Introduction to R and RStudio

Session outline

1. What is R?
2. Why should we use R?
3. R and RStudio
4. Packages
5. Using the console
6. Scripts

What is R?

- R is a free, open source language for statistical computing.

  1992 Development began as research project in Auckland, by Robert Gentleman and Ross Ihaka, based on the S (1976) and S-PLUS (1988) languages.

  1993 First release of R

  2000 Version 1.0 released

  2004 First UseR! conference in Vienna, Austria

  2013 Version 3.0 released

  2015 R Consortium founded

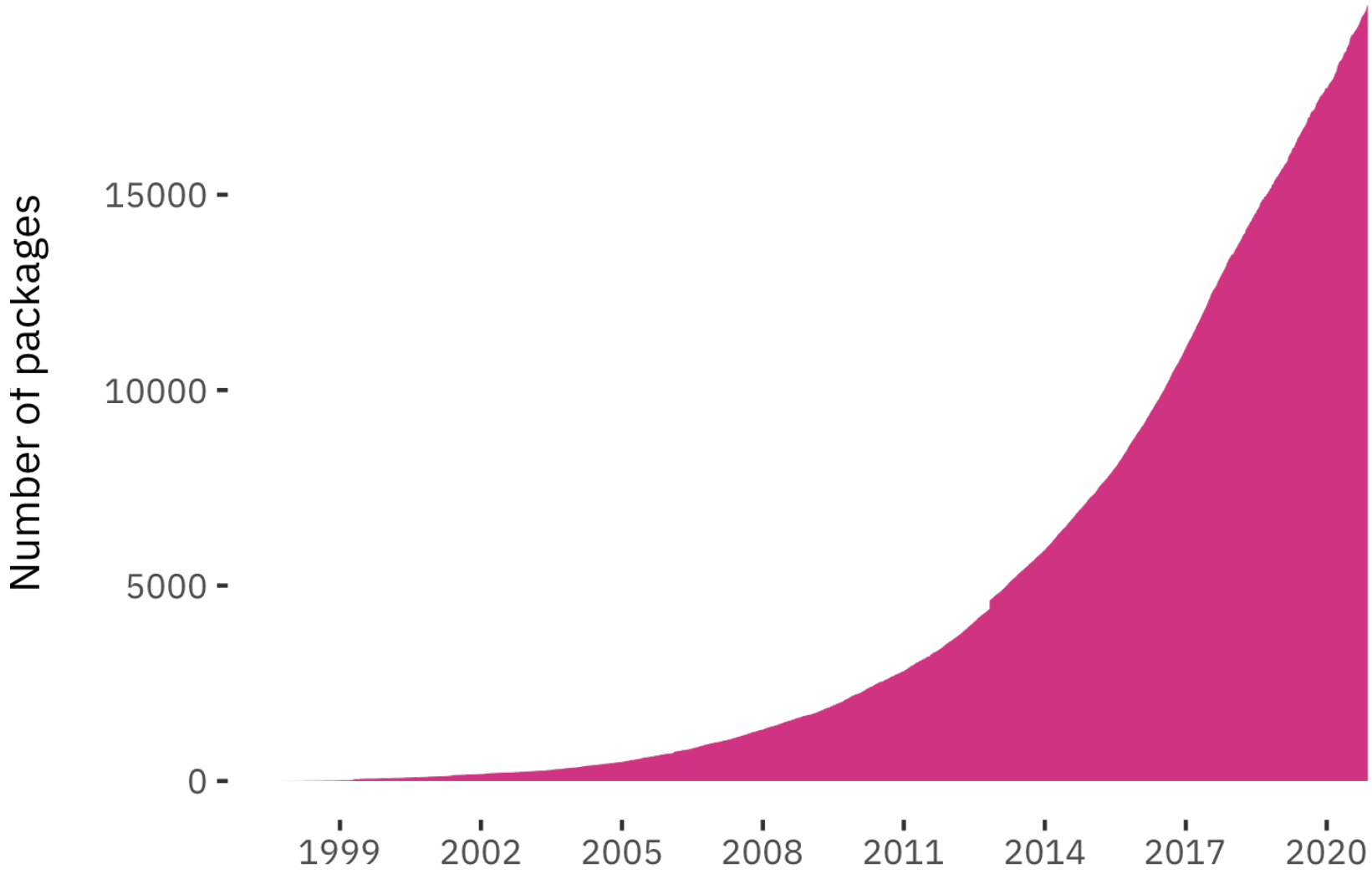
  2017 10,000 packages published on CRAN

  2020 Version 4.0 released

- R is one of the fastest growing programming languages, especially for statistics and data science.

- A big, friendly ecosystem.

Over 23,000 packages (as of 2026)



https://gist.github.com/daroczig/3cf06d6db4be2bbe3368

Why use R?

Strengths

- Free and open source
- Big user community
- Excellent resources
- Incredibly flexible
- >23k packages
- Rapid development
- Industry adoption
- It's a programming language

Why use R?

Strengths

- Free and open source
- Big user community
- Excellent resources
- Incredibly flexible
- >23k packages
- Rapid development
- Industry adoption
- It's a programming language

