Foundations of Data Science for Biologists

Class Orientation and Introduction to R

BIO 724D

2024-AUG-27

Instructors: Greg Wray and Paul Magwene

Orientation to BIO 724D

What is BIO 724D?

A class that introduces how to work with biological data

- Practical skills for wrangling, processing, filtering, and visualizing data
- Best practices for analysis, interpretation, reproducibility, and reusability of data
- *Assumes no prior programming experience*

A class that builds community

- Working with some data sets generated by Biology grad students
- Discussions with trainees, staff, and faculty in Biology and Duke more broadly

Not

- An introduction to statistics: though we will often discuss statistical methods
- An introduction to computer science: though we will introduce some concepts

Course structure

Lecture / lab: Tuesdays 3:05-5:35pm, 154 BioSci

Semi-flipped format: complete reading / video assignments *before* class

Class sessions a mix of traditional lecture and (mostly) hands-on engagement

Hands-on component involves individual follow-along as well as group exercises

Data lunch: Thursdays 12:00-1:00pm, 144 BioSci

Presentations by trainees, staff, and faculty over lunch

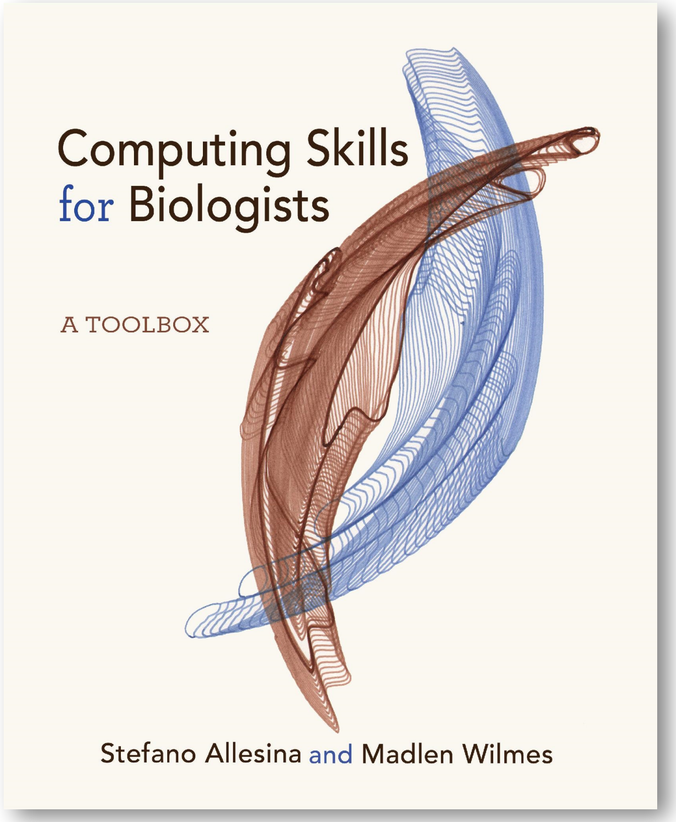
Discussions / questions / interaction encouraged

