D1–Introduction to Computers, Programming and R

Revised on March 18, 2016

---

1 ♡

1. Use the `seq()` function to create the following sequences.
   (a) `1 2 3 4 5 6 7 8 9 10`
   (b) `10 9 8 7 6 5 4 3 2 1`
   (c) `1 3 5 7 9 11 13 15`
2. Generate a sequence of length 12 that starts from 0 and ends up with 1.
3. Generate a sequence that starts from 0 and maximumly ends up with 1 with the increment of 0.03.
4. Generate a sequence of length of 7 that starts from 1 with increment of 1.3.
5. Generate a sequence of length of 7 that starts from 1 with increment of -1.3.
6. Generate a sequence of length 8 that ends up with 20 and with the increment of 2.2.
7. Generate a sequence of length 8 that ends up with 20 and with the increment of -2.2.
8. Compute the mean, variance, standard deviation of the sequences you have generated from **1.6**.
9. Use `rep()` function to generate the following sequences.
   (a) `"hello" "hello" "hello"`
   (b) `TRUE TRUE TRUE TRUE TRUE`
   (c) `2 2 2 2 2 2 2 2 2 2`
10. Use `seq()` or `rep()`, or both, to generate the following sequences and assign them to variables `s1.10a` and `s1.10b` respectively.
    (a) `1 2 3 4 5 1 2 3 4 5 1 2 3 4 5`
    (b) `1 4 9 16 25 36 49 64 81`
11. You are given a vector `w <- seq(5,10,1.2)`. Use **rep()** to generate a sequence that each element in `w` has been repeated three times, i.e.

    ```
    5.0 5.0 5.0 6.2 6.2 6.2 7.4 7.4 7.4 8.6 8.6 8.6 9.8 9.8 9.8
    ```

2

1. ♡ Check if the elements in the sequence in question **1.10.(a)** satisfy following conditions.
   (a) The elements are greater than 2.
   (b) The elements are smaller than 3.
   (c) The elements are greater or equal than 2.
   (d) The elements are smaller or equal than 3.
   (e) The elements are equal to 5.

(f) The elements are greater than 2 but smaller than 4.

(g) The elements are greater than 3 or smaller than 2.

2. Check the help of `which()` function and find out which elements in **1.10.(a)** satisfy the conditions in **2.1**.

3. Select the elements form **1.10.(a)** that satisfies the condition in **2.1**.

4. ♡ Consider the sequence in **1.10.(b)** and preform the following tasks.

   (a) Delete the first entry.

   (b) Delete the last entry.

   (c) Delete the last four elements.

   (d) Select the entries that in the even positions.

   (e) Select the entries that contains odd numbers[Hint: try `?%%`].

   (f) Select the entries that is greater than the mean value of the sequence.

3

1. Consider the following sequence

```
> a <-  c(seq(1, 10, 2.2), rep(NA, 3), seq(10, 20, 1.5), rep(NaN, 4))
> a
 [1]  1.0  3.2  5.4  7.6  9.8   NA   NA   NA 10.0 11.5 13.0 14.5 16.0 17.5 19.0
[16]  NaN  NaN  NaN  NaN
```

   and calculate the mean and variance of the sequence by using `mean(a)` and `var(a)`. What results do you find? Also check the type of the sequence.

2. Repeat the previous tasks in **3.1** but plug in an extra argument `na.rm = TRUE` in the `mean()` and `var()` functions. What results will you have now?

3. Use `is.na()` and `is.nan()` to check if the entries in the sequence are **not available** or **not a number**.

4. According to **3.3**, if you only want to remove entries with `NaN`, what command should you issue? If you only want to remove entries with `NA`, what command should you issue?

5. Remove `NA` and `NaN` in the sequence and calculate the mean and variance for the remaining elements. Compare your results with the results in **3.1** and **3.2**.

4 ♡

1. Assume you have a vector with the weekdays on which you need to study at the school,

```
> study <- c("Monday", "Tuesday", "Thursday")
> study
[1] "Monday"   "Tuesday"  "Thursday"
```

   but you also decided go to school on Wednesday. Insert `"Wednesday"` to your `study` vector at proper location.

2. You have a part-time job at the student union on Friday, so you have `job <- "Friday"`. Also, let `fun <- c("Saturday","Sunday")`. Merge all "`study`", "`job`" and "`fun`" into a new vector called "`weekdays`"

3. You create a vector to indicate how much you are going to work or study during that week,

```
> hours <- c(rep(4, 4), 6, 0, 0)
> hours
[1] 4 4 4 4 6 0 0
```

4. Your tasks for each day is then

```
> tasks <- c(rep("study", 4), "job", rep("fun", 2))
> tasks
[1] "study" "study" "study" "study" "job"   "fun"   "fun"
```

   Now convert your vector `tasks` to factor to indicate the type of tasks and name it as `tasks`.