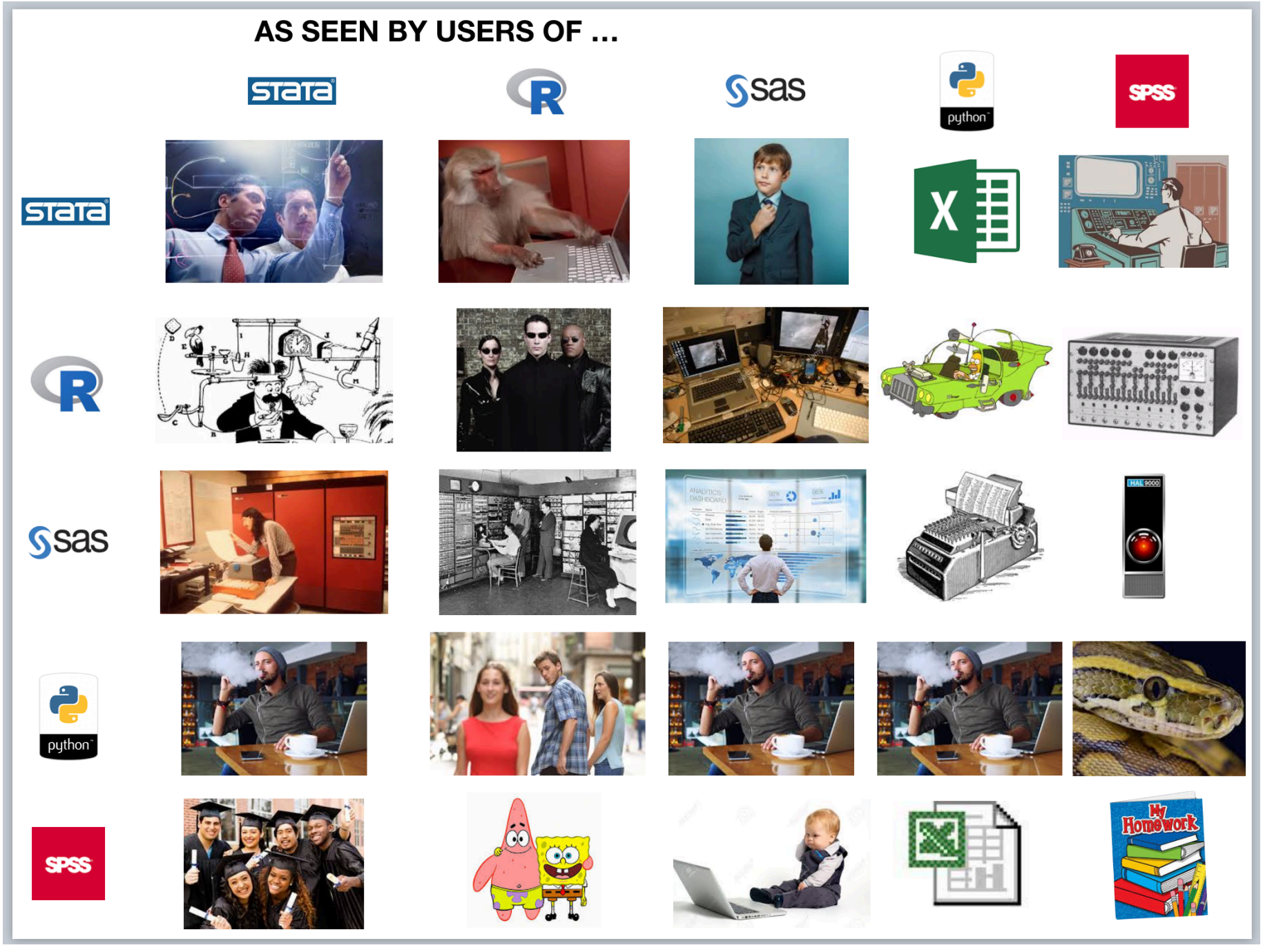
Weaknesses

- >23k packages. Which should I use?
- Rapid development
- Package management can be a pain
- ~~R is slow~~

  (Less true today; many functions written in faster, low-level languages likeC++ or Fortran.)
- It's a programming language

	R	Python	Julia	Stata	C++
					
Licence	Open	Open	Open	Closed	Open
Version 1.0	2000	1994	2018	1985	1985
Easy of use*	Moderate	Hard	Moderate	Easy	Very hard
Extensibility	Excellent	Good	Good	Limited	Very limited
Industry support	Excellent	Excellent	Limited	Limited	Good
Performance	OK	Good	Very good	Good	Excellent
Execution	Interpreted	Interpreted	JIT	Interpreted	Compiled

*For statistical computing.



Source: https://kieranhealy.org/blog/archives/2019/02/07/statswars/

Should I learn R or Python?

RStudio

Installing R and RStudio

R is a programming language.

RStudio is a software environment for working with R.

RStudio is developed by Posit, a data science company that builds tools for R, Python, and publishing.

You need both R and an editor (like RStudio) to get started.

Installing R and RStudio

R is a programming language.

RStudio is a software environment for working with R.

RStudio is developed by Posit, a data science company that builds tools for R, Python, and publishing.

You need both R and an editor (like RStudio) to get started.

Other options you might consider:

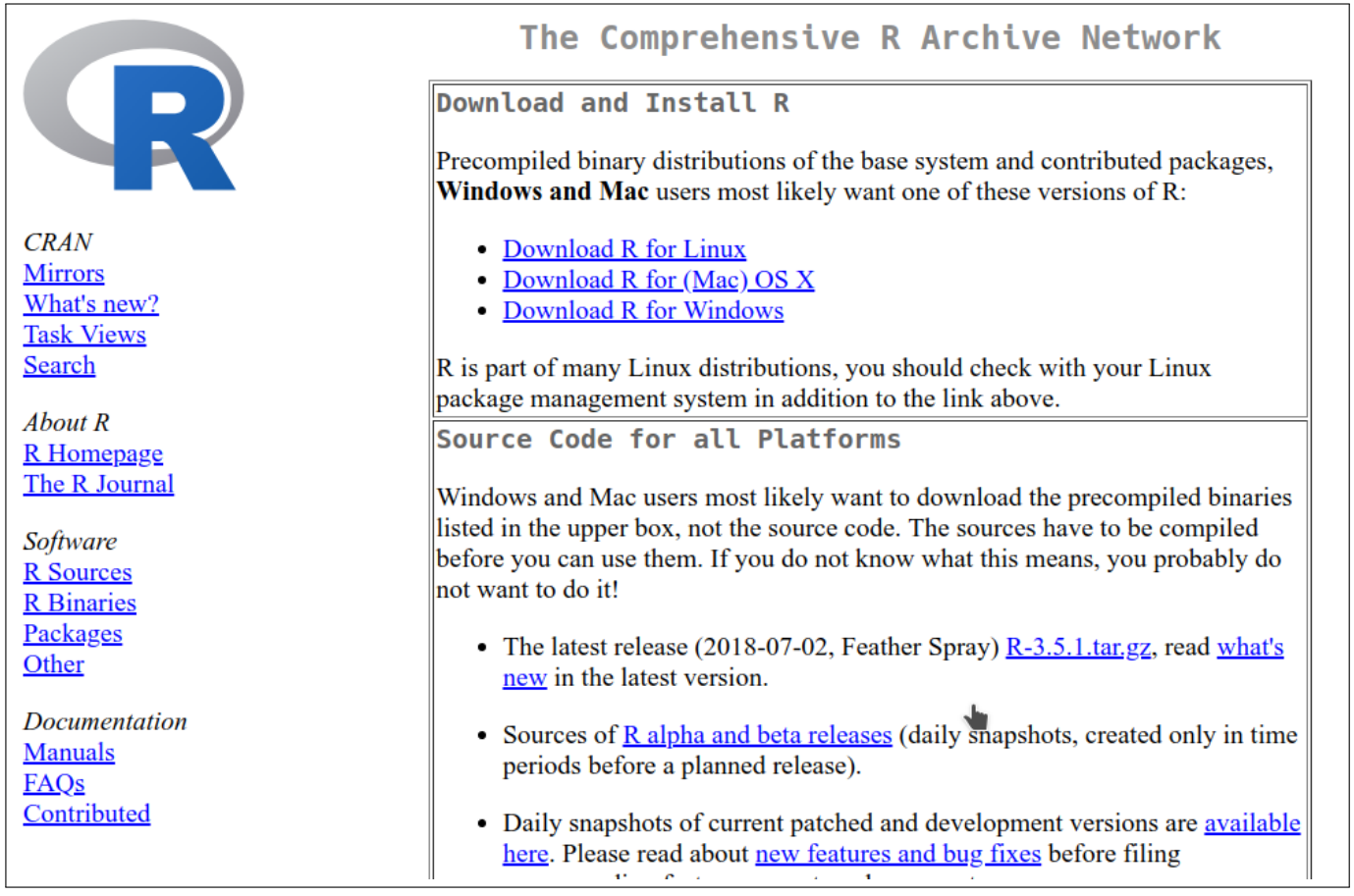
- Positron — a "data science code editor"