be respectful

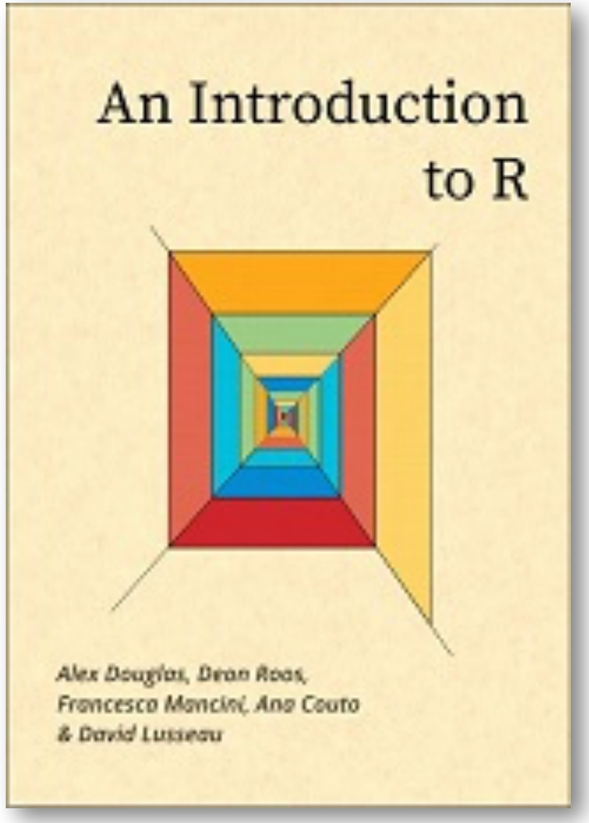
be polite

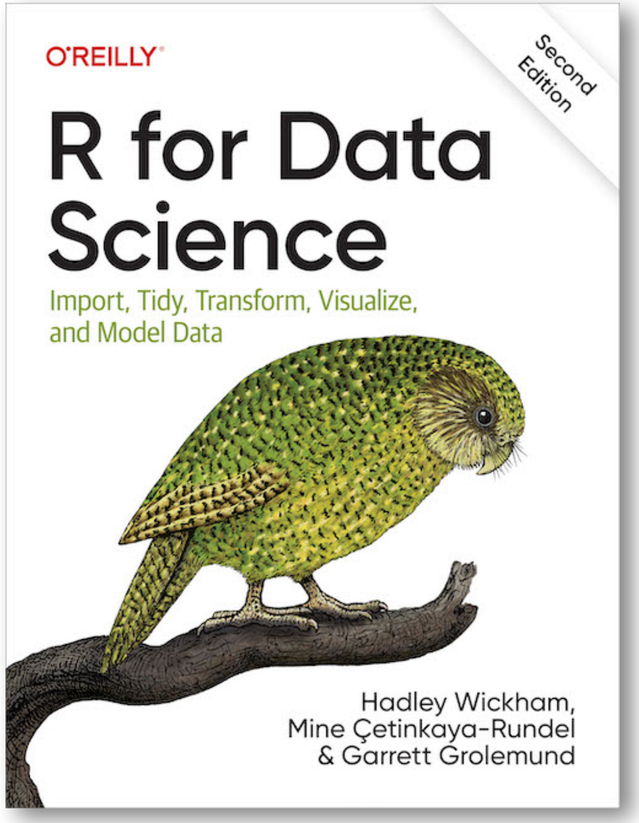
Course materials

Throughout the year:



For the R portion of the course:





*All are available electronically for **free**: see the course wiki*

Practicalities

Wiki: *the hub for most things related to the course*

- Schedule of lecture and lunch topics
- Information about course logistics, grading, and policies
- Materials for Tuesday sessions: readings, videos, data sets, notebooks, slides
- Assignments: problem sets and projects
- Links to resources: help with learning, reference material, deeper dives

GitHub: *where you turn in assignments*

- We will show you how to upload files
- Note the file naming convention in assignment instructions

Expectations

Adhere to the Duke Community Standard: collaboration and web resources are okay

Attend every class session and engage: follow along by actively writing your own code

Come to Tuesday sessions prepared: complete reading / video assignments *before* class

Submit homework and projects on time: late submissions will be scored lower

Stay on top of the material and ask questions: you will learn more if you engage and ask!!!

Familiarize yourself with the information on the course wiki

Assignments

Problem sets + project (90%)

- Problem sets posted on Tuesday; due the following Monday at midnight
- Graded for completion: code must run or include an explanation
- Project posted on 12 Nov, due 3 Dec at midnight (worth 2 problem sets)

Data lunch reflections (5%)

- Short description of new concept/method/application you learned
- Turn in with your problem set; graded for completion

Learning notebook (5%)

- File containing notes, code snippets, links, etc.; due 3 Dec at midnight

Resources for problem-solving

Be proactive: ask questions right away in class

- Don’t be shy: there are no “dumb” questions!
- Your classmates will thank you!

Many sources can help you when you are stuck with a homework problem

- Textbooks, Google, YouTube, LinkedIn Learning (free!)
- Built-in help resources, R vignettes, code snippets from official documentation
- Classmates (class Slack channel), friends, lab-mates, other grad students
- On-line forums: StackOverflow and others
- AI resources: ChatGPT, BingGPT, MS Copilot, etc.

Cite your sources and explain how they helped you

Code that isn’t working

There may be occasions when you can’t get your code to work properly: no problem!!

You will still get full credit if:

- You turn your homework in on time
- Identify specifically what is and is not working correctly
- Explain how you tried to fix the problem (be concise)

You will not get full credit if:

- You turn your assignment in late without prior arrangement
- You neglect to reveal and explain what isn’t working properly (see above)

Test your code carefully and be transparent about any issues you can’t fix!

What can you expect from BIO 724D?

