



The image displays two screenshots of the RStudio interface, showing R code being executed in a script editor. The code is organized into sections with comments explaining the functions used.

Top Screenshot:

```
1 #Jay Rathod
2 #Statistics Practical
3
4 #data() returns a list of currently loaded datasets or loads a dataset.
5 data()
6
7 #Prints first 10 rows of the dataset
8 head(airquality, 10)
9
10 #dim() is used to get or set the dimension of the specified matrix, array or data frame.
11 dim(airquality)
12
13 #name() returns names of the columns
14 names(airquality)
15
16 #The view() invokes a spreadsheet-style data viewer on a matrix-like R object.
17 View(airquality)
18
19 #str() is used for compactly displaying the internal structure of a R object.
20 str(airquality)
21
22 #ls() is used to list the names of all the objects that are present in the working directory.
23 ls(airquality)
24 #lists all objects from the memory
25 ls()
26
27 #rm() is used to delete objects from the memory.
28 rm("actuals_preds")
```

Bottom Screenshot:

```
3
4 #data() returns a list of currently loaded datasets or loads a dataset.
5 data()
6
7 #Prints first 10 rows of the dataset
8 head(airquality, 10)
9
10 #dim() is used to get or set the dimension of the specified matrix, array or data frame.
11 dim(airquality)
12
13 #name() returns names of the columns
14 names(airquality)
15
16 #The view() invokes a spreadsheet-style data viewer on a matrix-like R object.
17 View(airquality)
18
19 #str() is used for compactly displaying the internal structure of a R object.
20 str(airquality)
21
22 #ls() is used to list the names of all the objects that are present in the working directory.
23 ls(airquality)
24 #lists all objects from the memory
25 ls()
26
27 #rm() is used to delete objects from the memory.
28 rm("actuals_preds")
29 # It can be used with ls() function to delete all objects.
30 rm(list=ls())
```

Output:

The screenshot shows the RStudio interface. The top pane displays the 'R data sets' list, which includes various datasets such as AirPassengers, BJSales, BJSales.lead, BOD, CO2, ChickWeight, DNase, EuStockMarkets, Formaldehyde, HairEyeColor, Harman22.cor, Harman74.cor, Indometh, InsectSprays, JohnsonJohnson, LakeHuron, LifeCycleSavings, Loblolly, Nile, Orange, OrchardSprays, PlantGrowth, Puromycin, Seatbelts, Theoph, Titanic, and ToothGrowth. The bottom pane shows the 'airquality' dataset loaded, displaying a table with columns: Ozone, Solar.R, Wind, Temp, Month, and Day. The table shows 16 rows of data, with the first row being (1, 41, 190, 7.4, 67, 5, 1). The status bar at the bottom indicates 'Showing 1 to 17 of 153 entries, 6 total columns'.