  `https://posit.co/products/ide/positron/`

- VS Code: a popular general purpose editor

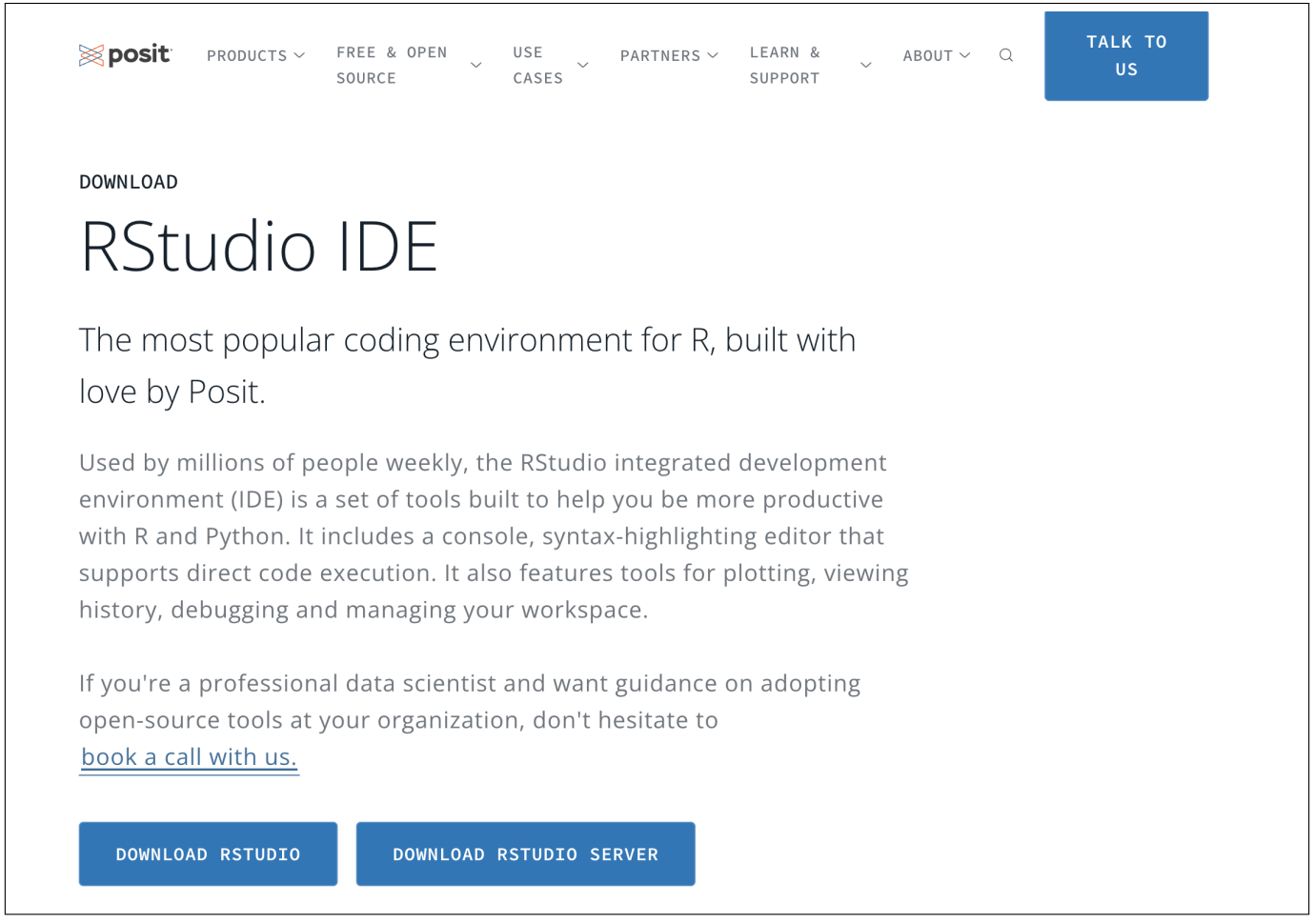
  `https://code.visualstudio.com/`

Install R from CRAN

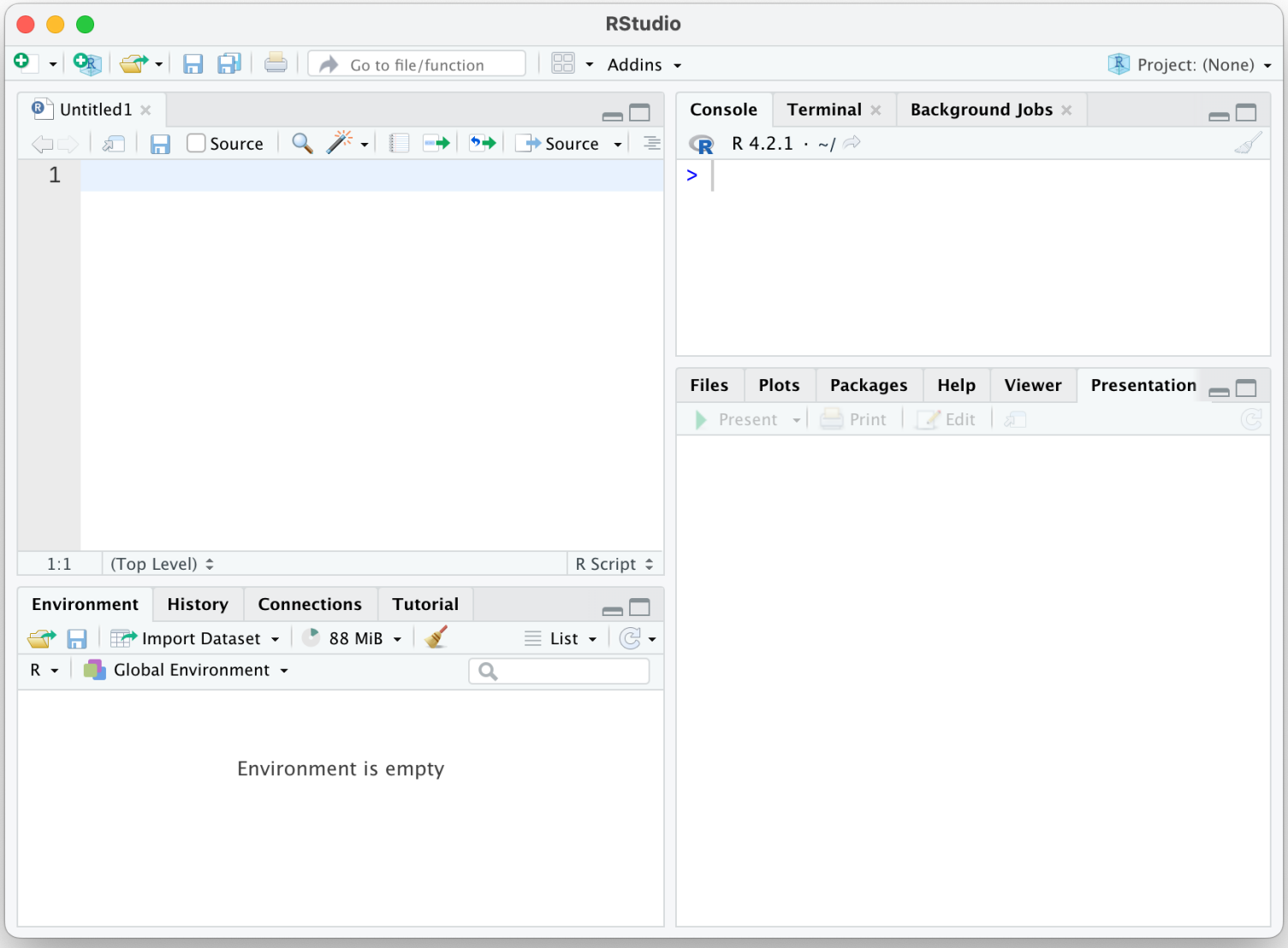


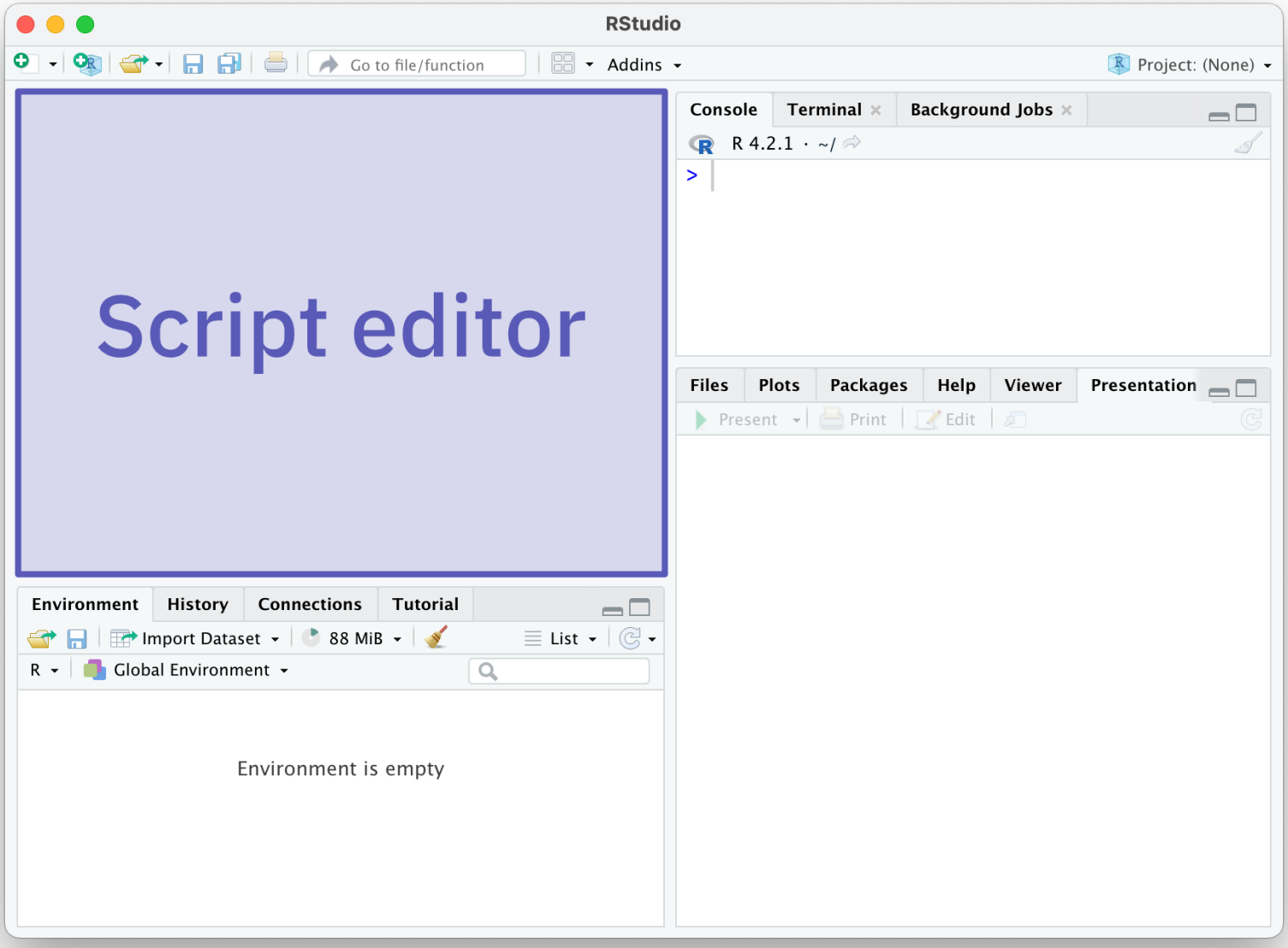
