

Introduction to R and RStudio

There are a few different names involved:

- **R** is a coding language for statistics and data analysis
- **RStudio** is a software interface for writing and running R code
- **Posit** is the name of the company that makes RStudio
- **posit.cloud** provides a way of using RStudio in your web browser

You can install R and RStudio on your own computer for free and do things that way, but using posit.cloud simplifies things immensely.

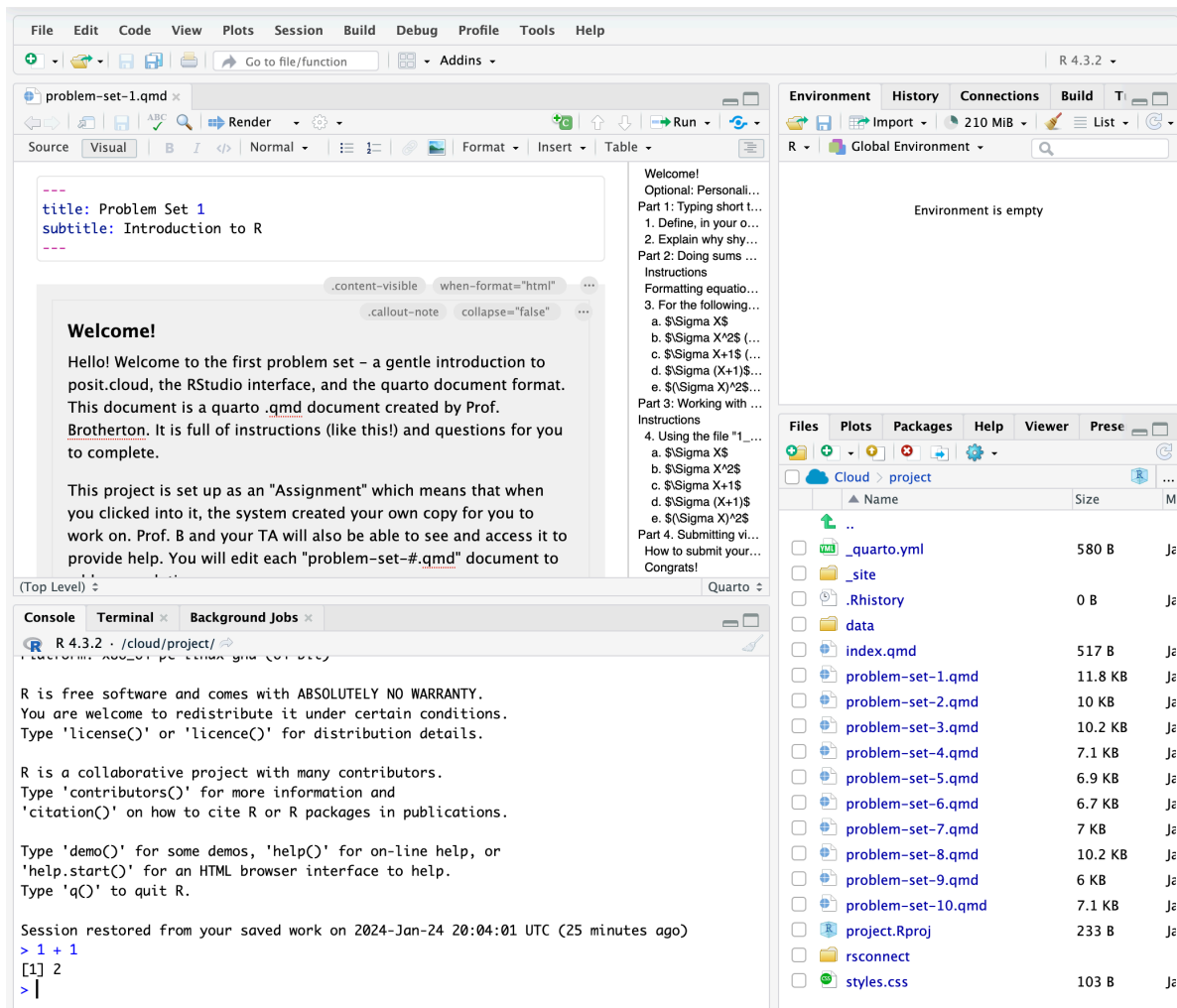
The general workflow

R is a programming language well-suited to interactive data exploration and analysis. It is widely used in social science research.

It might seem daunting if you've have no experience with coding, but the basic idea is that you have some data, like you are familiar with from a regular Excel or Google Sheets spreadsheet, and you perform operations on your data using functions a lot like you would in Excel/Sheets. For example, you might compute an average in Sheets by typing `=AVERAGE(A1:A10)`. In R you might type `mean(my_data$column_a)`. The specifics of the function names are different, but the basic idea is the same.

RStudio Interface

RStudio is the interface we'll use to write and run R code and see its output. The basic interface has 4 panels, each with a few tabs:



- Top-left: Code editor / data viewer
 - Open, edit, and save code documents
 - Execute code within files
 - View data
 - You can have multiple ‘tabs’ open at once,
- Bottom-left: R console
 - You can type code directly and run it by pressing enter.
 - You won’t be saving your code as a document like when you type in in the editor, so this is useful for testing something simple out
- Top-right: Environment
 - As you execute code you may be creating objects like sets of numbers or data.frames. Those objects will appear here.

- You can click the name of some objects, like `data.frames`, and it will open a view of the data as a tab in the editor pane
- Bottom-right: Files/folders, Plots, Viewer, help window
 - You can navigate the file tree

Running code

Writing some code in an `.R` document does not cause it to be executed automatically. You need to run the code yourself. There are several ways of doing so.

Additional packages

The R language has many functions built in. Generally speaking, you can find a way to do pretty much anything you would like to do using just ‘base’ R.

However there are many common tasks that are a bit tedious or unintuitive to do using base R. One of R’s strengths is how extensible it is: anyone can write their own functions, turn the code into an R package, and make that package available to other R users.

Tidyverse. Actually, the tidyverse package is a container for multiple individual packages. The whole family of tidyverse packages are written with a consistent syntax and logic.

```
install.packages("tidyverse")
```

```
install.packages("lme4")
```

Packages only need to be installed on your system once (or once per project in `posit.cloud`, since every cloud project represents a brand new virtual system).

However, you must ‘activate’ the package when you want to use its contents. This must be done every time you have a new ‘session’.

```
library(tidyverse)
```

```
library(lme4)
```

As an alternative, if you are just using one function from a package as a one-off, you can use the double-colon `::` operator in the form `package::function()`, i.e.

```
lme4::lmer(...)
```