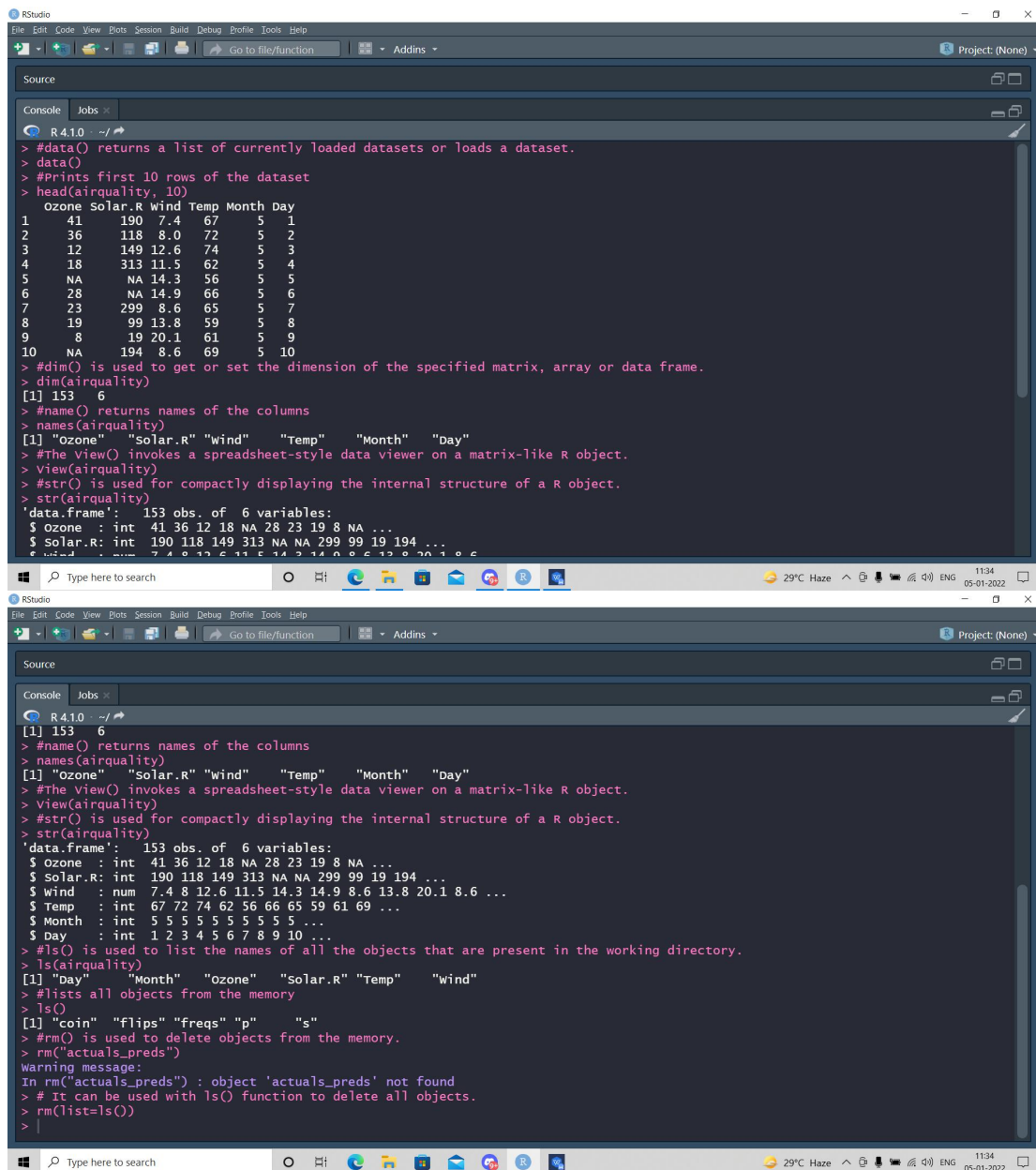
R data sets in package 'datasets':

- AirPassengers: Monthly Airline Passenger Numbers 1949-1960
- BJSales: Sales Data with Leading Indicator
- BJSales.lead (BJSales): Sales Data with Leading Indicator
- BOD: Biochemical Oxygen Demand
- CO2: Carbon Dioxide Uptake in Grass Plants
- ChickWeight: Weight versus age of chicks on different diets
- DNase: Elisa assay of DNase
- EuStockMarkets: Daily Closing Prices of Major European Stock Indices, 1991-1998
- Formaldehyde: Determination of Formaldehyde
- HairEyeColor: Hair and Eye Color of Statistics Students
- Harman22.cor: Harman Example 2.3
- Harman74.cor: Harman Example 7.4
- Indometh: Pharmacokinetics of Indomethacin
- InsectSprays: Effectiveness of Insect Sprays
- JohnsonJohnson: Quarterly Earnings per Johnson & Johnson Share
- LakeHuron: Level of Lake Huron 1875-1972
- LifeCycleSavings: Intercountry Life-Cycle Savings Data
- Loblolly: Growth of Loblolly pine trees
- Nile: Flow of the River Nile
- Orange: Growth of Orange Trees
- OrchardSprays: Potency of Orchard Sprays
- PlantGrowth: Results from an Experiment on Plant Growth
- Puromycin: Reaction Velocity of an Enzymatic Reaction
- Seatbelts: Road Casualties in Great Britain 1969-84
- Theoph: Pharmacokinetics of Theophylline
- Titanic: Survival of passengers on the Titanic
- ToothGrowth: The Effect of Vitamin C on Tooth Growth in Guinea Pigs

airquality

	Ozone	Solar.R	Wind	Temp	Month	Day
1	41	190	7.4	67	5	1
2	36	118	8.0	72	5	2
3	12	149	12.6	74	5	3
4	18	313	11.5	62	5	4
5	NA	NA	14.3	56	5	5
6	28	NA	14.9	66	5	6
7	23	299	8.6	65	5	7
8	19	99	13.8	59	5	8
9	8	19	20.1	61	5	9
10	NA	194	8.6	69	5	10
11	7	NA	6.9	74	5	11
12	16	256	9.7	69	5	12
13	11	290	9.2	66	5	13
14	14	274	10.9	68	5	14
15	18	65	13.2	58	5	15
16	14	334	11.5	64	5	16

Showing 1 to 17 of 153 entries, 6 total columns



The image displays two screenshots of the RStudio interface, specifically the Console window, showing R commands and their output for the 'airquality' dataset.