cran.r-project.org

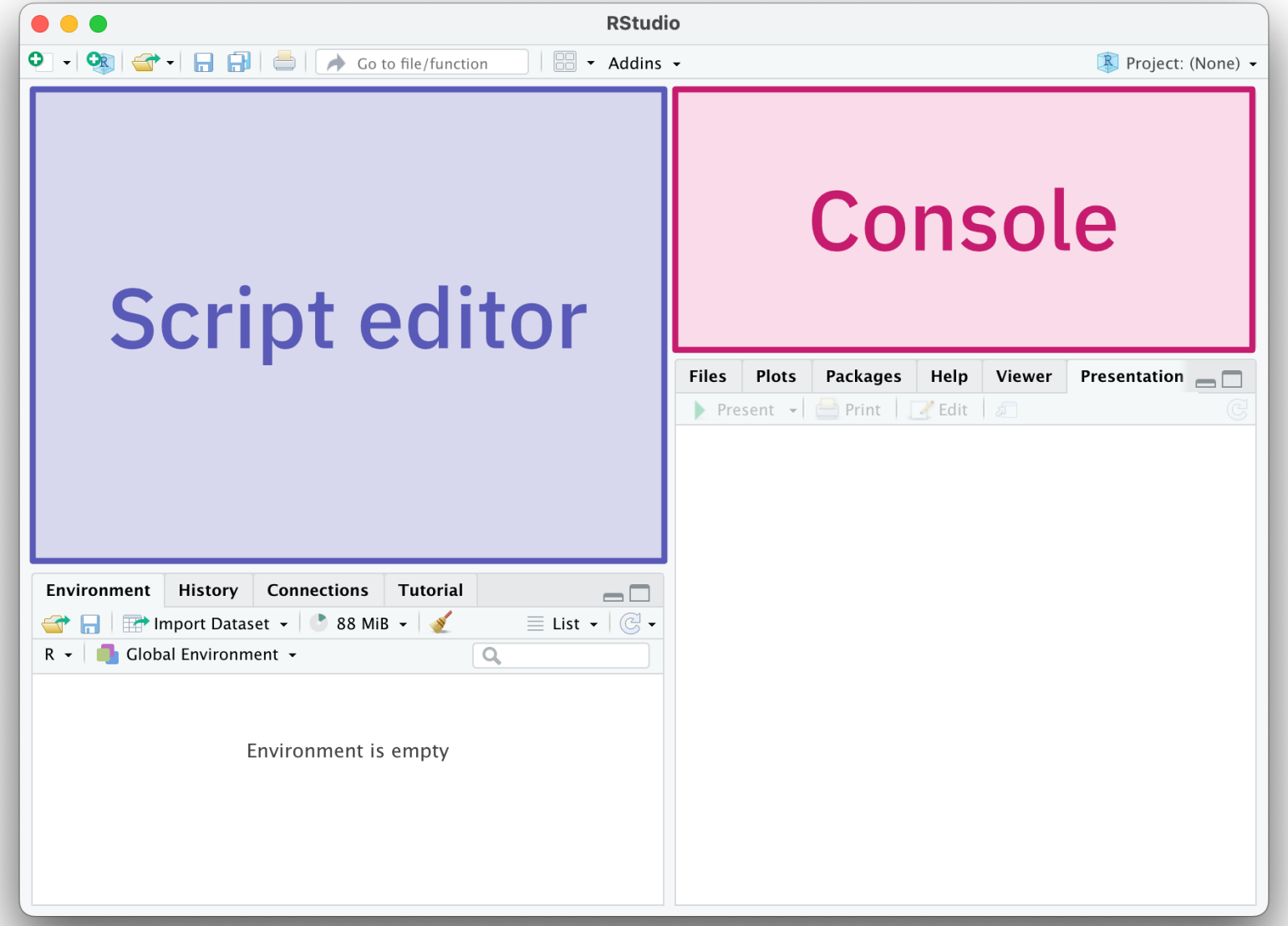
Install RStudio from Posit

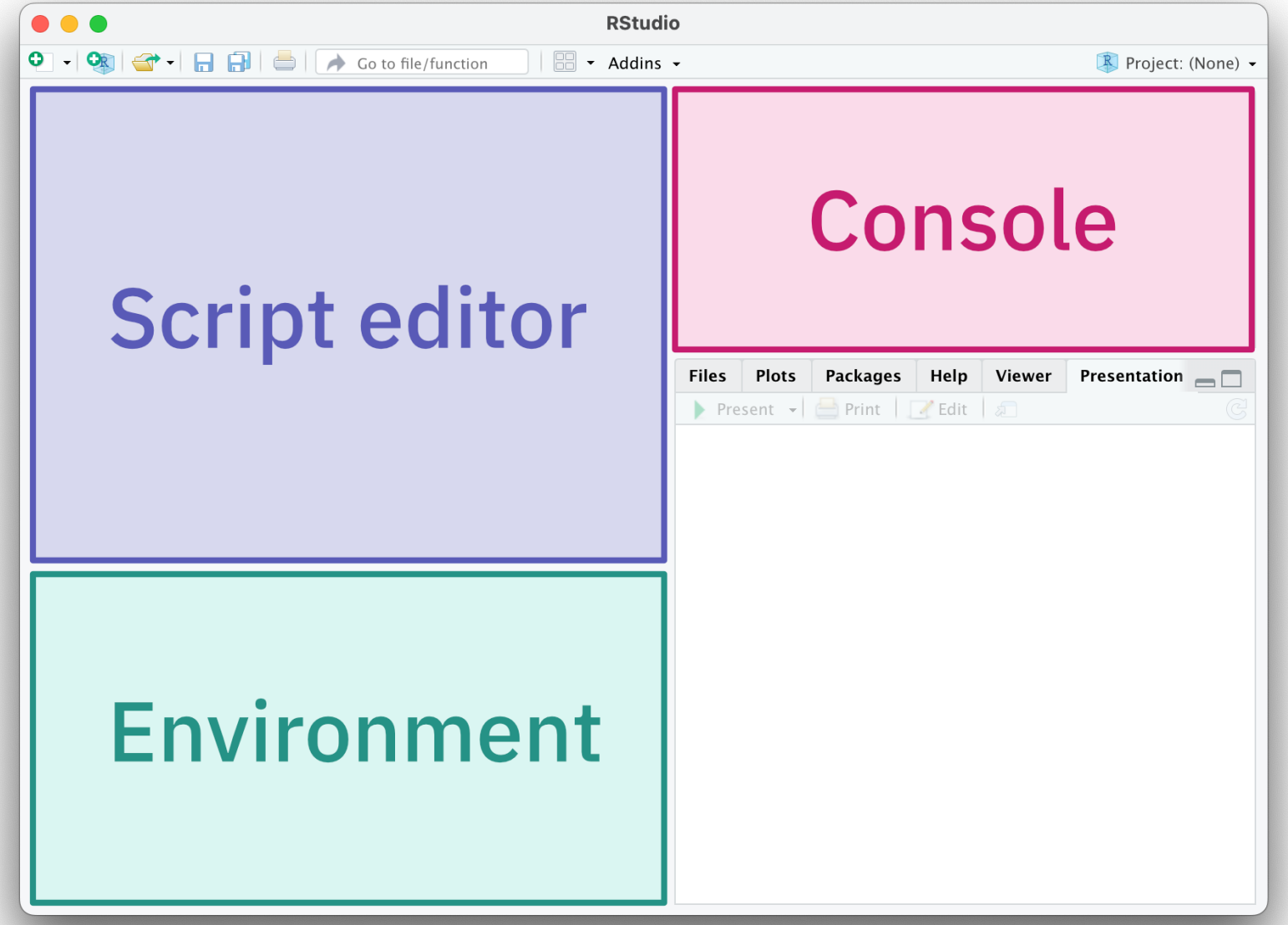


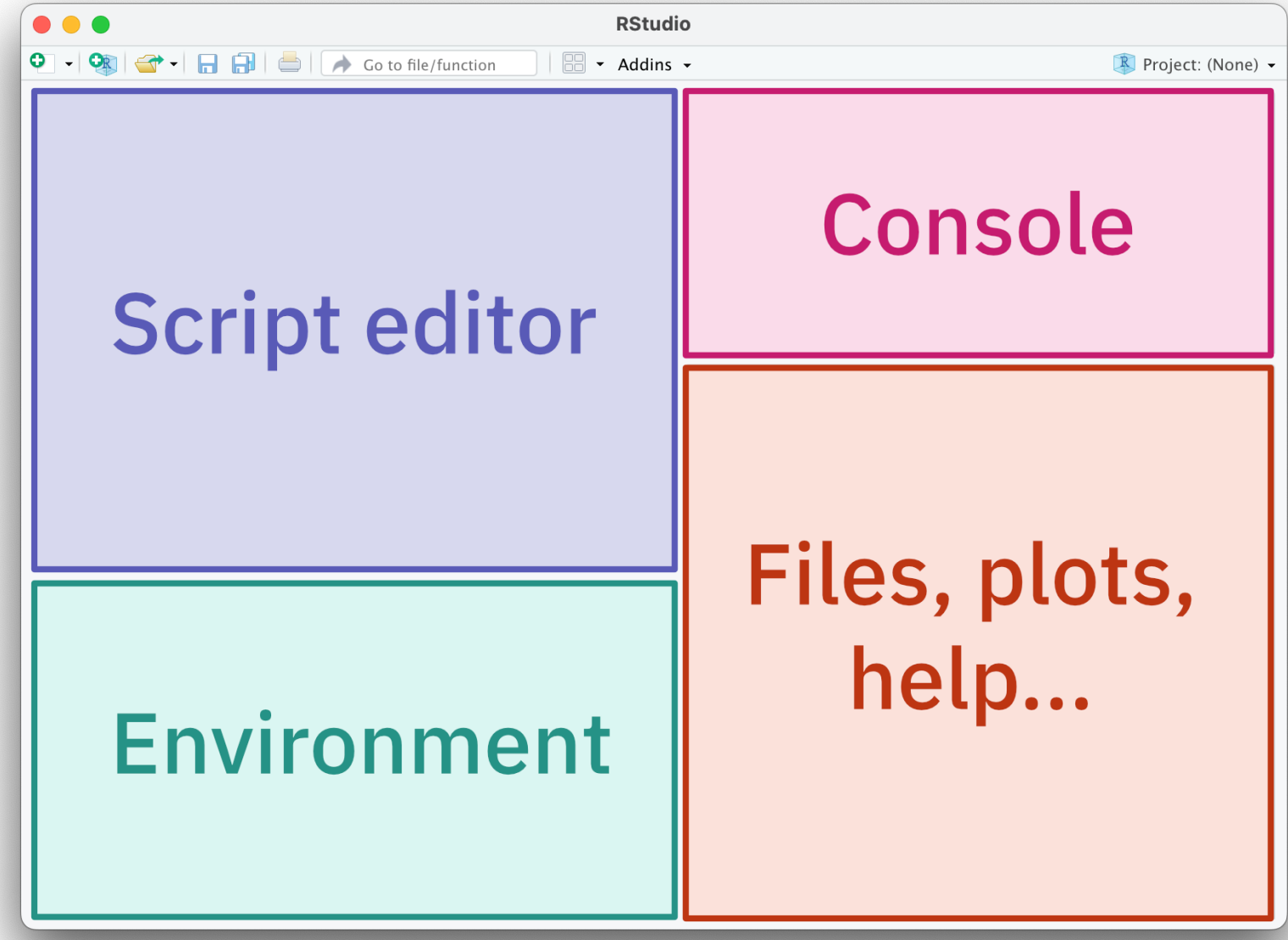
`posit.co/download/rstudio-desktop`


RStudio
Go to file/function
Addins
Project: (None)
Untitled1
Source
Source
1
1:1
(Top Level)
R Script
Environment
History
Connections
Tutorial
Import Dataset
88 MiB
List
R
Global Environment
Environment is empty
Console
Terminal
Background Jobs
R 4.2.1 · ~/
>
Files
Plots
Packages
Help
Viewer
Presentation
Present
Print
Edit