You will learn concepts and skills that will:

- Be useful throughout your research career
- Save you lots of time and frustration
- Reduce errors in your work
- Make it easier for you to collaborate

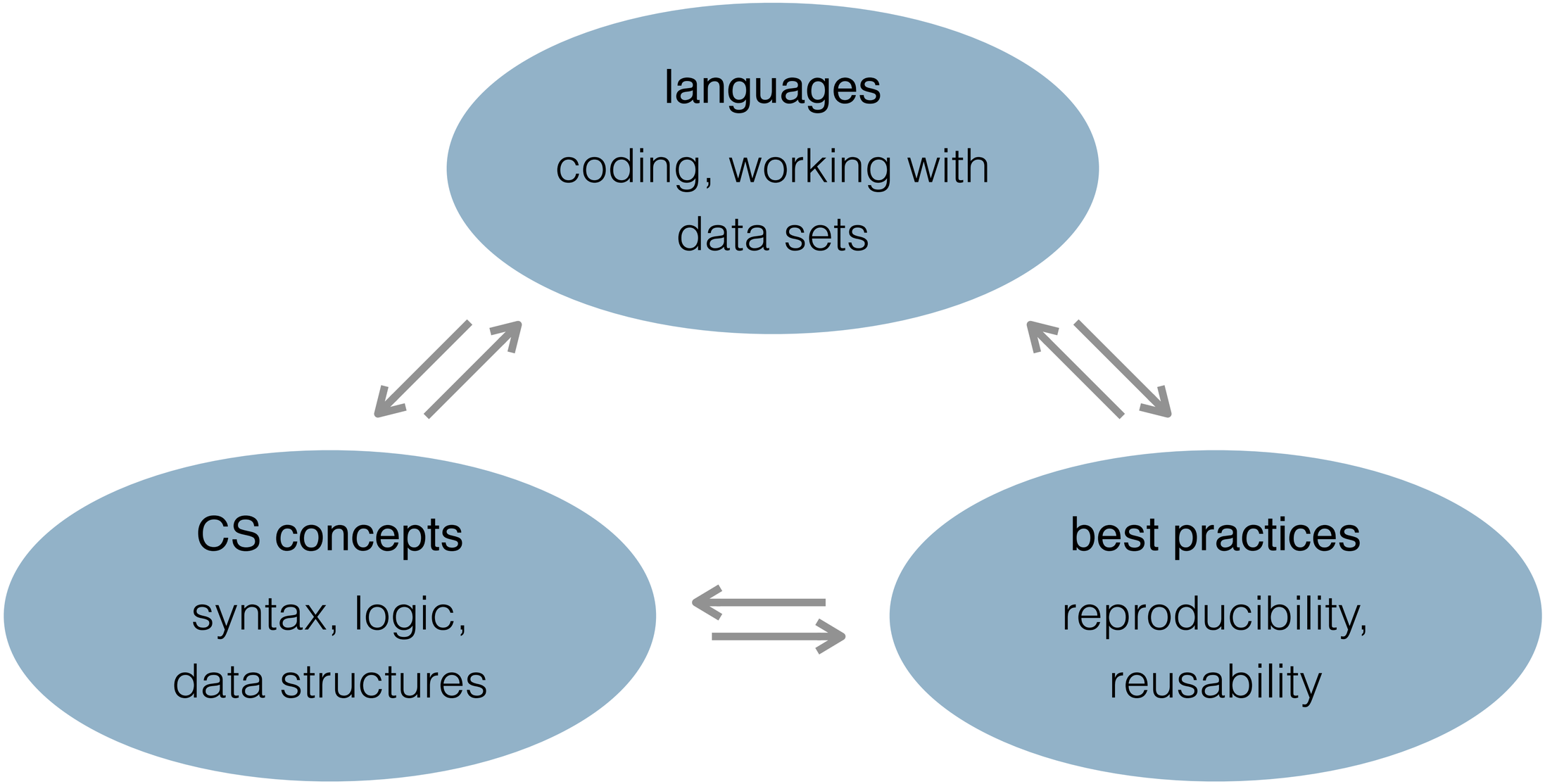
This comes at a cost

- You will need to practice, practice, practice
- You may feel frustrated at times, but persevere and *it will be worth it!*

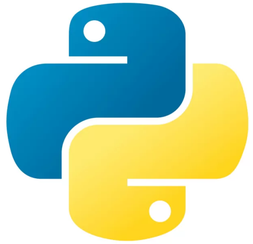
Your instructors will strive to make this process as smooth as possible

- Maintain a safe, respectful, and constructive learning environment
- Encourage questions and foster curiosity (there are no “dumb” questions)
- Provide support and resources to help you thrive and learn
- 2-semester format is intended to allow you to absorb information and reduce stress

