R Coding Practical Session

Arianna Krinos

WHOI Summer Math Review 2023

Icebreaker – carousel activity!

- Around the room you'll find a few prompts about problems you might encounter that can be solved using R programming language tools
- As you go around, brainstorm in your small groups some ways you would approach solving this problem
- Jot down
  - Approaches
  - Terms that you think are important
  - Even code snippets or functions! These can be from other languages

Learning objectives for this session

- Link **core concepts** in programming (even from other sessions) to the use of the R statistical computing language
- Be able to identify code-based solutions for common **practical applications** in science
- Be able to generate bar, line, histogram, scatter, and heatmap plots **in ggplot2** from example data
- Recognize the R framework for a **statistical regression model**

The R statistical computing language: what it is an is not

What R is

- Flexible, user-driven community of source code repositories
- Optimized for ease of use and user experience
- A great choice for tabular data, in particular spreadsheets with a lot of attribute data of interest
- A great choice for plotting out-of-the-box
- A great choice for powerful graphics

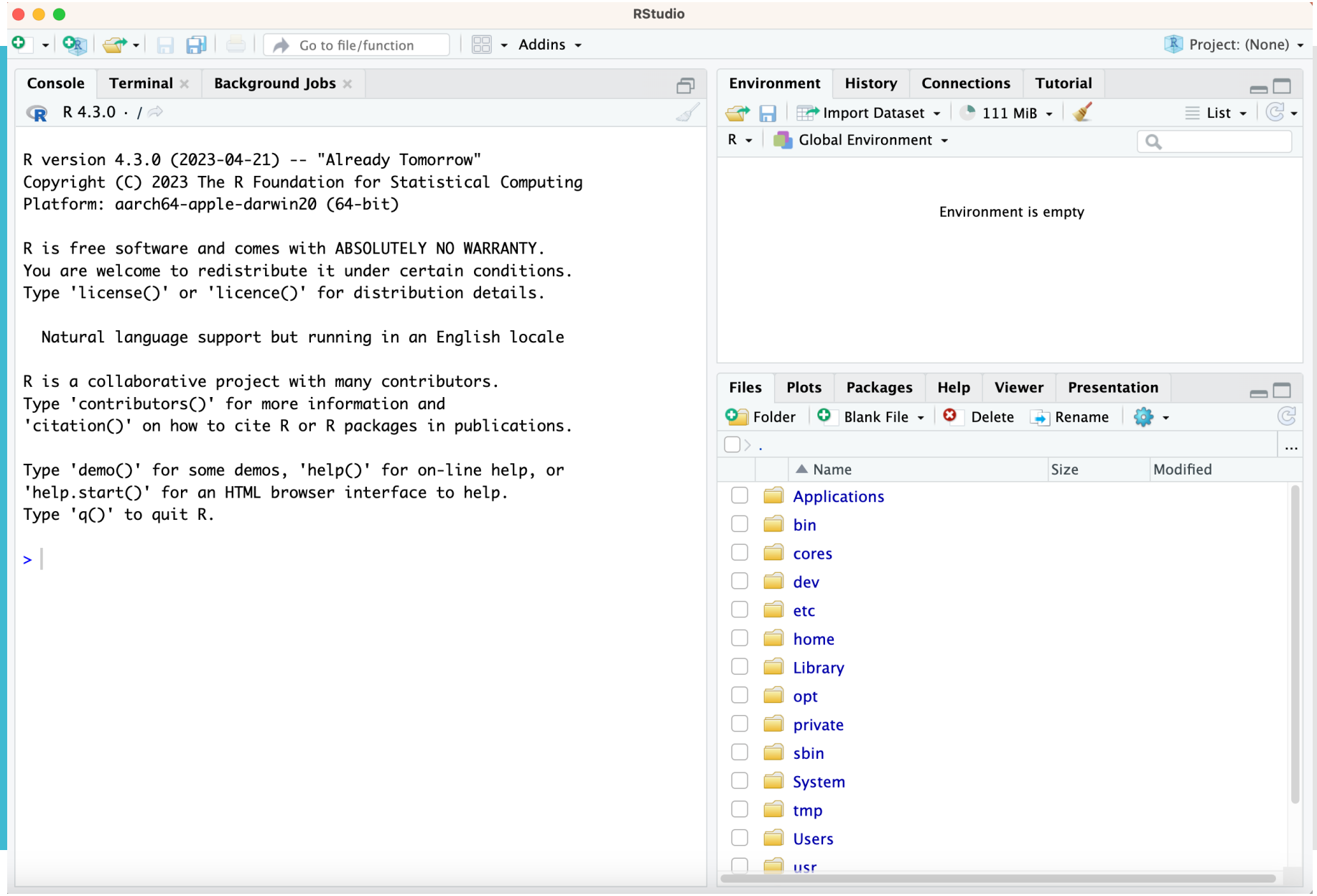
What R is not

- A particularly fast or memory-efficient programming language
- A good choice for object-oriented programming

RStudio – who has the software installed?

- Follow the links here: https://swcarpentry.github.io/r-novice-gapminder/ on the Software Carpentries page to install the R programming language