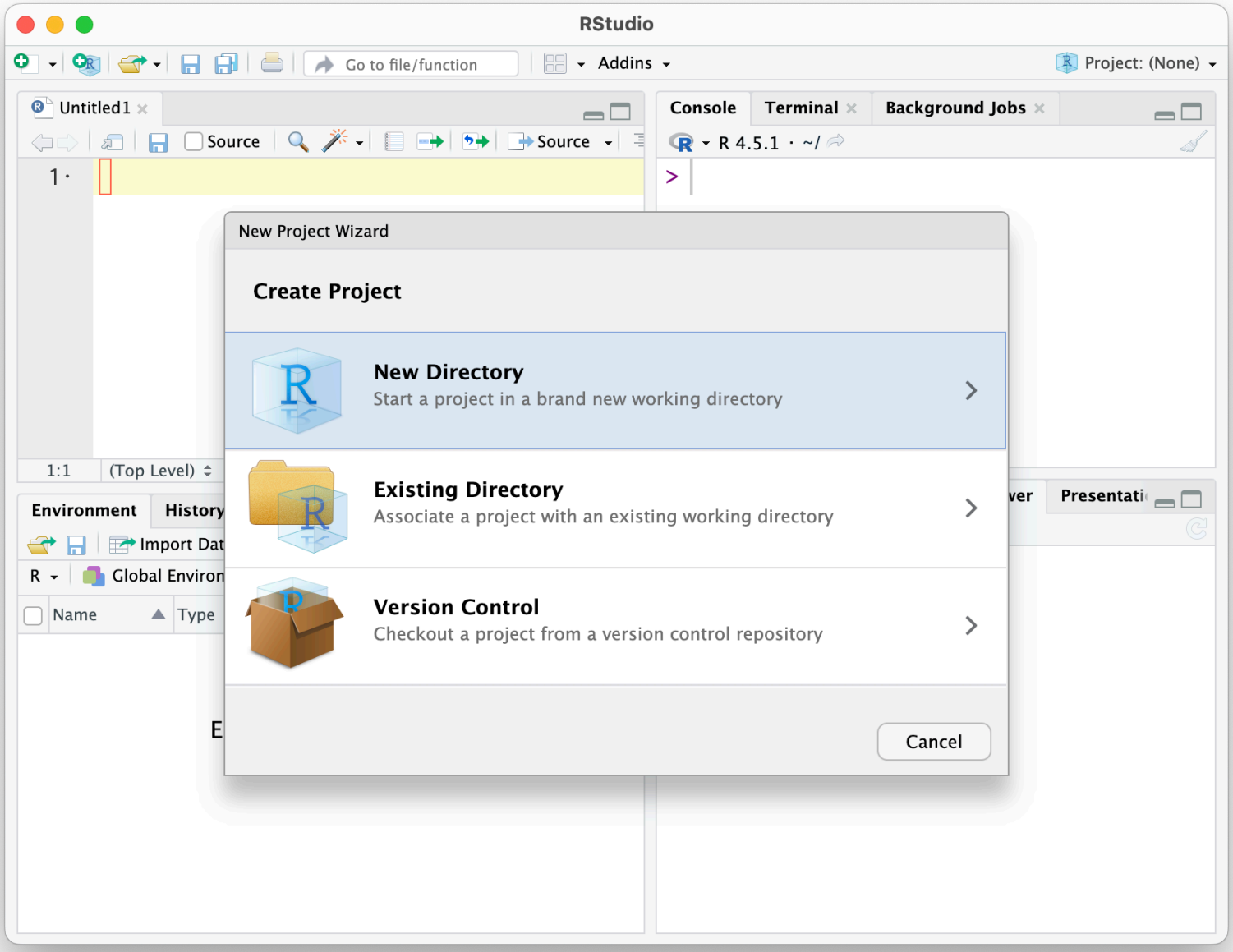

RStudio
Go to file/function
Addins
Project: (None)
Script editor
Console
Terminal
Background Jobs
R 4.2.1 · ~/
>
Environment
History
Connections
Tutorial
Import Dataset
88 MiB
List
R
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Presentation
Present
Print
Edit


RStudio
Go to file/function
Addins
Project: (None)
Script editor
Console
Environment
History
Connections
Tutorial
Import Dataset
88 MiB
List
R
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Presentation
Present
Print
Edit


RStudio
Go to file/function
Addins
Project: (None)
Script editor
Console
Files
Plots
Packages
Help
Viewer
Presentation
Present
Print
Edit
Environment


RStudio
Go to file/function
Addins
Project: (None)
Script editor
Console
Environment
Files, plots, help...

Projects in RStudio



R packages

R packages

We use packages to add functionality to R.

> *A collection of scripts designed to implement a particular function or method.*

They represent one of the principle strengths of R: if a method exists, someone has probably written a package.

R comes with many packages out-of-the-box, such as `datasets`, `graphics`, and `stats`. These are referred to as 'base' packages.

Installing packages

We can install packages by typing:

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

To load this package, we'd then type:

```
library(lme4)
```

(Note the use of quotation marks.)

You should install packages once. But you need to load them every time you use them.

How to choose which packages to use?

My preferences:

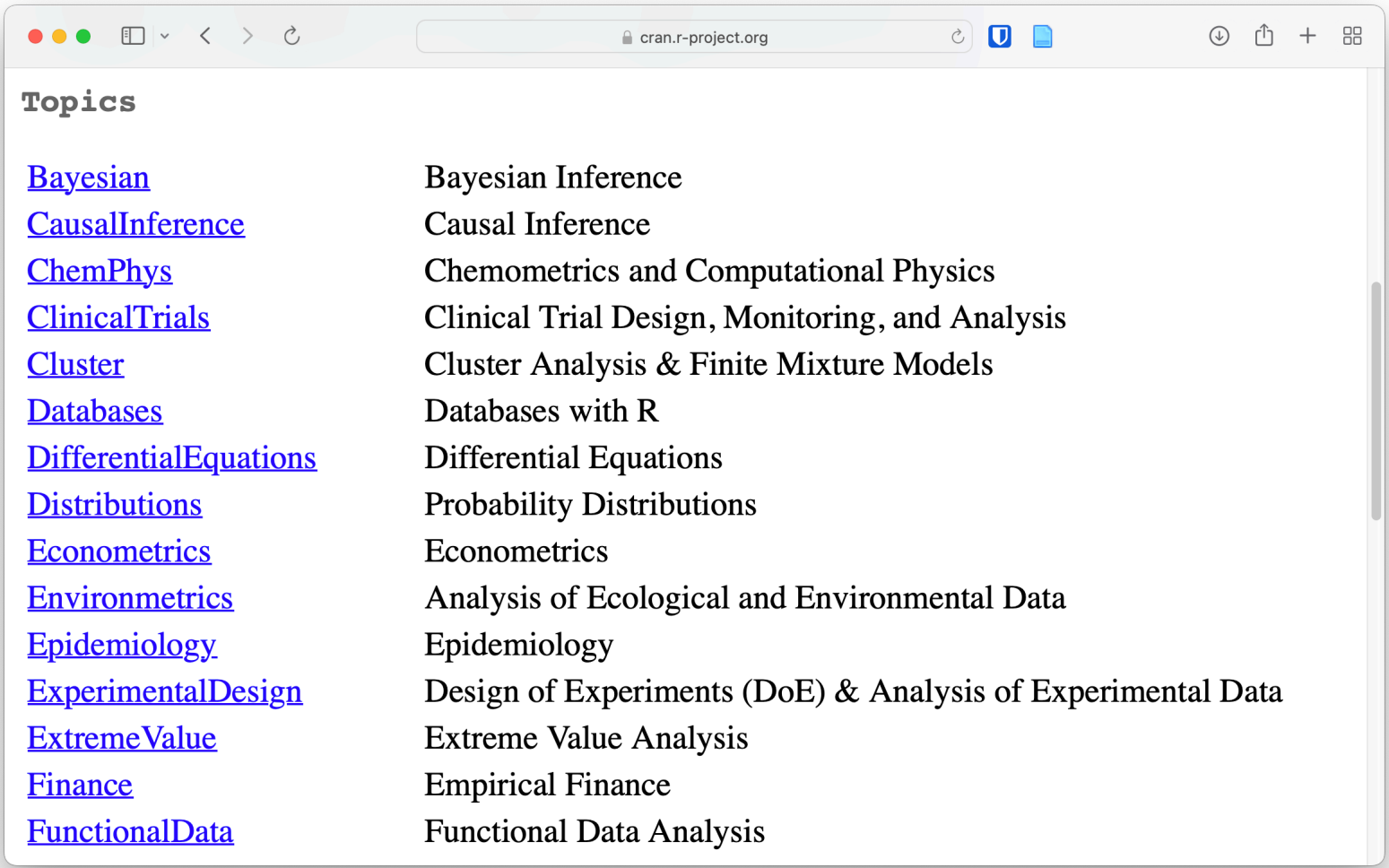
1. Use as few packages as possible (but as many as needed).
2. Use big general packages (rather than small specialised ones).
3. Use packages that have been around for a long time and are regularly updated.
4. Use stable, reliable packages.
5. At times, ignore all these rules.