Top Screenshot:

```

R 4.1.0 ~/
> #data() returns a list of currently loaded datasets or loads a dataset.
> data()
> #Prints first 10 rows of the dataset
> head(airquality, 10)
  ozone solar.R wind temp month day
1    41     190   7.4   67     5    1
2    36     118   8.0   72     5    2
3    12     149  12.6   74     5    3
4    18     313  11.5   62     5    4
5    NA      NA  14.3   56     5    5
6    28      NA  14.9   66     5    6
7    23     299   8.6   65     5    7
8    19     299  13.8   59     5    8
9     8      19  20.1   61     5    9
10   NA     194   8.6   69     5   10
> #dim() is used to get or set the dimension of the specified matrix, array or data frame.
> dim(airquality)
[1] 153  6
> #name() returns names of the columns
> names(airquality)
[1] "ozone" "solar.R" "wind" "temp" "month" "day"
> #The View() invokes a spreadsheet-style data viewer on a matrix-like R object.
> view(airquality)
> #str() is used for compactly displaying the internal structure of a R object.
> str(airquality)
'data.frame': 153 obs. of  6 variables:
 $ ozone : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ solar.R: int 190 118 149 313 NA NA 299 99 19 194 ...
 $ wind  : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ temp  : int  67 72 74 62 56 66 65 59 61 69 ...
 $ month : int  5 5 5 5 5 5 5 5 5 5 ...
 $ day   : int  1 2 3 4 5 6 7 8 9 10 ...

```

Bottom Screenshot:

```

R 4.1.0 ~/
[1] 153  6
> #name() returns names of the columns
> names(airquality)
[1] "ozone" "solar.R" "wind" "temp" "month" "day"
> #The View() invokes a spreadsheet-style data viewer on a matrix-like R object.
> view(airquality)
> #str() is used for compactly displaying the internal structure of a R object.
> str(airquality)
'data.frame': 153 obs. of  6 variables:
 $ ozone : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ solar.R: int 190 118 149 313 NA NA 299 99 19 194 ...
 $ wind  : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ temp  : int  67 72 74 62 56 66 65 59 61 69 ...
 $ month : int  5 5 5 5 5 5 5 5 5 5 ...
 $ day   : int  1 2 3 4 5 6 7 8 9 10 ...
> #ls() is used to list the names of all the objects that are present in the working directory.
> ls(airquality)
[1] "day" "month" "ozone" "solar.R" "temp" "wind"
> #lists all objects from the memory
> ls()
[1] "coin" "flips" "freqs" "p" "s"
> #rm() is used to delete objects from the memory.
> rm("actuals_preds")
warning message:
In rm("actuals_preds") : object 'actuals_preds' not found
> # It can be used with ls() function to delete all objects.
> rm(list=ls())
>

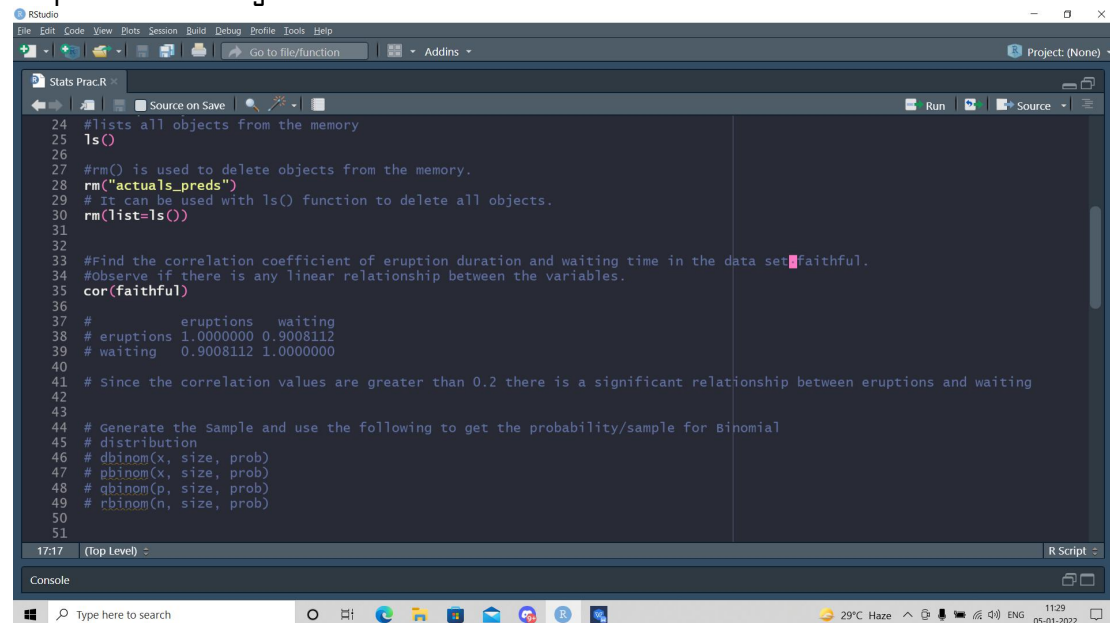
```

2. Find the correlation coefficient of eruption duration and waiting time in the data set faithful.

Observe if there is any linear relationship between the variables.

