

csci297b Exercise 2

Basic R Operations

1. Log on to the RStudio Workbench Server. In the same project as the previous exercise, open a new R script called `exercise_2`. Make sure you include any metadata you feel is appropriate (title, description of task, date of script creation etc). Don't forget to comment out your metadata with a `#` at the beginning of the line.
2. Let's use R as a fancy calculator. Find the natural log, log to the base 10, log to the base 2, square root and the natural antilog of 12.43. See Section 2.1 of the Introduction to R book for more information on mathematical functions in R. Don't forget to write your code in RStudio's script editor and source the code into the console.
3. Next, use R to determine the area of a circle with a diameter of 20 cm and assign the result to an object called `area_circle`. Google is your friend if you can't remember the formula to calculate the area of a circle! Also, remember that R already knows about pi.
4. Now for something a little more tricky. Calculate the cube root of 14×0.51 . You might need to think creatively for a solution (hint: think exponents), and remember that R follows the usual order of mathematical operators so you might need to use brackets in your code (see https://en.wikipedia.org/wiki/Order_of_operations if you've never heard of this). The point of this question is not to torture you with math (so please don't stress!), its to get you used to writing mathematical equations in R and highlight the order of operations.
5. Now let's do some serious computation. The quadratic formula finds the roots of the equation

$$ax^2 + bx + c = 0$$

and is given by

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Use this knowledge to solve the following quadratic equations:

$$\begin{aligned}(x - 2)(x - 3) &= x^2 - 5x + 6 \\ (2x - 8)(x + 4) &= 2x^2 - 32 \\ (3x + 9)(x - 5) &= 3x^2 - 6x - 45\end{aligned}$$

Note: the factorings on the left should give you a hint about what the right answers are.

Solve each problem by assigning to three variables, `a`, `b`, and `c`, the values from the quadratic equation in your R script.

Then find each of the solutions by entering the two forms of the quadratic formula on separate lines, sourcing them to the console, and checking that the output is what you expect.

You can solve the other problems by copying and pasting your solution to the first problem, and then changing just the values of `a`, `b`, and `c`.

6. Ok, you're now ready to explore one of R's basic (but very useful) data structures - vectors. A vector is a sequence of elements (or components) that are all of the same data type (see Section 3.2.1 for an introduction to vectors). Although technically not correct it might be useful to think of a vector as something like a single column in a spreadsheet. There are a multitude of ways to create vectors in R but you will use the concatenate function `c()` to create a vector called `weight` containing the weight (in kg) of 10 children: 69, 62, 57, 59, 59, 64, 56, 66, 67, 66.
7. Now you can do some useful stuff to your weight vector. Get R to calculate the mean, variance, standard deviation, range of weights and the number of children of your `weight` vector (see Section 2.3 for more details). Now read Section 2.4 of the R book to learn how to work with vectors. After reading this section you should be able to extract the weights for the first five children using Positional indexes and store these weights in a new variable called `first_five`. Remember, you will need to use the square brackets `[]` to extract (aka index, subset) elements from a vector variable.
8. We're now going to use the `c()` function again to create another vector called `height` containing the height (in cm) of the same 10 children: 112, 102, 83, 84, 99, 90, 77, 112, 133, 112.

Use the `summary()` function to summarise these data in the height object. Extract the height of the 2nd, 3rd, 9th and 10th child and assign these heights to a variable called `some_child` (take a look at the section Positional indexes in the R book if you're stuck).

9. We can also extract elements using Logical indexes. Let's extract all the heights of children less than or equal to 99 cm and assign to a variable called `shorter_child`.
10. Now you can use the information in your `weight` and `height` variables to calculate the body mass index (BMI) for each child. The BMI is calculated as weight (in kg) divided by the square of the height (in meters). Store the results of this calculation in a variable called `bmi`.

Note: you don't need to do this calculation for each child individually, you can use both vectors in the BMI equation - this is called vectorisation (see Section 2.4.4 of the Introduction to R book).

11. Now let's practice a very useful skill - creating sequences (honestly it is...). Take a look at Section 2.3 in the R book (the bit on creating sequences) to see the myriad ways you can create sequences in R. Let's use the `seq()` function to create a sequence of numbers ranging from 0 to 1 in steps of 0.1 (this is also a vector by the way) and assign this sequence to a variable called `seq1`.
12. Next, see if you can figure out how to create a sequence from 10 to 1 in steps of 0.5. Assign this sequence to a variable called `seq2`
13. Let's go sequence crazy! Generate the following sequences. You will need to experiment with the arguments to the `rep()` function to generate these sequences (see Section 2.3 for some clues):

```
1 2 3 1 2 3 1 2 3
1 1 1 2 2 2 3 3 3 1 1 1 2 2 2 3 3 3
1 1 1 1 1 2 2 2 2 3 3 3 4 4 5
7 7 7 7 2 2 2 8 1 1 1 1 1
"a" "a" "a" "c" "c" "c" "e" "e" "e" "g" "g" "g"
"a" "c" "e" "g" "a" "c" "e" "g" "a" "c" "e" "g" "a" "c" "e" "g"
```

14. Ok, back to the variable `height` you created above. Sort the values of height into ascending order (shortest to tallest) and assign the sorted vector to a new variable called `height_sorted`. See Section 2.4.3 for an introduction to sorting and ordering vectors. Now sort all heights into descending order and assign the new vector a name of your choice.
15. Let's give the children some names. Create a new vector called `child_names` with the following names of the 10 children: "Alfred", "Barbara", "James", "Jane", "John", "Judy", "Louise", "Mary", "Ronald", "William".
16. A really useful (and common) task is to order the values of one variable by the order of another variable. To do this you will need to use the `order()` function in combination with the square bracket notation `[]`. Have a peep at Section 2.4.3 for some details. Create a new variable called `names_sort` to store the names of the children ordered by child height (from shortest to tallest). Who is the shortest? who is the tallest child?
17. Now order the names of the children by descending values of weight and assign the result to a variable called `weight_rev` (Hint: perhaps include the `rev()` function?). Who is the heaviest? Who is the lightest?
18. Almost there! In R, missing values are usually represented with an `NA`. Missing data can be tricky to deal with in R (and in statistics more generally) and cause some surprising behaviour when using some functions (see Section 2.4.5 of the Introduction to R book).

To explore this a little further let's create a vector called `mydata` with the values 2, 4, 1, 6, 8, 5, `NA`, 4, 7.

Notice the value of the 7th element of `mydata` is missing and represented with an `NA`. Now use the `mean()` function to calculate the mean of the values in `mydata`. What does R return? Confused? Next, take a look at the help page for the function `mean()`. Can you figure out how to alter your use of the `mean()` function to calculate the mean without this missing value?

19. Finally, list all variables in your workspace that you have created in this exercise. Remove the variable `seq1` from the workspace using the `rm()` function.
20. Don't to forget to save your R script. Since we put this script in the same project as the last one, it should already be shared with your instructor.

Close your Project by selecting File→Close Project on the main menu.