Your preferences may differ!

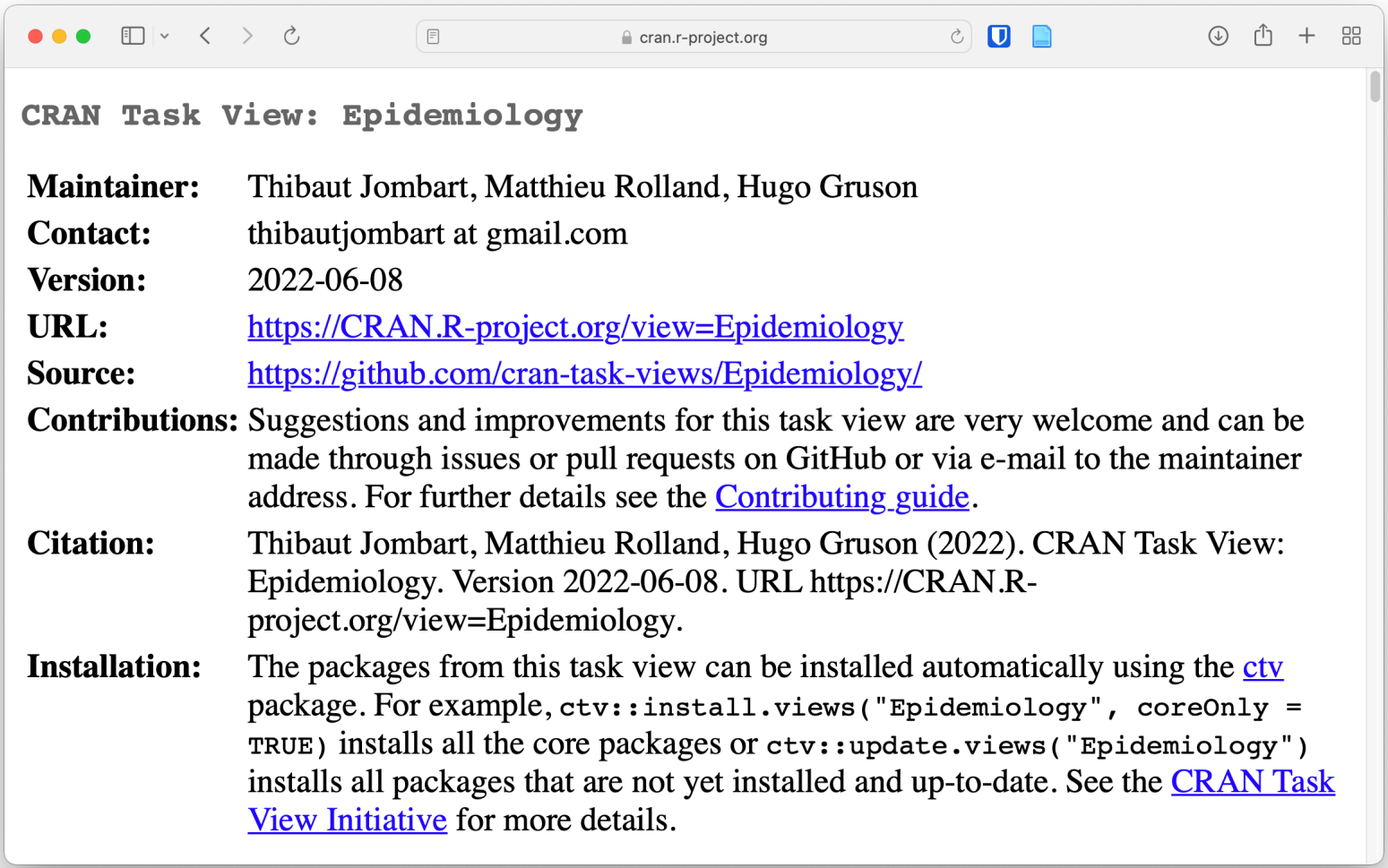
For example: `ggplot2`, `dplyr`, `tidyr`, `data.table`, `lme4`

Use the CRAN Task Views

`cran.r-project.org/web/views`



cran.r-project.org/web/views/Epidemiology



Tip: Install packages with pak

pak

> A Fresh Approach to R Package Installation

pak installs R packages from CRAN, Bioconductor, GitHub, URLs, git repositories, local files and directories. It is an alternative to `install.packages()` and `devtools::install_github()`. pak is fast, safe and convenient.

- 🚀 Short tour
- 🔗 Quick links (start here if in doubt!)
- ✨ Features
- ⬇️ Installation
- 📘 License

LINKS

View on CRAN

Browse source code

Report a bug

LICENSE

GPL-3

COMMUNITY

Code of conduct

CITATION

Citing pak

DEVELOPERS

Gábor Csárdi
Author, maintainer

```
install.packages("pak")
pak::pak("dplyr")
```

Scripts

Scripts are text files containing R code

Similar to Stata do-files or SPSS syntax files.

```r
# Example of a script
# Ewan Carr
# 2026-03-02

library(tidyverse)
library(lme4)

# Import raw data
raw <- read_csv("raw.csv")

# Fit a linear model
model <- lm(y ~ x1 + x2, data = raw)
```

Header explaining what the script does

Load packages at the top

Comments

Scripts (cont.)

- You should annotate your scripts with comments.

```
# This is a comment.
```

What makes a good comment?

- Save your scripts with the `.R` file extension

e.g., `1-Data-prep.R`.

- Install packages once, load them every time you use them. Place at the top of your script.

```
library(tidyverse)
library(here)
```

Use the console for exploration; use scripts for anything you want to keep.