Learning objectives



Languages we will work with

	strengths	*weaknesses*
	Designed for statistics and visualization Good for numerical modeling Widely used in biological research	Awkward for tasks outside its niche Size of data sets limited by RAM Slow, limited graphics capabilities
	Designed for relational databases Extraordinarily powerful in this role Dominant database environment	Nearly useless outside its niche Procedural programming is awkward Not as flexible as some other QLs
UNIX	Nearly universal computing environment Extremely powerful, uniquely capable The most future-proof way to code	Terse syntax, not easily readable Does not provide much feedback Designed to work primarily with text
	General-purpose high-level language Readable code, consistent syntax Among most widely used languages	Not specialized: 2nd best at everything Slower than some other GPLs

Learning objectives: the fine print

How to work with data:

- Format data so that it is easy to work with
- Reproducibly identify missing data, improperly formatted data, and outliers

How to write code:

- Works as intended
- Makes sense to future-you and others
- Saves time and effort, reduces errors, and improves reproducibility
- Can be re-purposed for other tasks, even unrelated ones

How to produce visuals:

- Explore and understand complex data sets with ease
- Portray your results to others with integrity, clarity, and accuracy
- Customized to your precise specifications

Preview of topics

session	topics
1 - 10	**R**: importing, transforming, querying, and displaying data; Quarto
11 - 12	**SQL**: designing and querying relational databases
13	**R**: factors, date/time, geospatial
	winter break
14 - 19	**UNIX**: working with the command line and with text and tabular files
20 - 27	**Python**: programming; working with data in Python; Jupyter

Introduction to programming with R

Why learn about R?

Designed specifically for data analysis, modeling, and visualization

Designed by statisticians for statisticians: rigorous, tested, and respected

Free, open-source, non-proprietary, cross-platform, actively maintained

Encourages best practices in reproducibility, visualization, and sharing

Huge set of tested libraries for specialized tasks

Used by biologists in many areas of study, familiar to many collaborators

Numerous libraries specific for biological research (e.g., GIS, modeling, genomics)

What is the difference between R and RStudio?

programming language

does the computing

“the engine”

