Minilab 1c Worksheet

Vectors and Lists

In this minilab, we will introduce *vectors* and *lists* in R as these are fundamental to how data is stored in R, so very helpful to understand.

1. What is a Vector in R?

Vectors are one-dimensional collections of values that are all stored in a single variable. For example, you can make a vector people that contains strings.

Warning! — All the elements in a vector need to have the same *type*, e.g., numeric, character, logical. You can't have a vector whose elements include both numbers and character strings. These were the semi-finalists and the 2020 Australian Open (tennis).

```
people = c("Djokovic","Federer","Thiem","Zverev",
           "Barty","Kenin","Halep","Muguruza")
print(people)
```

Other functions can also help with creating vectors.

```
odds = seq(1,20,2)
print(odds)
```

As a shorthand, you can produce a sequence with the colon operator (a:b), which returns a vector from a to b with the element values being incremented by 1.

```
one_to_seventy = 1:70
print(one_to_seventy)
```

The output is as follows:

```
> one_to_seventy = 1:70
> print(one_to_seventy)
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
[18] 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
[35] 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
[52] 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68
[69] 69 70
```

When you print out `one_to_seventy`, in addition to the leading [1], there are numbers in square brackets at the start of each output line. These tell you the starting position (index) of elements printed on that line. Which index appears at the beginning of the line depends on the width of the R console pane in RStudio.

Vectorised operations. When performing operations (such as addition) on vectors, the operation is applied to vectors element-by-element. Vectors support any operators that apply to their type.

```
a = c(3,1,4,1,5)
b = c(1,6,1,8,0)
a+b
```

Vector indices. Vectors are the fundamental structure for storing data. Yet, you often want to work with just *some* of the data in a vector. The simplest way that you can refer to individual elements in a vector is by their *index*, which is the number of their position in the vector.

```
vowels = c("a","e","i","o","u")
print(vowels[1])
print(length(vowels))
print(vowels[length(vowels)])
print(vowels[2:4])
print(vowels[c(1,3,5)])
```

Warning! — In R, vector elements are indexed starting with 1. This is different from most other programming languages, which are *zero-based* and so reference the first element in a set at index 0.

Exercise.

(a) What happens if you specify an index that is *out-of-bounds*, e.g., vowels[10]?

(b) What happens if you specify a negative index, e.g., vowels[-3]?

Vector filtering. The previous examples used a vector of indices (*numeric* values) to retrieve a subset of elements from a vector. Alternatively, you can put a vector of *logical* (Boolean) values (i.e. TRUE or FALSE) inside the square brackets to specify which elements you want to return. TRUE in the corresponding position

```
shoe_sizes = c(5.5, 11, 7, 8, 4)
filter = c(TRUE, FALSE, FALSE, FALSE, TRUE)
print(shoe_sizes[filter])
shoe_is_small = (shoe_sizes<6)
print(shoe_is_small)
print(shoe_sizes[shoe_is_small])
print(shoe_sizes[shoe_sizes>6])
```

2. What is a List in R?

A list is a lot like a vector, in that it is a one-dimensional collection of data. However, unlike a vector, you can store elements of *different types* in a list, e.g., a list can contain numeric data *and* string data. Lists can also contain more complex data types, including vectors and even other lists.

Elements in a list can also be tagged with names that you can use to easily refer to them. This allows you to use lists to create a type of map (like a dictionary in Python). As a result, lists are extremely useful for organising data. They allow you to group together data like a person's name (string), job title (string), salary (number), and whether the person has a carparking permit (logical).

Warning! — The definition of a list in R is different from how some other languages use the term "list".

```
person = list(
  first_name = "Ada",
  job = "Programmer",
  salary = 100000,
  carparking_permit = TRUE
)
print(person)
names(person)
```

```
person$first_name
person$job
```

You can also access the elements of a list by index using *double-bracket notation*, useful for when the elements of the list are not tagged with a label. Notice the difference between single-square-bracket indexing (returns a list containing one element) and double-square-bracket indexing (returns an element from the list).

```
person[["salary"]]
animals = list("Aardvark","Baboon","Camel")
print(animals)
animals[1]
animals[[1]]
is.list(animals)
is.list(animals[1])
is.list(animals[[1]])
```

Note. "1e+05" is scientific notation that means $1 \times 10^5 = 100000$. We will often see scientific notation in graphical plots, so it is helpful to know how to interpret these values.

Exercise.

(a) Create your own vector, e.g., names of some of your friends.

(b) Create your own list, e.g., some details of your life.

(c) Create a list which has another list as one of its elements.

(d) Create a list which has a vector as one of its elements.

(e) Is it possible (in R) to create a vector which has a list as one its elements?

Summary

In this minilab, we have briefly looked at vectors and lists, which are the most basic way data is stored in R. Of course, typical datasets include more data that is stored in a single vector or list, as we will see in the next minilab.