Organising your scripts

You can organise your scripts with sections and headers.

```
# Section One ----------------------------

# Section Two ============================

### Section Three ##########################
```

Organising your scripts

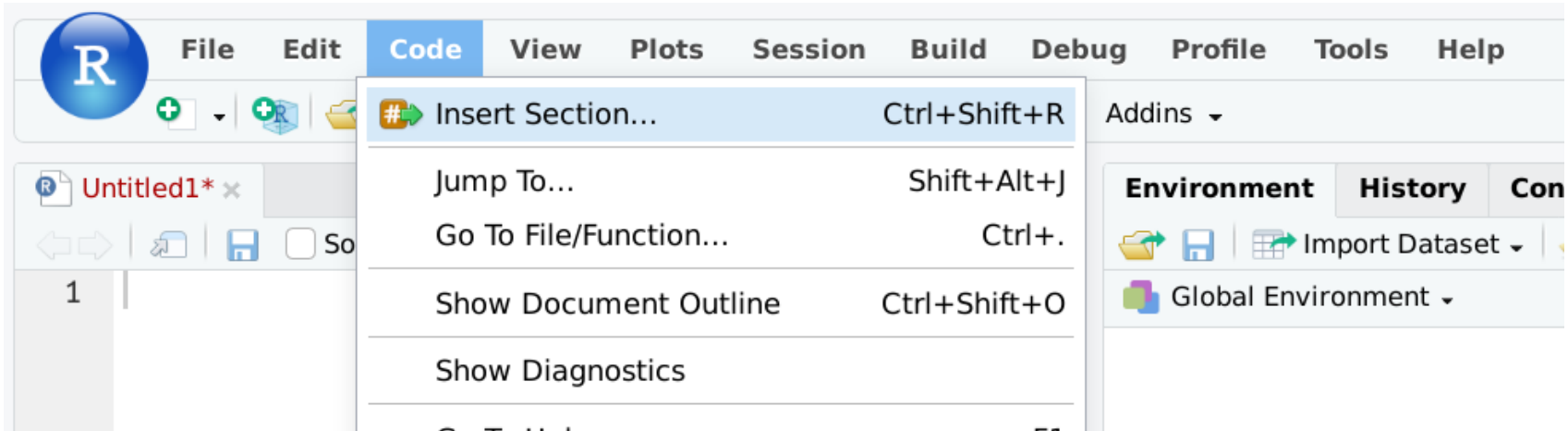
You can organise your scripts with sections and headers.

```
# Section One ----------------------------

# Section Two ============================

### Section Three ##########################
```

These can be inserted in RStudio with Ctrl + ⇧ + R or Code > Insert Section...



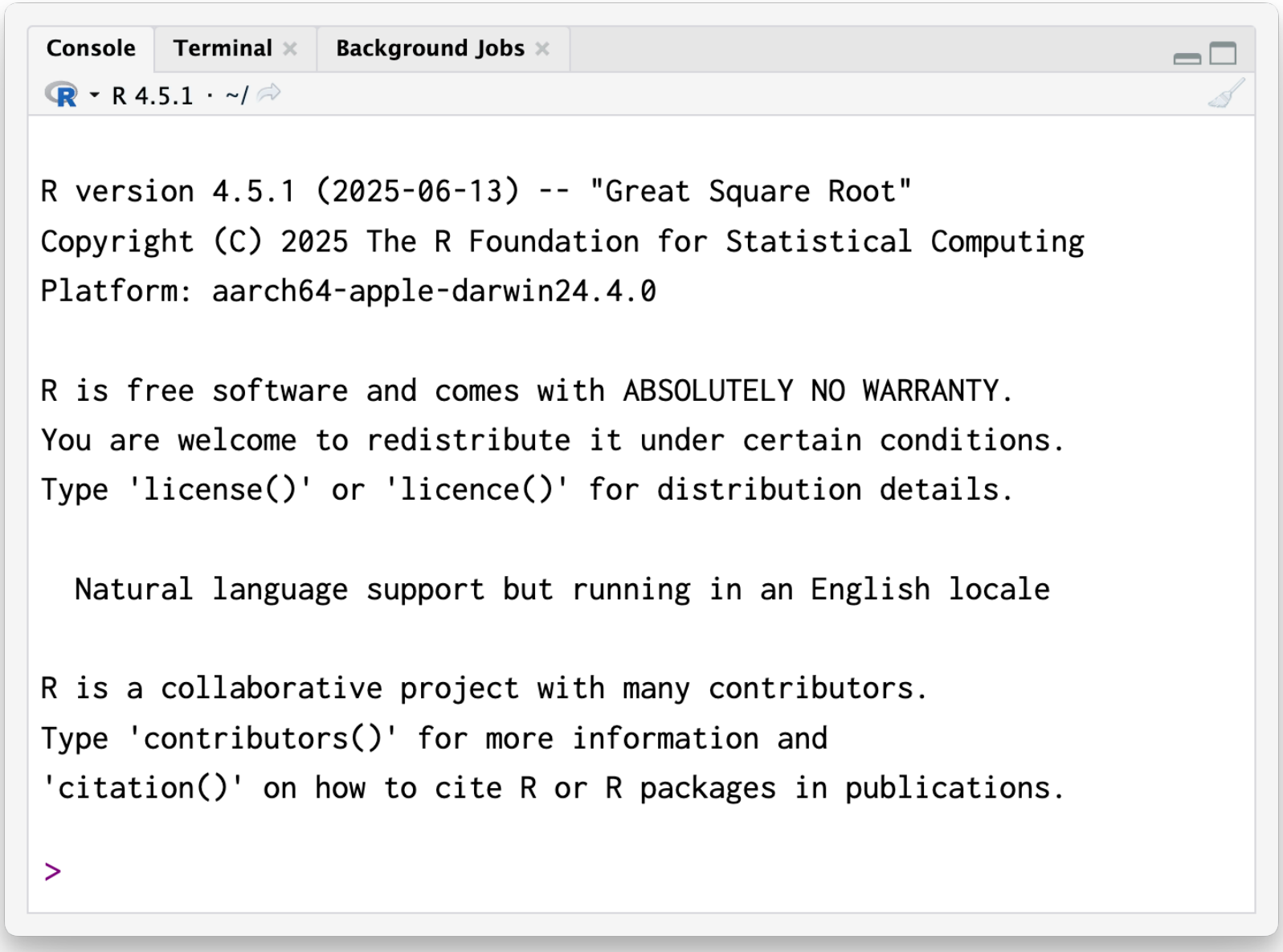
Combining several scripts

- Most analyses won't fit in a single script/file.
  - ⇝ You might have separate scripts for data import, cleaning, analysis, plotting…

- We can use the `source` command to execute code contained in another file.

```
source("1-Import-raw-data.R")
source("2-Data-cleaning.R")
```

The R console



Tips for the R console

- Access previous commands with ↑ / ↓.
- Search previous commands with Ctrl + R.
- Clear the screen with Ctrl + L.

Operators

> *Symbols that tell the compiler to perform specific mathematical or logical manipulations.*

Arithmetic

`+`	addition
`-`	subtraction
`*`	multiplication
`/`	division
`^` or `**`	raising to a power

Relational

`x < y`	less than
`x > y`	greater than
`x <= y`	less than or equal to
`x == y`	equal
`x != y`	not equal

For more, see `https://www.statmethods.net/management/operators.html`

Getting help

You can get help for any function with `?`.

```
?rnorm   # Help for the 'rnorm' function
?install.packages
```

All help files have a consistent structure:

Description → *Usage* → *Arguments* → *Details* → *See Also* → *Examples*

- I find the 'Arguments' and 'Examples' sections the most useful.
- It's worth familiarising yourself with this structure.

Resources

1. ## A Succinct Introduction to R

   Steve Haroz

   `http://r-guide.steveharoz.com`

2. ## R for Data Science

   Hadley Wickham and Garrett Grolemund

   `https://r4ds.hadley.nz`

3. ## Advanced R

   Hadley Wickham

   `https://adv-r.hadley.nz`

Should I use generative AI to learn R?

Generative AI for Learning R

Good uses

- Quick help debugging
- Explain why code fails
- Speed up routine work

Be careful

- When it matters
- When you don't yet understand the problem
- Out-of-date or overconfident answers
- Data science has minimal boilerplate
- We need more thinking, not less