Tutorial 1: Introduction to Data Visualisation

Table of contents

Learning Objectives 1
Preparation 1
Exercise 1: R Packages 1
Exercise 2: Assign a Variable 2
Exercise 3: Basic Computation 3
Exercise 4: Vectors 4

Learning Objectives

- Get you started using R
- Learn your way around RStudio.
- Become familiar with R, RStudio, RStudio projects.
- Become confident doing basic computation in R.

Preparation

- Install R and RStudio
- Watch/Read the Introductory materials on R provided

Exercise 1: R Packages

R has many standard functions, like `mean()`, `sum()` and `sqrt()`. Other functions are stored in R packages that can be loaded as needed for specific tasks.

One of the common packages is the `tidyverse` it contains lots of useful functions for data wrangling and visualisation. In fact the `tidyverse` is actually a collection of packages.

The below code checks if the package is already downloaded. If the package is not there if then it uses the `install.package` function to download it.

```
if(!require(tidyverse)){
    install.packages("tidyverse")
}
```

You can't use function in the `tidyverse` yet though. When you start a new session in RStudio, you will need to load the package into R using the `library` function.

```
library(tidyverse)
```

A useful was to think of it is `install.packages()` is like screwing in a lightbulb, and running the `library()` function is light turning the light when you want to use it.

To check the package has installed correctly try calling `?ggplot`. This will bring up the help menu for this function. If you scroll down the bottom you'll see some examples you can run.

Your turn:

1. Change the above code to install the `cowsay` package
2. Load `cowsay` package using the `library()` function
3. Try running `cowsay::say()`
4. Now try running `cowsay::say(what = "I'm learning R", by = "alligator")`
5. Call the help menu to read more about the `?say`function. Then change the function inputs to your name, your favourite animal and your favourite colour (based on the options available).

Exercise 2: Assign a Variable

R is more than just as a calculator, you can store things in a **variable**. Here the word variable has a slightly different meaning to the usual statistical meaning.

Let us start with a simple command, type in `a = 5`. Here we assign the value 5 to the variable a.

```
a = 5
```

Note that you can use either `<-` or `=` to assign variables.

Your turn:

1. Let try `b = 4`
2. We can store more than just numbers in a variable. Try storing your name. e.g. `name = 'Jane Doe'`.
3. We can use `str(name)`, `str(a)` to check the structure of the variable, it will show you that variable `name` contains a character and the variable `a` contains numeric.
4. R is case sensitive. We created a variable `name` in previous step, but try typing `NAME`

Exercise 3: Basic Computation

Let continue with: `x = 5`, `y = sqrt(16)`, `z = -3`, and `w = x + y + z`

```
x = 5
y = sqrt(16)
z = -3
w = x + y + z
```

`sqrt(16)` is a function for square root of 16, therefore the value for `y` is 4. To find the value of `w`, type it in to your console. (The answer should be 6).

```
w
```

R can be used as a calculator, to subtract use `-`, to multiply use `*`, to divide use `/` and to take powers use `^`.

To see all assigned variables so far use:

```
ls()
```

You can also look in the environment tab in the top right hand panel of Rstudio to see what variables have been stored.

Your turn:

1. Try `h = x^2` and `f = z/y`.

Exercise 4: Vectors

A vector is a list of the same type of object and is a basic data structure in R that can hold values. Previously `w` had only one value, but we can store multiple values in `w` by using the command `c()`. Here c stands for "combine" or "concatenate".

For example try:

```
litres_drank <- c(50, 40, 25, 0)
litres_drank
str(litres_drank)
```

To create a vector of `characters`:

```
type_of_drink <- c('Coke', 'Pepsi', 'Coke', 'Homebrand')
type_of_drink
str(type_of_drink)
```

We can also access different elements of our vector using square brackets `[]`

```
litres_drank[2]
type_of_drink[2]
```

Sometimes when we apply a function to a vector it will work on each element in the vector

```
litres_drank*2
log(litres_drank)
```

Or the function might summarise something about our vector

```
table(type_of_drink)
max(litres_drank)
```

Your turn:

1. Create two variables: `x1` contains "1,2,3,4,5" and `y1` contains "-1,-2,-3,-4,-5"
2. Compute the mean for `x1` and `y1` by using `mean()`
3. We can find the variance of the vectors `x1` and `y1` using the `var` function.
4. We can find the correlation between the two vectors `x1` and `y1` using the `cor` function.
5. Now try changing the middle element of `x1` to -3 and the middle element of `y1` to 0, and redo the above exercises.

© Copyright 2025 Monash University