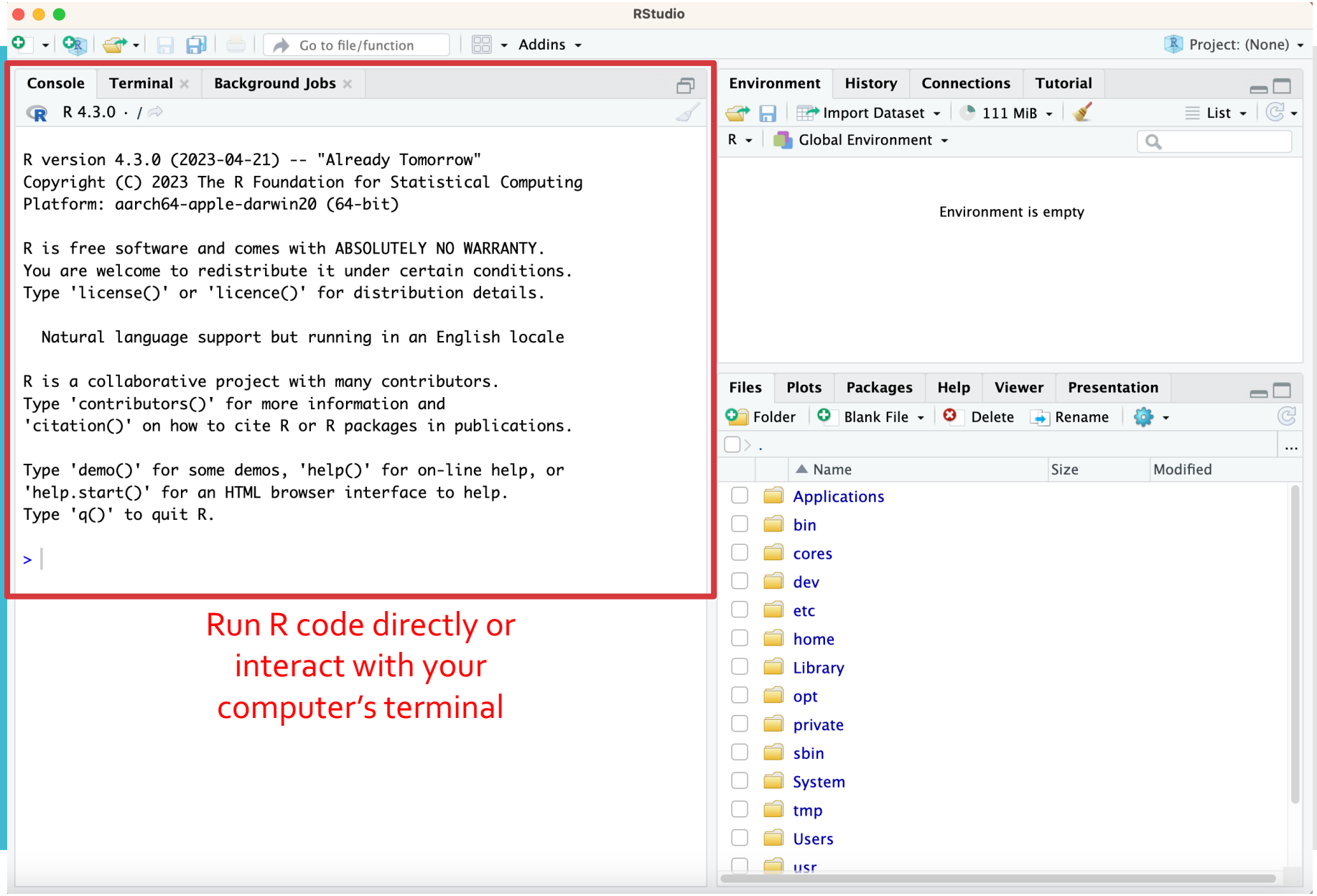
The RStudio interface



What is RStudio?

- A [well-maintained & open-source] example of an **integrated development environment**
- This means that all inside RStudio, we can
  - Access the console & the file system
  - Draft and execute code
  - Install packages
  - Knit and pretty-publish coding outputs

