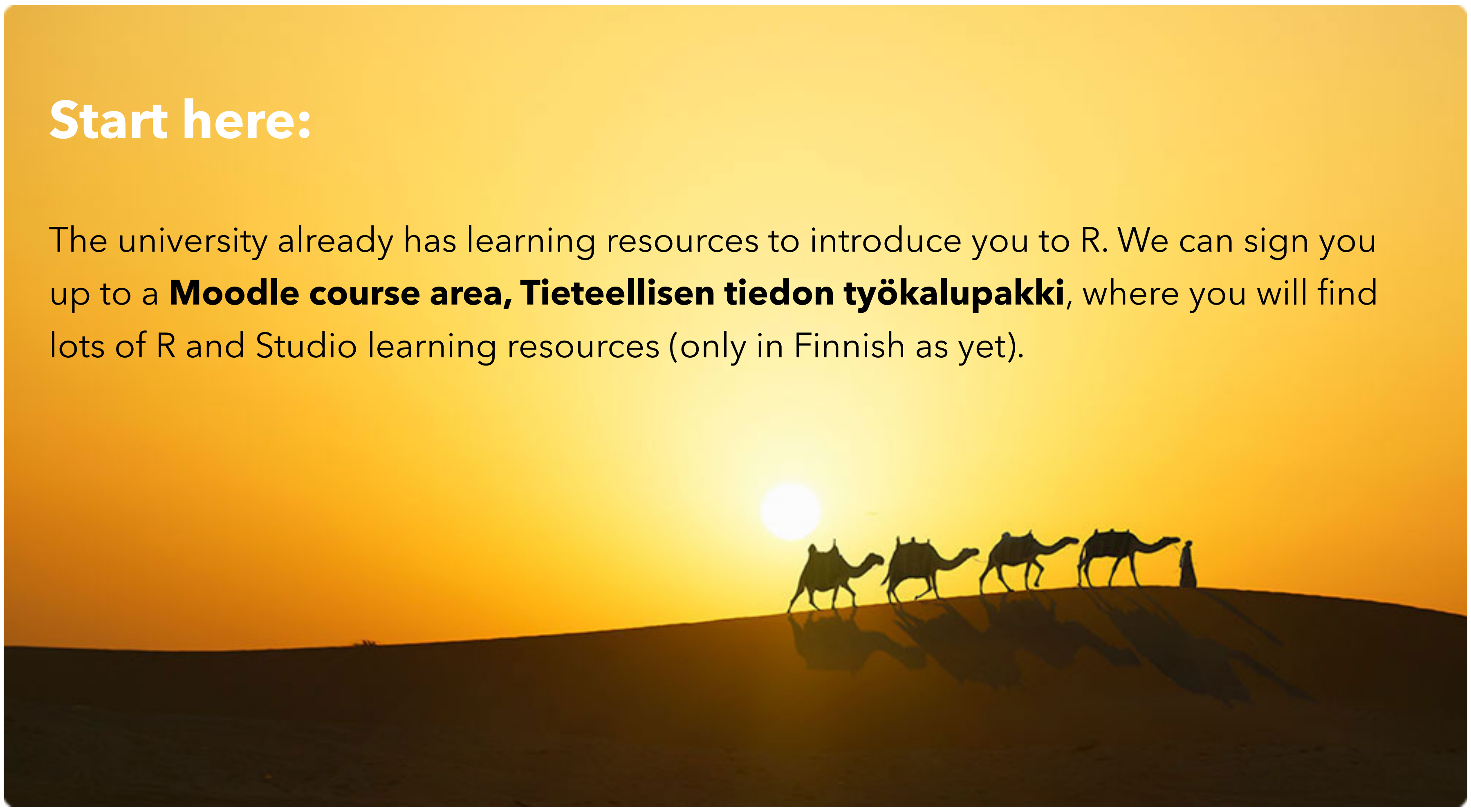
Biostatistics Unit - Consulting Service

Get started with R

...OR...

How to navigate a desert of code

The biostatistics unit gets a lot of queries along the lines of “I need to do my analysis in R but I can’t get my code to work”. This is a difficult request for us to respond to because there are so many ways in which it could arise - anything from the data file being wrongly constructed, to mistakes in the precise syntax of your code, to an incorrect decision around which statistical test to use. Deconstructing where things are going wrong is *part of doing analysis in R*, and therefore in the end something that the researcher needs to learn. This document is a first *support for this learning process*. Unfortunately, we can’t provide one-on-one “get started with R” sessions due to resource limitations.