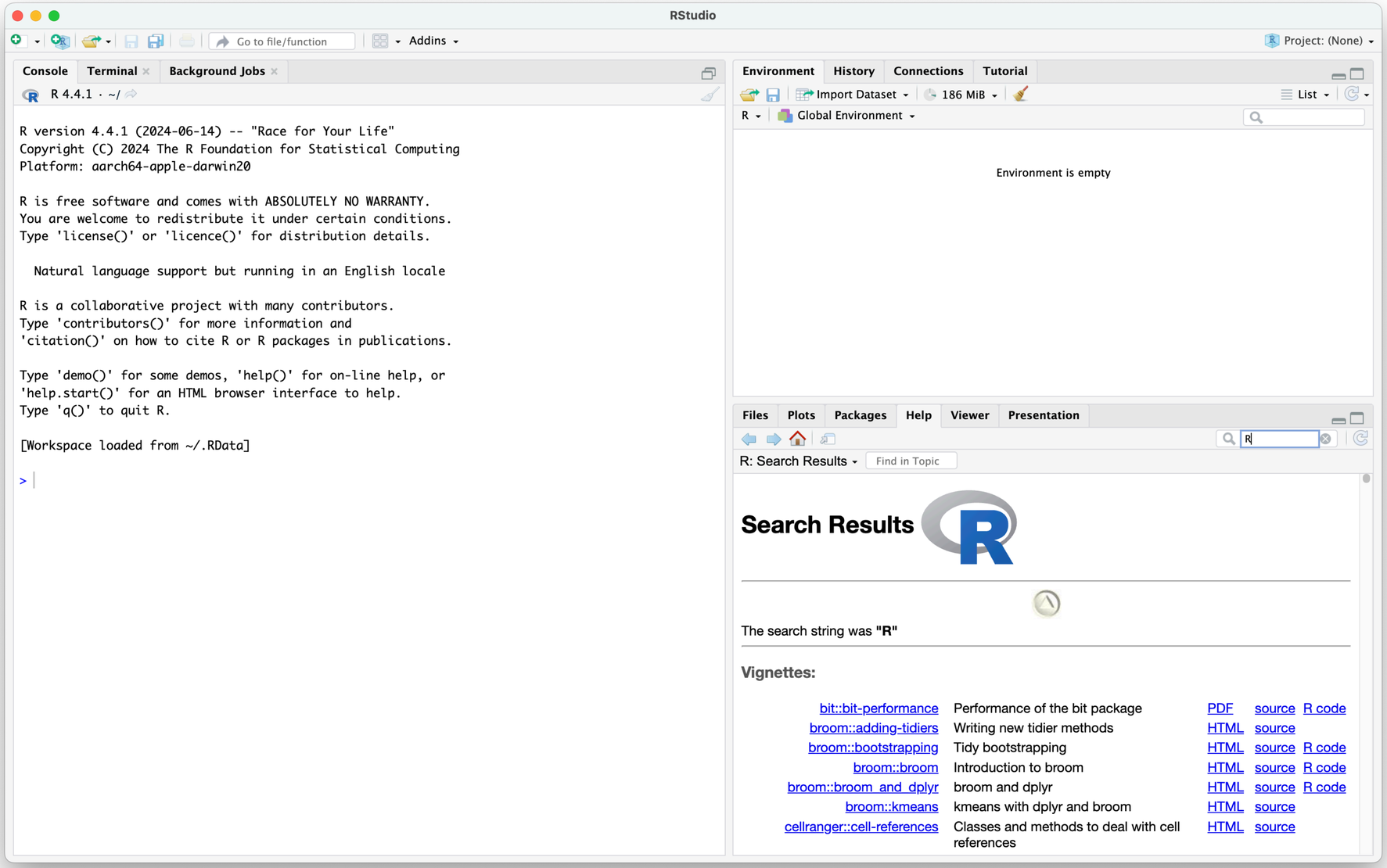
programming environment

provides an interface to R

“the dashboard”

The RStudio interface

console →



Typing at the console

If you enter an expression at the console, R will return the result:

```
> 6 * 7                                  # multiplication
[1] 42
> sqrt(9)                                # using a function to find the square root
[1] 3
```

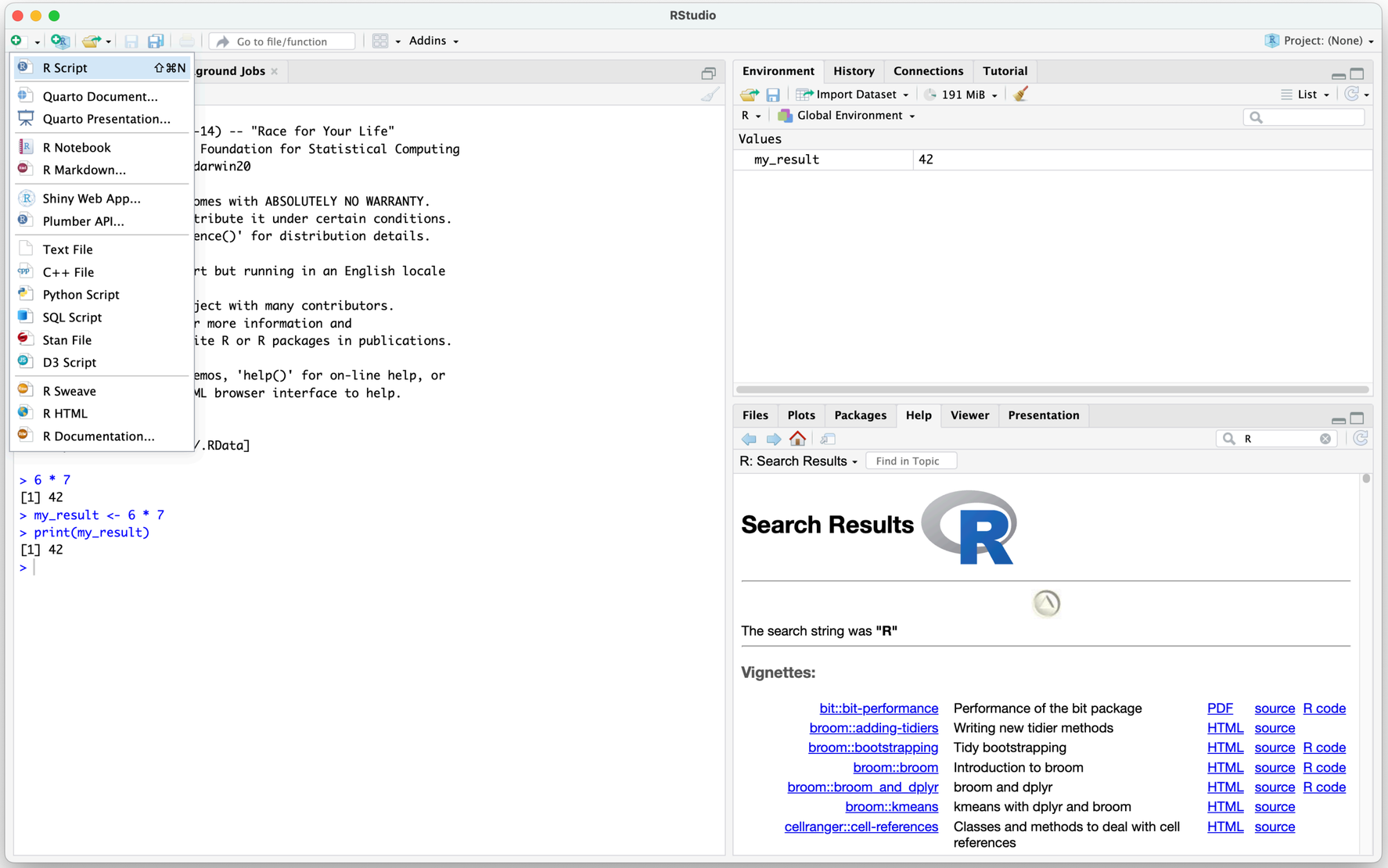
To store a result, assign it to a variable name:

```
> my_result <- 6 * 7                     # <- stores the result of an expression
> print(my_result)                       # displays the value stored in my_result
[1] 42
```

