R code for Data Science for Beginners

Day 3: Individual Exercise

Aric Jensen

2025-09-19

1. Vectors

Create an object called `vec.a` which is a vector consisting of the numbers, 1, 3, 5, 7. You need to use the c function.

```
vec.a=c(1,3,5,7)
```

Create a vector called `vec.b` consisting of the numbers, 2, 4, 6, 8.

```
vec.b=c(2,4,6,8)
```

Subtract vec.b from vec.a

```
vec.a-vec.b
```

```
[1] -1 -1 -1 -1
```

Create a new vector called vec.c by multiplying vec.a by vector vec.b

```
vec.c=vec.a*vec.b
```

Create a new vector called vec.d by taking the square root of each member of vec.c

```
vec.d=sqrt(vec.c)
```

What is the third element of the `vec.d` vector? Find out using square bracket. Note that since this is a vector, you only need to provide a single number inside the brackets.

```
vec.d[3]
```

```
[1] 5.477226
```

Create a new vector called vec.e consisting of all the integers from 1 through 100. You should use the seq function, rather than writing down all the 100 integers individually.

```
vec.e=seq(1:100)
```

The mean function calculates the arithmetic mean of the numbers stored in an object. Using the mean function, calculate the mean of the `vec.e` vector.

```
mean(vec.e)
```

```
[1] 50.5
```

As we saw in the joint exercise, the sum function calculates the sum of all the elements in an object. Calculate the sum of the `vec.e` vector.

```
sum(vec.e)
```

```
[1] 5050
```

The length function returns the number of elements stored in an object. Using the length function, find the number of elements stored in the `vec.e` vector.

```
length(vec.e)
```

```
[1] 100
```

The mean of an object can be obtained by sum(X)/length(X) because the definition of the mean is the sum of elements divided by the number of elements. Now, using the sum and length functions, calculate the mean of the `vec.e` vector. Compare the answer with that obtained with the mean function

We have learned that the by argument specifies an increment. For example,

```
seq(from = 0, to = 10, by = 2)
```

```
[1]  0  2  4  6  8 10
```

This creates a sequence that starts from 0 and ends with 10, and with an increment of 2.

Now, create a new object called olympic which is a sequence that starts from 1896 and ends with 2012, with an increment of 4.

```
olympic=seq(from = 1896, to = 2012, by = 4)
```

How many elements does the olympic vector contain? That is, what is the length of this vector? Find out by applying a function (not by manually counting the number of elements).

```
length(olympic)
```

```
[1] 30
```

So there are 30 elements in the olympic vector. Display all the elements contained in the olympic vector. These are the years where olympic games were (supposed to be) held. Display the contents of the olympic vector.

```
olympic
```

```
 [1] 1896 1900 1904 1908 1912 1916 1920 1924 1928 1932 1936 1940 1944 1948 1952
[16] 1956 1960 1964 1968 1972 1976 1980 1984 1988 1992 1996 2000 2004 2008 2012
```

Find out how many olympic games will have been held by the year 2400. Use the length and seq functions.

```
olympic2400=length(seq(from = 1896, to = 2400, by = 4))
olympic2400
```

```
[1] 127
```

print(olympic2400)

2. Matrices

Create a new vector called `v1` consisting of the following numbers: 1, 3, 5, 7, 9, 11

```
v1=c(1, 3, 5, 7, 9, 11)
```

Find out the length of this vector (Don't count the numbers by hand; use an appropriate function).

```
length(v1)
```

```
[1] 6
```

We will convert this vector into a matrix. That is, we will rearrange this vector so that it will have two dimensions (rows and columns). Since this vector has 6 numbers, if we want the matrix to have two rows, how many columns will there be?

```r
length(v1)/2
```

```
[1] 3
```

Create a matrix called mat.v using the following command:

```r
# matrix(data = v1, nrow = 2)
```

```r
mat.v=matrix(data = v1, nrow = 2)
mat.v
```

```
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    3    7   11
```

Take a look at the content of this matrix. How many columns are there?

Notice how the numbers in vec.v are used to fill up the cells of `mat.v`.We can see that R did it "by column". That is, R first filled up the first column of mat.v with the first two elements of vec.v, then moved on to the second and third columns.

You can use the `byrow` argument to change this. This argument takes one of two values, TRUE or FALSE (or T or F). That is, we write `matrix(data = v1, nrow = 2, byrow = TRUE)` Now, create an object called mat.w using the command above.

```r
mat.w = matrix(data = v1, nrow = 2, byrow = TRUE)
mat.w
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    7    9   11
```

Compare mat.v and mat.w. Do you see that R filled up the cells "by row" to create the mat.w matrix ?

Many functions in R have arguments that take TRUE or FALSE like the byrow argument we just used. In most cases, functions have a default value. In the case of the matrix function, the default value for the byrow argument is FALSE, meaning that, if you don't specify anything, R will automatically sets byrow = FALSE.

Find the number in the second row, second column of `mat.w`

```
mat.w[2, 2]
```

```
[1] 9
```

Find the number in the second row, second column of `mat.v`

```
mat.v[2, 2]
```

```
[1] 7
```

3. Lists

Create a list of months (as the names of the elements) with how many days each month has as the elements in the list

```
month_names = c("January", "February", "March", "April", "May", "June", "July", "August", "Se
days = list(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
names(days) = month_names
```

Display the number of days August has from the list

```
days$August
```

```
[1] 31
```

Convert the list to a vector

```
vdays = unlist(days)
vdays
```

```
  January  February     March     April       May      June      July    August
       31        28        31        30        31        30        31        31
September   October  November  December
       30        31        30        31
```

4. Apply functions

Load R default data set mtcars

```
dt<-mtcars
dt
```

```
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C         17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE        16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL        17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC       15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128          32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic       30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla    33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona     21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
```

```
Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

Use one of the apply functions to calculate the min value for each column/variable

```
min_val = lapply(dt, min)
min_val
```

```
$mpg
[1] 10.4

$cyl
[1] 4

$disp
[1] 71.1

$hp
[1] 52

$drat
[1] 2.76

$wt
[1] 1.513

$qsec
[1] 14.5

$vs
[1] 0
```

```
$am
[1] 0

$gear
[1] 3

$carb
[1] 1
```

Use one of the apply functions to indicate zero values in each column/variable

```
sapply(dt, function(x) which(x==0))
```

```
$mpg
integer(0)

$cyl
integer(0)

$disp
integer(0)

$hp
integer(0)

$drat
integer(0)

$wt
integer(0)

$qsec
integer(0)

$vs
 [1]  1  2  5  7 12 13 14 15 16 17 22 23 24 25 27 29 30 31

$am
 [1]  4  5  6  7  8  9 10 11 12 13 14 15 16 17 21 22 23 24 25
```

```
$gear
integer(0)

$carb
integer(0)
```

Finally, execute the entire contents of this file, making sure there is no error messages.