Diagram of the RStudio interface



Run R code directly or interact with your computer's terminal

Diagram of the RStudio interface

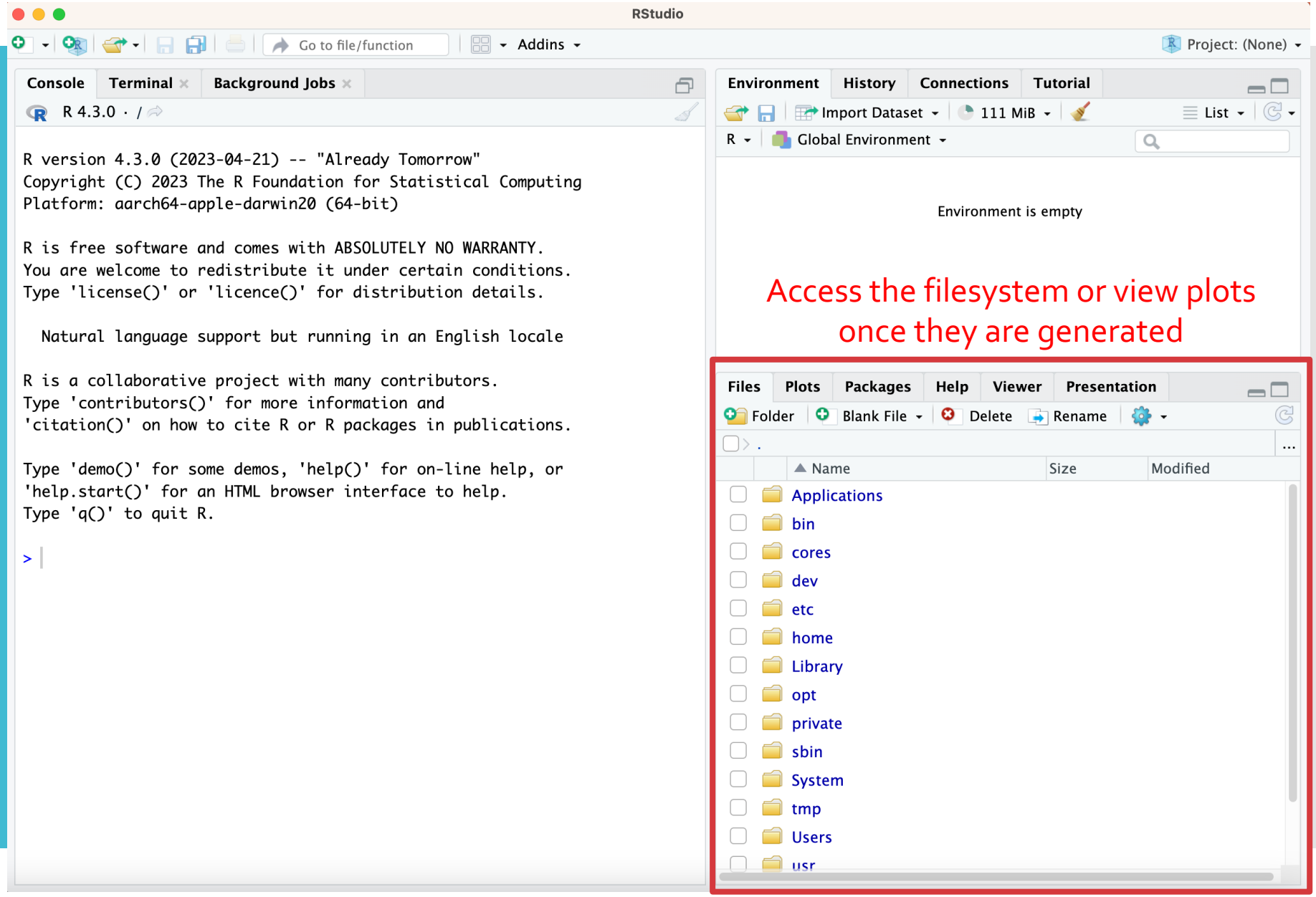
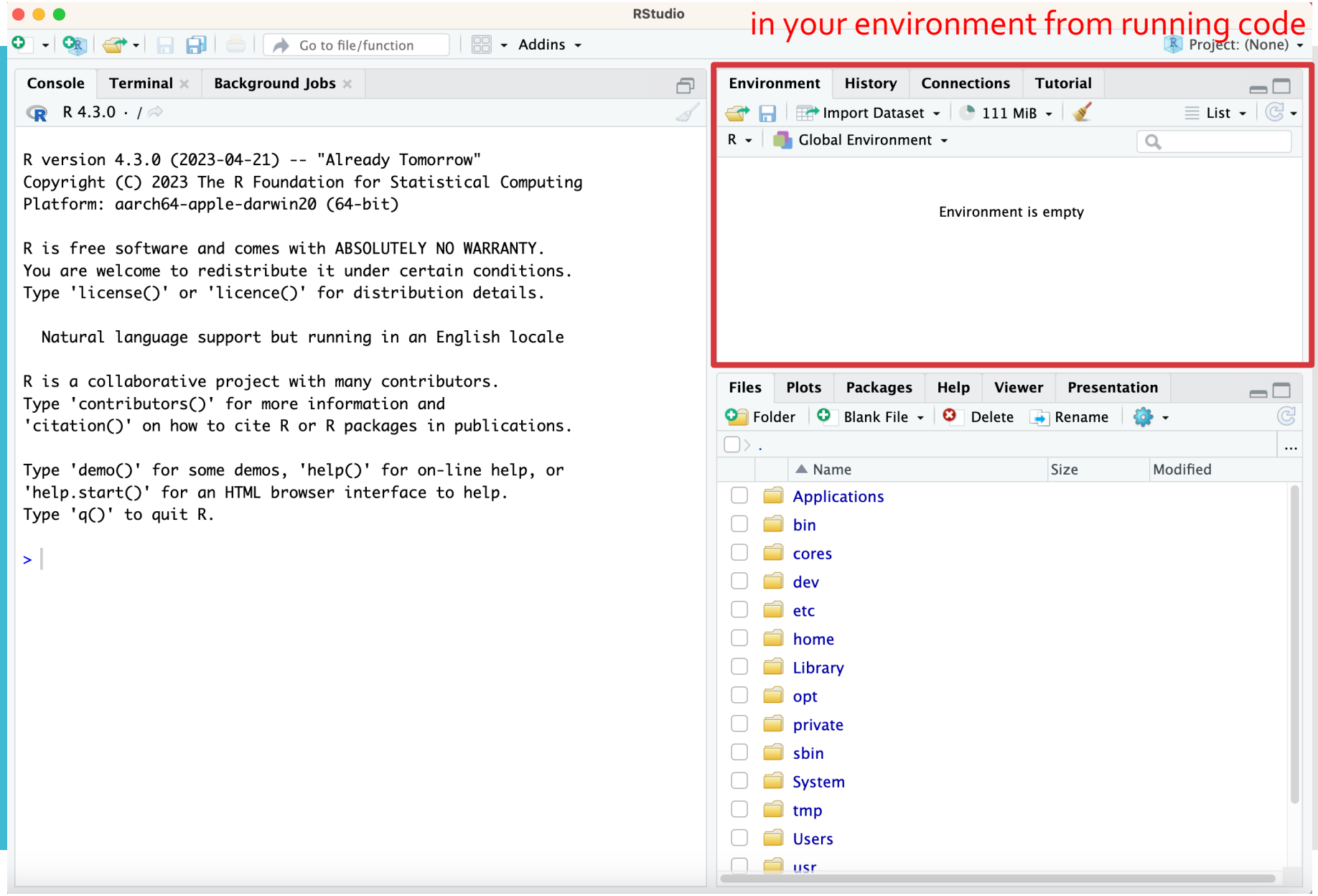