When you quit RStudio, any stored values will be lost! The solution is to store code.

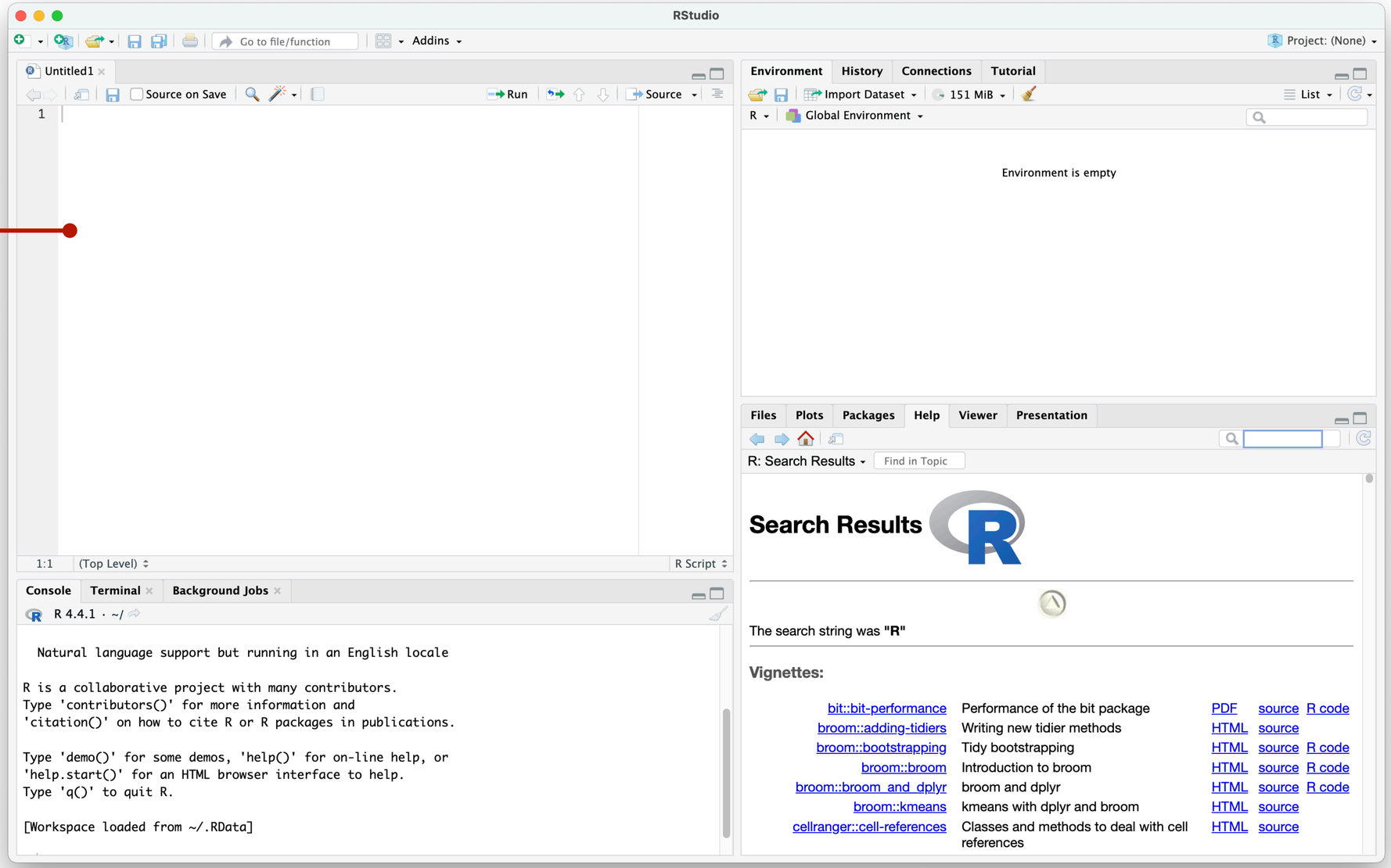
Creating a script

click on the left-most button and select **R Script**



The RStudio interface

type code here and save for later use



Store code in your script

Let's write some code:

```
# my script to learn about R and RStudio


my_result <- 6 * 7
print(my_result)


my_message <- 'hello world'
print(my_message)
```