Start here:

The university already has learning resources to introduce you to R. We can sign you up to a **Moodle course area, Tieteellisen tiedon työkalupakki**, where you will find lots of R and Studio learning resources (only in Finnish as yet).

Top tips for self-learning R

The biggest challenge is that often new R users are at the same time trying to **learn how to code AND learn how to do statistics**. Be aware that these are different skills and both take time to learn!

Tip #1: PRACTICE without pressure

Using a suitable "basics of R" walkthrough, WRITE OUT AND RUN THE CODE for each step. Think properly about what each line of code does, but don't think about your own analysis. Just follow the instructions and use the dummy data provided.

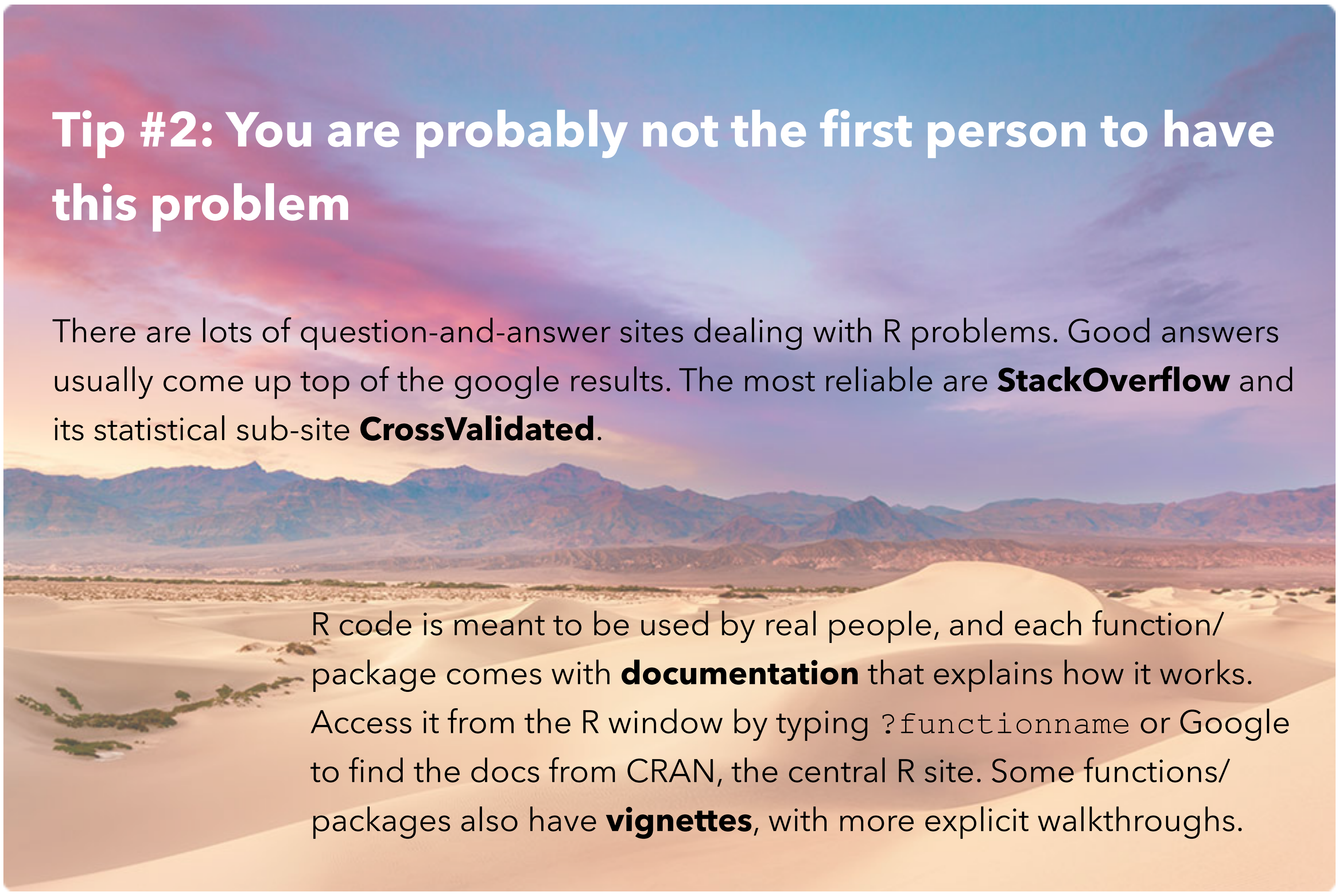
This will help you understand how the syntax works (how to identify objects, what a function is, how to use brackets etc.). Only once you know this can you usefully use R for your own analysis.