Diagram of the RStudio interface

View which variables have been stored in your environment from running code



Some key concepts to keep in mind

- Data structure: a means of organizing data into a format that a computer (and, hopefully, also you) can read
- Vectorization: instead of repeating the same operation many times, it's cleaner and more interpretable to convert the data to a list of items & tell R to apply the same step to everything in the vector
  - Matrix-wise vs. element-wise operations: in many programming languages, matrix-wise operations are default, in R it will be the opposite
- Functions: A way of organizing code into small, repeatable chunks that can be used again and again
- Hard-coding: Taking values that we expect will change and instead writing a very specific value in our script

The five data types in R

1. Double
    1. Floating point number (e.g. 4.32)
2. Integer
    1. "Number line" number (…-3,-2,-1,0,1,2,3…)
3. Complex
    1. A number with an imaginary + real component (1+1i)
4. Logical
    1. TRUE or FALSE Boolean result
5. Character
    1. A string of length zero or more

You can check type using the **typeof** command, or view other attributes using **str**

Errors you might run into

- In R, integers and doubles can be combined without a problem – this is part of R's flexibility
- However, if you e.g. try to add a string and a double, you'll get an error
- "Non-numeric argument to binary operator" is the error you'll get
- Unlike in Python, strings can never be added to represent concatenation

Vectors in R

- Vectors can be created in base R using **c()**, the combine function
- Any vector can be added to using c() as well – the elements will simply be concatenated into a new vector
- Vectors can be coerced into different types all at once the same way that individual variables can (a vector must all be of the same type)
  - as.double
  - as.numeric
  - as.character

Named vectors are possible in R

- Similar to a column slot in a data frame, we can name the elements of a vector in R, e.g. using c(name1 = ”value1”)
- We can only access the elements of named lists using bracket syntax, not using other operators
- You can access or change all the names at once in a vector using the names() function

Data frames in R

- We can create a dataframe using the data.frame function, however for most applications you will read in a CSV instead
- To create a dataframe, you use the data.frame function, e.g. data.frame("Hello"=c(1,2,3)).
  - Every column name will be in quotations and the contents of the column will be expressed as a vector.

Data frames in R

- Most applications involve reading in a text file, separated by tabs, commas, or spaces
- This command is read.csv or read.table
- You can change the separator to indicate what character you want to signal a column separation in either case

Modifying data frames

- Important commands: nrow(), ncol(), colnames()
- We can slice data frames using numerical indexers and remove elements using negative numbers
- We can add columns using **cbind** and add rows using **rbind**
  - Using these commands requires all dimensions to be consistent between the data frame and the new content
- In our practical, we will encounter the powerful dplyr package, which will help us seamlessly make row and column changes and perform calculations all within a single command

Let's solve our icebreaker problems using R

- Moving to live coding and demos! Open up RStudio if you have not already.