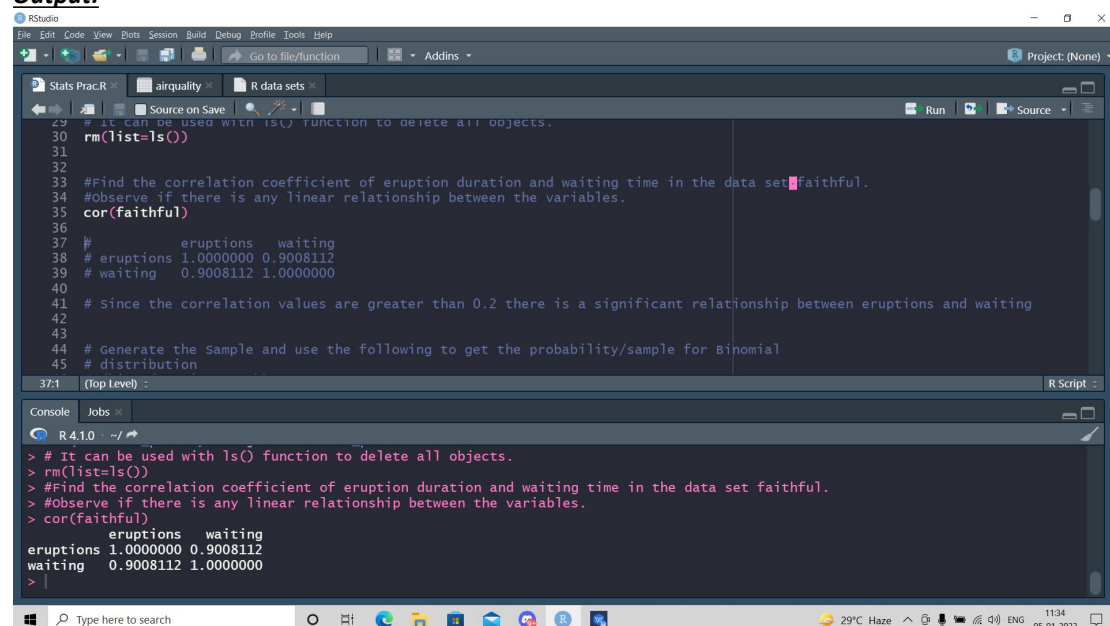
```
#           eruptions waiting
# eruptions 1.00000000 0.9008112
# waiting   0.9008112 1.0000000
```

Since the correlation values are greater than 0.2 there is a significant relationship between eruptions and waiting



```
24 #lists all objects from the memory
25 ls()
26
27 #rm() is used to delete objects from the memory.
28 rm("actuals_preds")
29 # It can be used with ls() function to delete all objects.
30 rm(list=ls())
31
32
33 #Find the correlation coefficient of eruption duration and waiting time in the data set faithful.
34 #observe if there is any linear relationship between the variables.
35 cor(faithful)
36
37 #           eruptions waiting
38 # eruptions 1.00000000 0.9008112
39 # waiting   0.9008112 1.0000000
40
41 # Since the correlation values are greater than 0.2 there is a significant relationship between eruptions and waiting
42
43
44 # Generate the Sample and use the following to get the probability/sample for Binomial
45 # distribution
46 # dbinom(x, size, prob)
47 # pbinom(x, size, prob)
48 # qbinom(p, size, prob)
49 # rbinom(n, size, prob)
50
51
```

Output:



```
29 # It can be used with ls() function to delete all objects.
30 rm(list=ls())
31
32
33 #Find the correlation coefficient of eruption duration and waiting time in the data set faithful.
34 #observe if there is any linear relationship between the variables.
35 cor(faithful)
36
37 #           eruptions waiting
38 # eruptions 1.00000000 0.9008112
39 # waiting   0.9008112 1.0000000
40
41 # Since the correlation values are greater than 0.2 there is a significant relationship between eruptions and waiting
42
43
44 # Generate the Sample and use the following to get the probability/sample for Binomial
45 # distribution
```

Console

```
> # It can be used with ls() function to delete all objects.
> rm(list=ls())
> #Find the correlation coefficient of eruption duration and waiting time in the data set faithful.
> #observe if there is any linear relationship between the variables.
> cor(faithful)
           eruptions waiting
eruptions 1.00000000 0.9008112
waiting   0.9008112 1.0000000
>
```