Save the file to preserve your work — you can retrieve it later by opening the script

Scripts are very useful for: saving time, record keeping, reproducibility, reusability

Some tips for writing scripts

Use comments at the beginning of the script to indicate overall purpose, date, author, etc.

Use comments throughout to explain what your code does

Use descriptive names for variables

Use whitespace for readability (blank lines and extra spaces are generally ignored)

Save time!

Use `tab` to autocomplete names of functions and variables

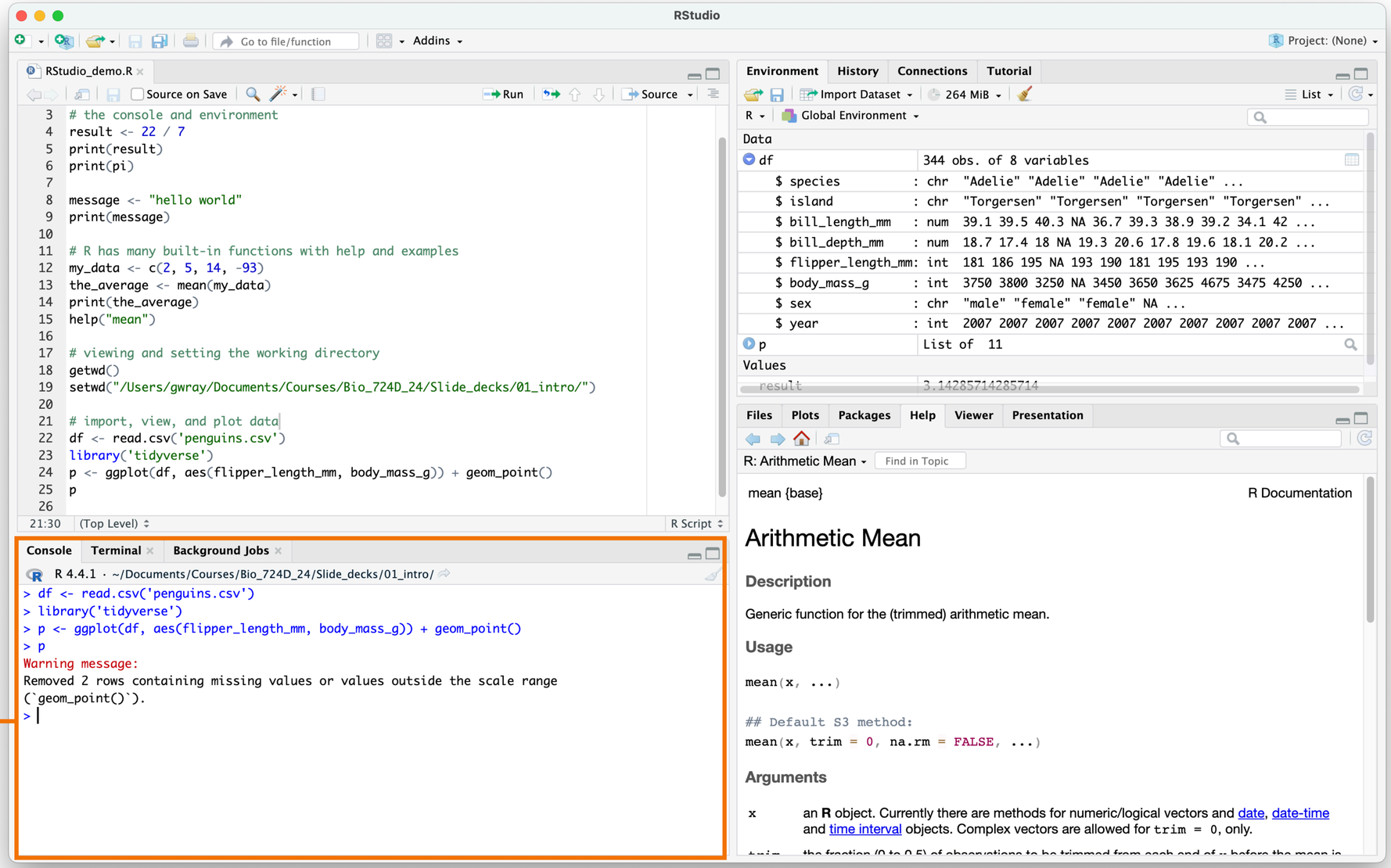
Use arrow keys to select the desired value when there are multiple options