5. Check the class type of the four vectors you have now using `class()`.

6. Create a list with the vectors `weekdays`, `hours` and `tasks` and name it as `weekPlan` and print `weekPlan` on the R prompt.

7. Use the `names()` function to check if you have attributed the names to the elements of your list. If not, name each element with the vector's name so that your `weekPlan` will be something like this

```
> weekPlan
$weekdays
[1] "Monday"    "Tuesday"   "Wednesday" "Thursday"  "Friday"    "Saturday"
[7] "Sunday"

$hours
[1] 4 4 4 4 6 0 0

$tasks
[1] study study study study job   fun   fun
Levels: fun job study
```

8. Perform some basic tasks to check the type and length of `weekPlan`. Check if there are missing values in the list. Check how much memory this object occupies [Hint `?object.size()`]. Try `str(weekPlan)` to display the internal structure of this object.

9. Append the new variable `note <- "plan for week 10"` to the current list `weekPlan`. How many ways can you find to do this append?

10. Extract `weekdays` from the list `weekPlan` using "`[[`" and "`$`".

11. Select the first entry in the list `weekPlan`.

12. Create a new list named `weekPlan2` with entries `weekdays` and `tasks` from `weekPlan`.

13. Create a new list named `weekPlan3` with the first and second entries from `weekPlan`.

14. Delete `note` from the list `weekPlan`.

15. Find out on which days you have to work for more than 4 hours.

5

1. Create a data frame with the vectors `weekdays`, `hours`, `tasks` that you obtained in **4.1**–**4.4** and name it as `weekPlanNew`

```
> weekPlanNew
   weekdays hours tasks
1    Monday     4 study
2   Tuesday     4 study
3 Wednesday     4 study
4  Thursday     4 study
```

```
5    Friday     6   job
6  Saturday     0   fun
7    Sunday     0   fun
```

2. Perform the following basis tasks for the data frame `weekPlanNew`
   (a) Check the structure of the data frame.
   (b) Check the number of rows and columns in the data frame.
   (c) What is the dimension of the data frame?
   (d) Check the row and column names of the data frame.
3. Create a new vector called `costs` for your daily cost in SEK. `costs <- c(70, 75, 58, 62, 140, 90, 70)` and append it to the end of the data frame.
4. ♡ Answer the following questions for your new `weekPlanNew`
   (a) What are you going to do on the first day of the week? How long will you work on that day?
   (b) What are you going to do for the last three days?
   (c) How much money do you spend during the weekend in total, how about the whole week?
   (d) How many hours do you work (not including study times) in total for this week?
   (e) Which days do you work or study less than five hours?
   (f) Which day do you spend more than 100SEK?
   (g) You decide to go for work on Tuesday instead of studying. Change the corresponding task to job.
   (h) Extract the first column from the data frame, and check the dimension of that column. Now extract the first and third columns of the data frame and then check the dimension of that newly obtained data frame.
   (i) Repeat **5.4.(h)** but insert `drop = FALSE` argument when you subtract the columns.
5. You think the next week you will do the similar plan as this week, so you copy this week's plan and append it after the last row of current `weekPlanNew`.
6. You feel you went over budget this week, so you decide not to spend more than the smallest daily spending during this week. Modify the corresponding entries.
7. Discussion: when should you use data frame or list?

6 ♡

1. Write a function `mySummary` where the input argument is `x` can be any vector and the output should contain the basic summary (mean, variance, length, max and minimum values, type) of the vector you have supplied to the function.
2. Test your function with some vectors (that you make up by yourself).
3. What will happen if your input is not a vector (e.g. a data frame `weekPlanNew`) in our previous example?

7 ♡

1. The roots for the quadratic equation $ax^2 + bx + c = 0$ are of the form

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

2. Write a function named `quaroot` to solve the roots of given quadratic equation with `a ,b , c,` as input arguments. [Hint: you may need the `sqrt()` function]

3. Test your function on the following equations

$$\begin{aligned} x^2 + 4x - 1 &= 0 \\ -2x^2 + 2x &= 0 \\ 3x^2 - 9x + 1 &= 0 \\ x^2 - 4 &= 0 \end{aligned}$$

4. Test your function with the equation $5x^2 + 2x + 1 = 0$. What are the results? Why? [Hint: check $b^2 - 4ac$]?

5. Modify your function and return `NA` if $b^2 - 4ac < 0$.