3. Generate the Sample and use the following to get the probability/sample for Binomial distribution

`dbinom(x, size, prob)`

`pbinom(x, size, prob)`

`qbinom(p, size, prob)`

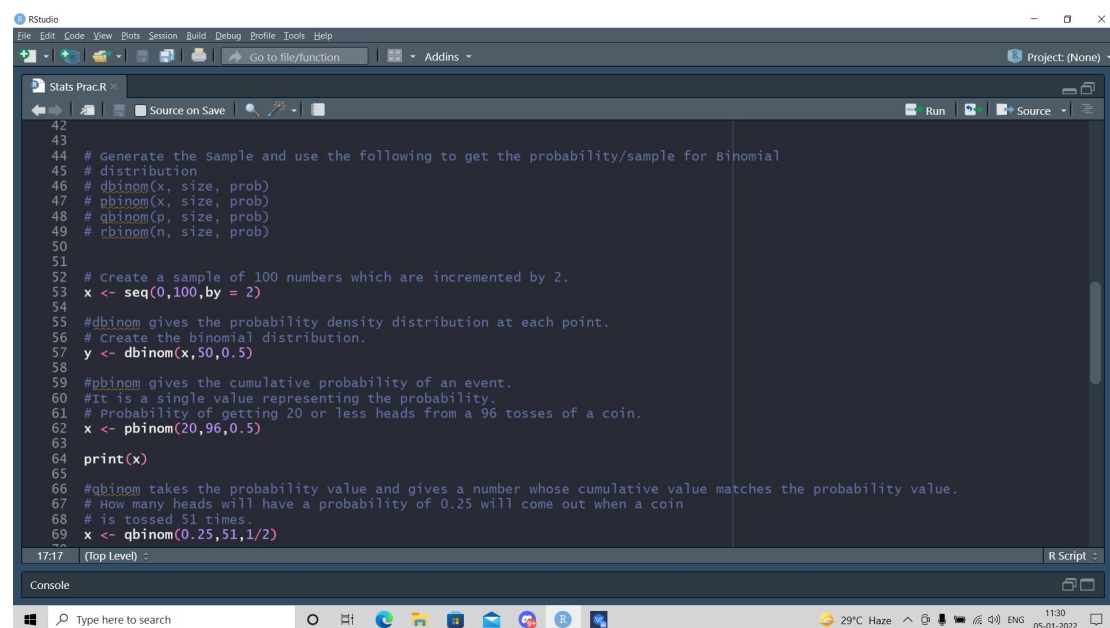
`rbinom(n, size, prob)`

`#dbinom` gives the probability density distribution at each point.

`#pbinom` gives the cumulative probability of an event.

`#qbinom` takes the probability value and gives a number whose cumulative value matches the probability value.

`#rbinom` generates required number of random values of given probability from a given sample



```
42
43
44 # Generate the Sample and use the following to get the probability/sample for Binomial
45 # distribution
46 # dbinom(x, size, prob)
47 # pbinom(x, size, prob)
48 # qbinom(p, size, prob)
49 # rbinom(n, size, prob)
50
51
52 # Create a sample of 100 numbers which are incremented by 2.
53 x <- seq(0,100,by = 2)
54
55 #dbinom gives the probability density distribution at each point.
56 # create the binomial distribution.
57 y <- dbinom(x,50,0.5)
58
59 #pbinom gives the cumulative probability of an event.
60 #It is a single value representing the probability.
61 # Probability of getting 20 or less heads from a 96 tosses of a coin.
62 x <- pbinom(20,96,0.5)
63
64 print(x)
65
66 #qbinom takes the probability value and gives a number whose cumulative value matches the probability value.
67 # How many heads will have a probability of 0.25 will come out when a coin
68 # is tossed 51 times.
69 x <- qbinom(0.25,51,1/2)
```

```

63 print(x)
64
65
66 #qbinom takes the probability value and gives a number whose cumulative value matches the probability value.
67 # How many heads will have a probability of 0.25 will come out when a coin
68 # is tossed 51 times.
69 x <- qbinom(0.25,51,1/2)
70
71 print(x)
72
73 #rbinom generates required number of random values of given probability from a given sample
74 # Find 8 random values from a sample of 150 with probability of 0.4.
75 x <- rbinom(8,150,0.4)
76
77 print(x)
78
79
80 # Generate the random sample between 4 to 8 where
81 # i. replacement is allowed
82 #means numbers can be repeated
84
85 sample(4:8,10, replace=TRUE)
86
87
88 # ii. replacement is not allowed
89 #means numbers cannot be repeated
90 sample(4:8,5, replace=FALSE)