Some favourite sources for writing your first few lines of R:

- The Rstudio website has a good compilation of sources https://education.rstudio.com/learn/beginner/ . This one is particularly aimed at people who have never coded/programmed before: https://moderndive.netlify.app/1-getting-started.html
- An actual book, classic R reference for both coding and analysis (though a little old-fashioned in its teaching style): "The R Book" by Michael Crawley
- Online basics for analysis and graphing, clear with nice code to copy & paste: http://www.cookbook-r.com/

- Deeper walkthrough from zero to pro, good if you think you might use R a lot: https://r4ds.had.co.nz/introduction.html

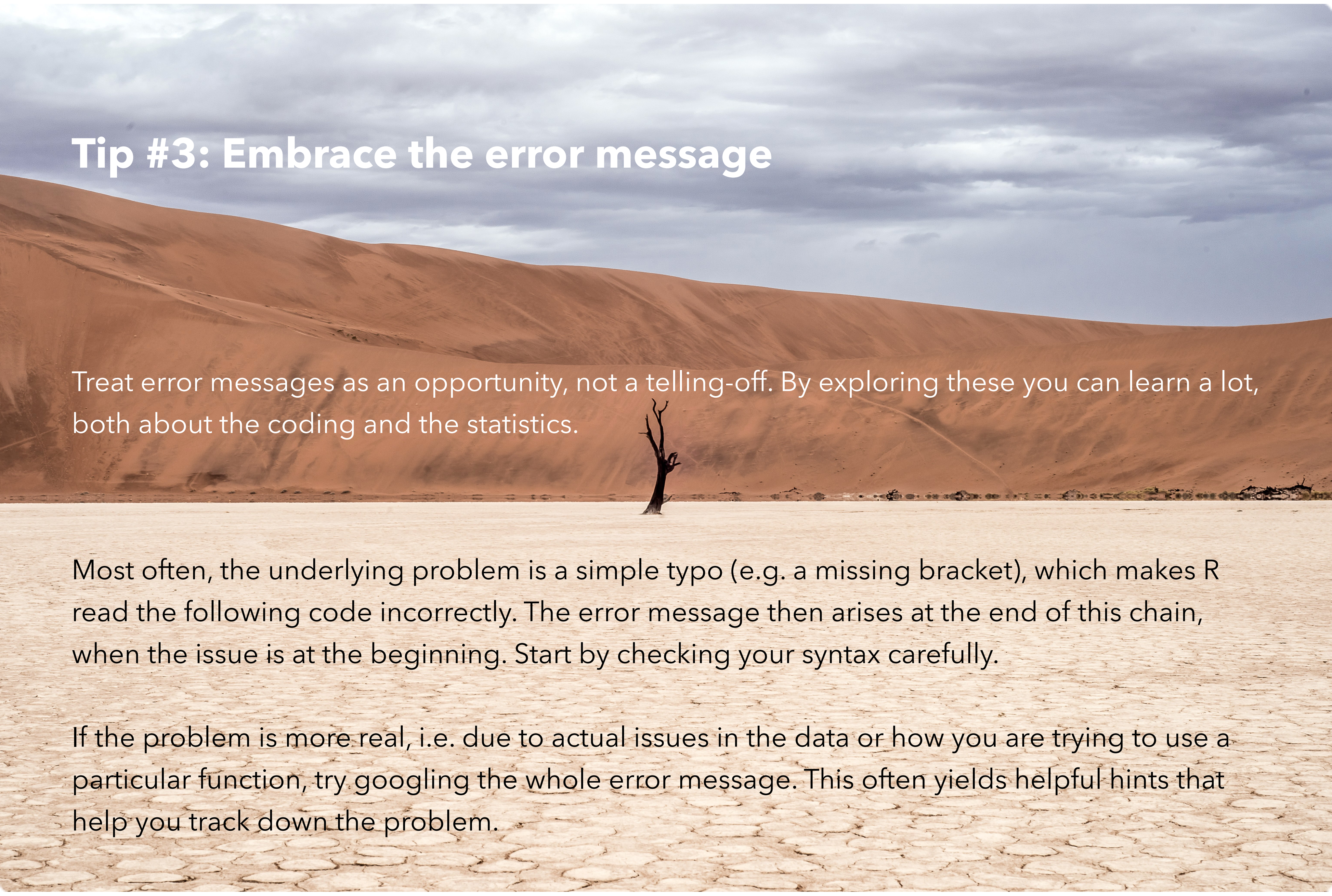
Different sources use different styles of teaching and of coding - different things will work for different people. Try a few before giving up.



