```
---
title: "R code for Data Science for Beginners"
subtitle: " Day 3: Individual Exercise"
author: "Benjamin Cook"  #If multiple, 'c("A", "B")'
date: "2024-9-11"  #r Sys.Date()
output:
  pdf_document: default
  html_document: default
---


## 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.

WRITE YOUR ANSWER (code) HERE  > vec.a <- c(1, 3, 5, 7)
>
> print(vec.a)

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

WRITE YOUR ANSWER (code) HERE  > vec.b <- c(2, 4, 6, 8)
>
> print(vec.b)

### Subtract vec.b from vec.a

WRITE YOUR ANSWER (code) HERE  > vec.a <- c(1, 3, 5, 7)
> vec.b <- c(2, 4, 6, 8)
>
> result <- vec.a - vec.b
>
> print(result)

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

WRITE YOUR ANSWER (code) HERE  > vec.a <- c(1, 3, 5, 7)
> vec.b <- c(2, 4, 6, 8)
>
> vec.c <- vec.a * vec.b
>
> print(vec.c)

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

WRITE YOUR ANSWER (code) HERE  > vec.c <- c(2, 12, 30, 56)
>
> vec.d <- sqrt(vec.c)
>
> print(vec.d)

### 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.

WRITE YOUR ANSWER (code) HERE  > third_element <- vec.d[3]
>
> print(third_element)


### 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.
```

```
WRITE YOUR ANSWER (code) HERE  > vec.e <- seq(1, 100)
>
> print(vec.e)
```

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.

```
WRITE YOUR ANSWER (code) HERE > mean_vec_e <- mean(vec.e)
>
> print(mean_vec_e)
```

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.

```
WRITE YOUR ANSWER (code) HERE  > sum_vec_e <- sum(vec.e)
>
> print(sum_vec_e)
```

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.

```
WRITE YOUR ANSWER (code) HERE  > length_vec_e <- length(vec.e)
>
> print(length_vec_e)
```

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

```
WRITE YOUR ANSWER (code) HERE  > sum_vec_e <- sum(vec.e)
> length_vec_e <- length(vec.e)
> manual_mean_vec_e <- sum_vec_e / length_vec_e
> print(manual_mean_vec_e)
[1] 50.5
> mean_vec_e <- mean(vec.e)
> print(mean_vec_e)
[1] 50.5
```

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

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

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.

```
WRITE YOUR ANSWER (code) HERE  > olympic <- seq(from = 1896, to = 2012, by = 4)
>
> print(olympic)
```

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).

```
WRITE YOUR ANSWER (code) HERE  > length_olympic <- length(olympic)
>
> print(length_olympic)
```

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.

WRITE YOUR ANSWER (code) HERE > print(olympic)

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

WRITE YOUR ANSWER (code) HERE > olympic_2400 <- seq(from = 1896, to = 2400, by = 4)
>
> num_olympic_2400 <- length(olympic_2400)
>
> print(num_olympic_2400)

2. Matrices

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

WRITE YOUR ANSWER (code) HERE > 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).

WRITE YOUR ANSWER (code) HERE > length_v1 <- length(v1)
>
> print(length_v1)

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?

WRITE YOUR ANSWER AS A COMMENT. 3

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

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

WRITE YOUR ANSWER (code) HERE > mat.v <- matrix(data = v1, nrow = 2)
>
> print(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.

WRITE YOUR ANSWER (code) HERE > mat.w <- matrix(data = v1, nrow = 2, byrow = TRUE)
>
> print(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`

WRITE YOUR ANSWER (code) HERE

```
> number <- mat.w[2, 2]
>
> print(number)
```

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

WRITE YOUR ANSWER (code) HERE

```
> number <- mat.v[2, 2]
>
> print(number)
```

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

WRITE YOUR ANSWER (code) HERE

```
> months_days <- list(
+     January = 31,
+     February = 28,
+     March = 31,
+     April = 30,
+     May = 31,
+     June = 30,
+     July = 31,
+     August = 31,
+     September = 30,
+     October = 31,
+     November = 30,
+     December = 31
+ )
>
> print(months_days)
```

Display the number of days August has from the list

WRITE YOUR ANSWER (code) HERE

```
> august_days <- months_days$August
>
> print(august_days)
```

Convert the list to a vector

WRITE YOUR ANSWER (code) HERE

```
> months_days_vector <- unlist(months_days)
>
> print(months_days_vector)
```

4. Apply functions

```
### Load R default data set mtcars

WRITE YOUR ANSWER (code) HERE  > data(mtcars)
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1

### Use one of the apply functions to calculate the min value for each column/variable
WRITE YOUR ANSWER (code) HERE  > min_values <- apply(mtcars, 2, min)
>
> print(min_values)

### Use one of the apply functions to indicate zero values in each column/variable
WRITE YOUR ANSWER (code) HERE  > zero_values <- apply(mtcars, 2, function(x) sum(x == 0))
>
> print(zero_values)


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