```

17:17 (top Level) R Script

Console

Output:

```

R 4.1.0 ~/
> #Observe if there is any linear relationship between the variables.
> cor(faithful)
eruptions waiting
eruptions 1.0000000 0.9008112
waiting 0.9008112 1.0000000
> # Create a sample of 100 numbers which are incremented by 2.
> x <- seq(0,100,by = 2)
> #dbinom gives the probability density distribution at each point.
> # Create the binomial distribution.
> y <- dbinom(x,50,0.5)
> #pbinom gives the cumulative probability of an event.
> #it is a single value representing the probability.
> # Probability of getting 20 or less heads from a 96 tosses of a coin.
> x <- pbinom(20,96,0.5)
> print(x)
[1] 3.659684e-09
> #qbinom takes the probability value and gives a number whose cumulative value matches the probability value.
> # How many heads will have a probability of 0.25 will come out when a coin
> # is tossed 51 times.
> x <- qbinom(0.25,51,1/2)
> print(x)
[1] 23
> #rbinom generates required number of random values of given probability from a given sample
> # Find 8 random values from a sample of 150 with probability of 0.4.
> x <- rbinom(8,150,0.4)
> print(x)
[1] 56 61 68 55 71 64 54 59
>

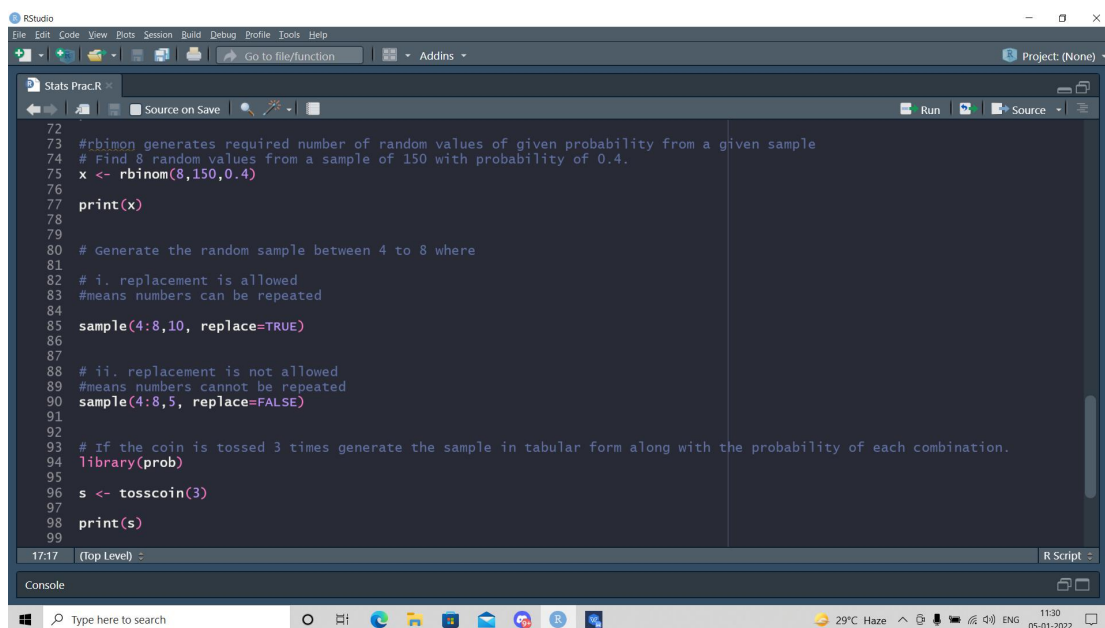
```