Tip #2: You are probably not the first person to have this problem

There are lots of question-and-answer sites dealing with R problems. Good answers usually come up top of the google results. The most reliable are **StackOverflow** and its statistical sub-site **CrossValidated**.

R code is meant to be used by real people, and each function/package comes with **documentation** that explains how it works. Access it from the R window by typing `?functionname` or Google to find the docs from CRAN, the central R site. Some functions/packages also have **vignettes**, with more explicit walkthroughs.



Tip #3: Embrace the error message

Treat error messages as an opportunity, not a telling-off. By exploring these you can learn a lot, both about the coding and the statistics.

Most often, the underlying problem is a simple typo (e.g. a missing bracket), which makes R read the following code incorrectly. The error message then arises at the end of this chain, when the issue is at the beginning. Start by checking your syntax carefully.

If the problem is more real, i.e. due to actual issues in the data or how you are trying to use a particular function, try googling the whole error message. This often yields helpful hints that help you track down the problem.

If you have never coded before, a bit of reading around the theme of "computational thinking" might help you understand how the computer is reading and using the stuff that you input.

If you are struggling to understand the statistics, try exploring your data and analysis ideas first in a different program that you are familiar with, e.g. SPSS. Then you will know what to expect, and it will be easier for you to work out the process in R.

An R glossary

For those new to coding, here are some basic explanations of common concepts. We hope this will help you understand explanations and other resources that you find as you attempt to solve your specific problem. The *word* is given *in italics* and the explanation shows the relevant part of the **syntax or computational thinking** in **bold**, with <u>concepts</u> that are explained elsewhere in the glossary <u>underlined</u>.

argument = the bit that goes inside a <u>function's</u> brackets and gives it its raw material, that it then produces something else from, e.g. chisq_test(**x** = …, **y** = …)

call = to pull out of memory and use, e.g. to run a chi-square, you call the <u>function</u> chisq.test(). In practice, this just means you **type the name** of the <u>function</u> or <u>object</u> you want!

data frame = the equivalent of an **Excel table**, roughly. This is the format your data will mostly be in. Has two dimensions: rows and columns. Columns have names, and each works as a <u>vector</u>.

factor = if a <u>vector</u> consists of <u>strings</u> that only take a few different <u>values</u>, that <u>vector</u> is of the data type "factor" and can be used as a **categorical variable**.

function = the bit of code that does something. Usually followed by brackets, e.g. **chisq.test**(…). A function returns a <u>value</u>, which is often printed by default but usually better stored in a new <u>object</u>.

list = see <u>vector</u>

object = anything containing data, e.g. <u>data frame</u> or <u>vector</u> or single number etc.

string = a <u>value</u> that is a series of characters, as opposed to numerals. In general, anything you would think of as **a word** is a string. R understands things as strings if you put quotes around them, e.g. **"my_data"** is a single string value, whereas **my_data** would be interpreted as a <u>variable</u> and would throw an error if you <u>called</u> it without having defined it first.

v*alue* = the data that is actually inside the <u>object</u>, **e.g. a <u>vector</u> of numbers**. Despite what it sounds like, a value doesn't have to be a single number, or even numerical - it can e.g. be a <u>vector</u> of *strings* (see relevant definitions here) or a <u>list</u> of <u>data frames</u>.

variable = the **name** of anything you have created. You can name variables almost anything you like, excepting most special characters, but avoid <u>function</u> names (can confuse both the program and you). Every <u>object</u> needs its own name, e.g. if splitting or filtering data, give each subset a new name.

vector = a **series of data points** - can be numbers, or factors, or character strings. But only has one dimension (unlike a data frame). Note: in R, a list is something different! For basic usage, you will need vectors a lot more than lists.

A simple example:

```
my_data<-as.vector(c(1,2,3,4,5,6))
```

This line of code creates a vector called `my_data`. It actually contains three functions! Let's read from right to left, to build up our understanding. The function `c()` concatenates its arguments, i.e. the numerals 1-6. Calling the function `as.vector()` around all this makes it explicit that this series of numbers is to be made into a vector. The <- is probably the most common function in R, and assigns a value to a variable. In this case, we assign the vector object to the variable name `my_data`. Now, whenever we want to use the object that is the vector of numbers 1-6, we just call it by typing `my_data` into the console.

If you accidentally put an extra comma at the end of the vector, or forgot to close the brackets for both functions, R would throw an error.