4. Generate the random sample between 4 to 8 where

- i. replacement is allowed
- ii. replacement is not allowed

i. replacement is allowed
#means numbers can be repeated

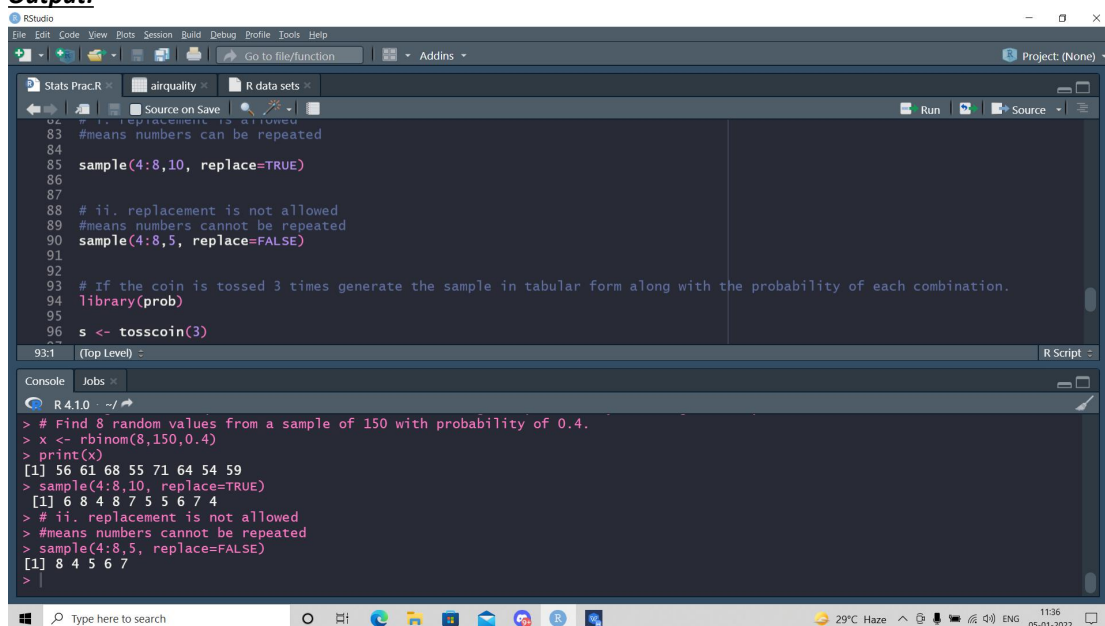
ii. replacement is not allowed
#means numbers cannot be repeated



```

72
73 #rbinom generates required number of random values of given probability from a given sample
74 # Find 8 random values from a sample of 150 with probability of 0.4.
75 x <- rbinom(8,150,0.4)
76
77 print(x)
78
79
80 # Generate the random sample between 4 to 8 where
81
82 # i. replacement is allowed
83 #means numbers can be repeated
84
85 sample(4:8,10, replace=TRUE)
86
87
88 # ii. replacement is not allowed
89 #means numbers cannot be repeated
90 sample(4:8,5, replace=FALSE)
91
92
93 # If the coin is tossed 3 times generate the sample in tabular form along with the probability of each combination.
94 library(prob)
95
96 s <- tosscoin(3)
97
98 print(s)
99
  
```

Output:



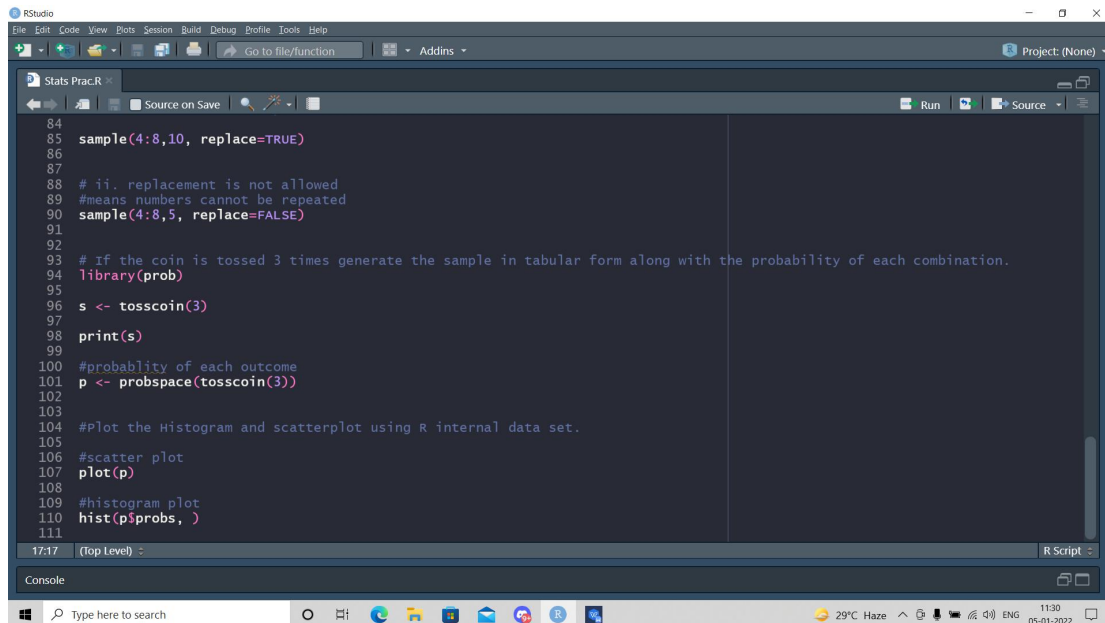
```

82 # i. replacement is allowed
83 #means numbers can be repeated
84
85 sample(4:8,10, replace=TRUE)
86
87
88 # ii. replacement is not allowed
89 #means numbers cannot be repeated
90 sample(4:8,5, replace=FALSE)
91
92
93 # If the coin is tossed 3 times generate the sample in tabular form along with the probability of each combination.
94 library(prob)
95
96 s <- tosscoin(3)
97
98
  
```

```

> # Find 8 random values from a sample of 150 with probability of 0.4.
> x <- rbinom(8,150,0.4)
> print(x)
[1] 56 61 68 55 71 64 54 59
> sample(4:8,10, replace=TRUE)
[1] 6 8 4 8 7 5 5 6 7 4
> # ii. replacement is not allowed
> #means numbers cannot be repeated
> sample(4:8,5, replace=FALSE)
[1] 8 4 5 6 7
>
  
```


5. If the coin is tossed 3 times generate the sample in tabular form along with the probability of each combination. Plot the Histogram and scatterplot using R internal data set.

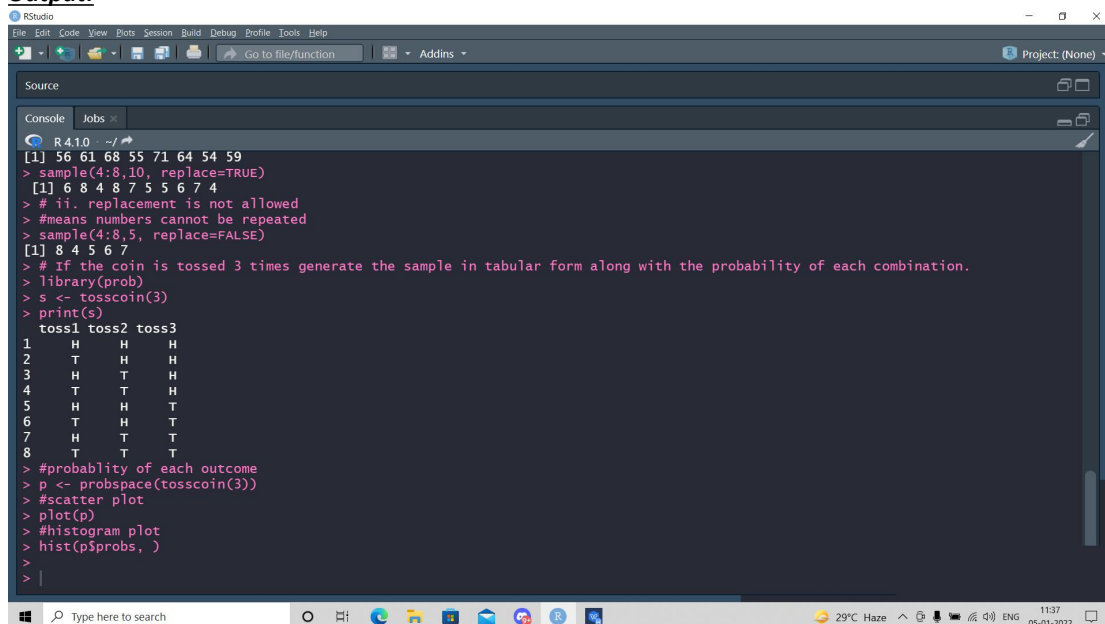


```

84
85 sample(4:8,10, replace=TRUE)
86
87
88 # ii. replacement is not allowed
89 #means numbers cannot be repeated
90 sample(4:8,5, replace=FALSE)
91
92
93 # If the coin is tossed 3 times generate the sample in tabular form along with the probability of each combination.
94 library(prob)
95
96 s <- tosscoin(3)
97
98 print(s)
99
100 #probability of each outcome
101 p <- probspace(tosscoin(3))
102
103
104 #Plot the Histogram and scatterplot using R internal data set.
105
106 #scatter plot
107 plot(p)
108
109 #histogram plot
110 hist(p$probs, )
111

```

Output:



```

R 4.1.0 ~./
[1] 56 61 68 55 71 64 54 59
> sample(4:8,10, replace=TRUE)
[1] 6 8 4 8 7 5 5 6 7 4
> # ii. replacement is not allowed
> #means numbers cannot be repeated
> sample(4:8,5, replace=FALSE)
[1] 8 4 5 6 7
> # If the coin is tossed 3 times generate the sample in tabular form along with the probability of each combination.
> library(prob)
> s <- tosscoin(3)
> print(s)
  toss1 toss2 toss3
1      H      H      H
2      T      H      H
3      H      T      H
4      T      T      H
5      H      H      T
6      T      H      T
7      H      T      T
8      T      T      T
> #probability of each outcome
> p <- probspace(tosscoin(3))
> #scatter plot
> plot(p)
> #histogram plot
> hist(p$probs, )
>

```