Use keyboard shortcuts:

<- `opt` + `-` on Mac, `alt` + `-` on Windows (note: minus key)

Full list under Help menu, Keyboard Shortcuts Help

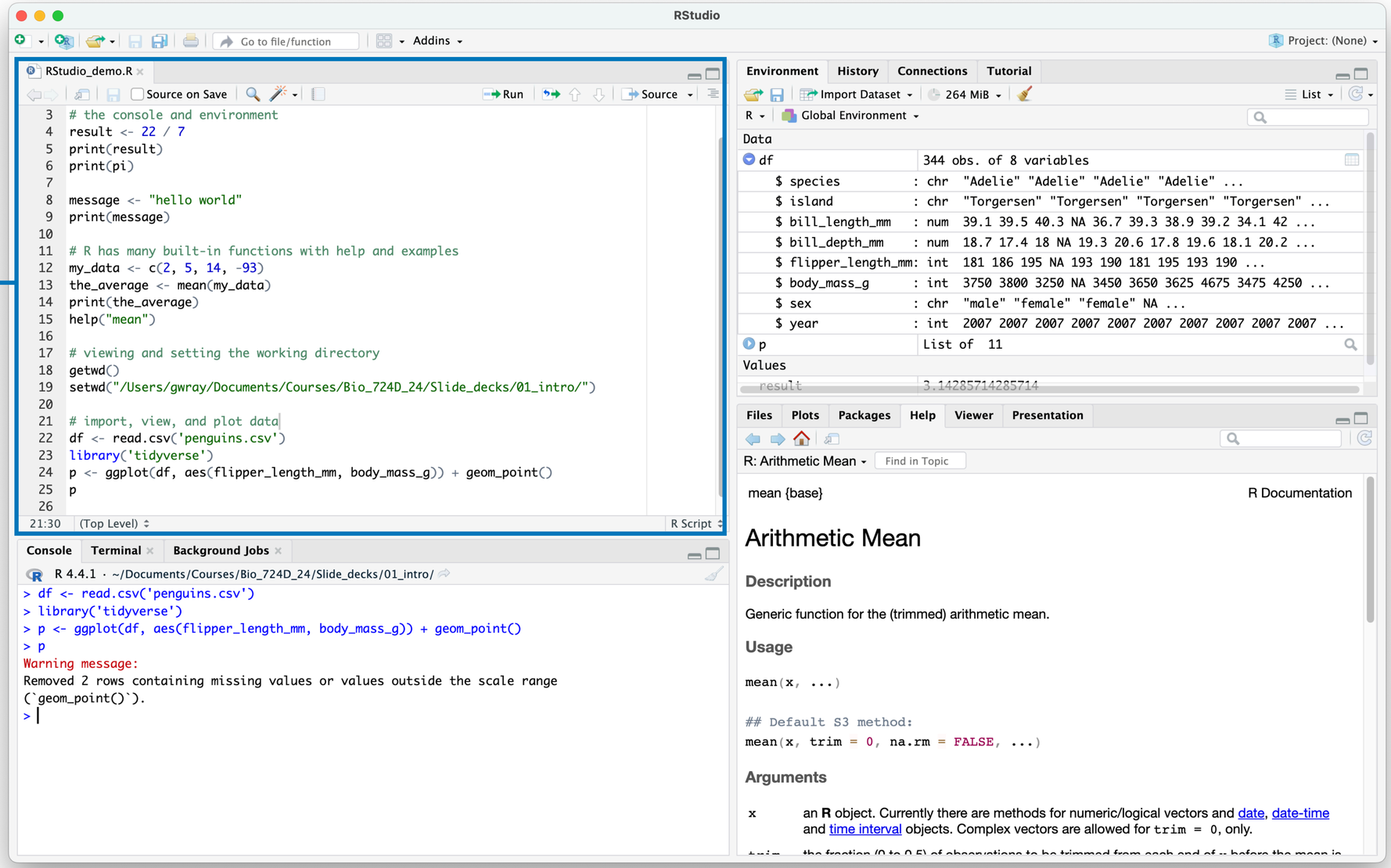
The RStudio interface



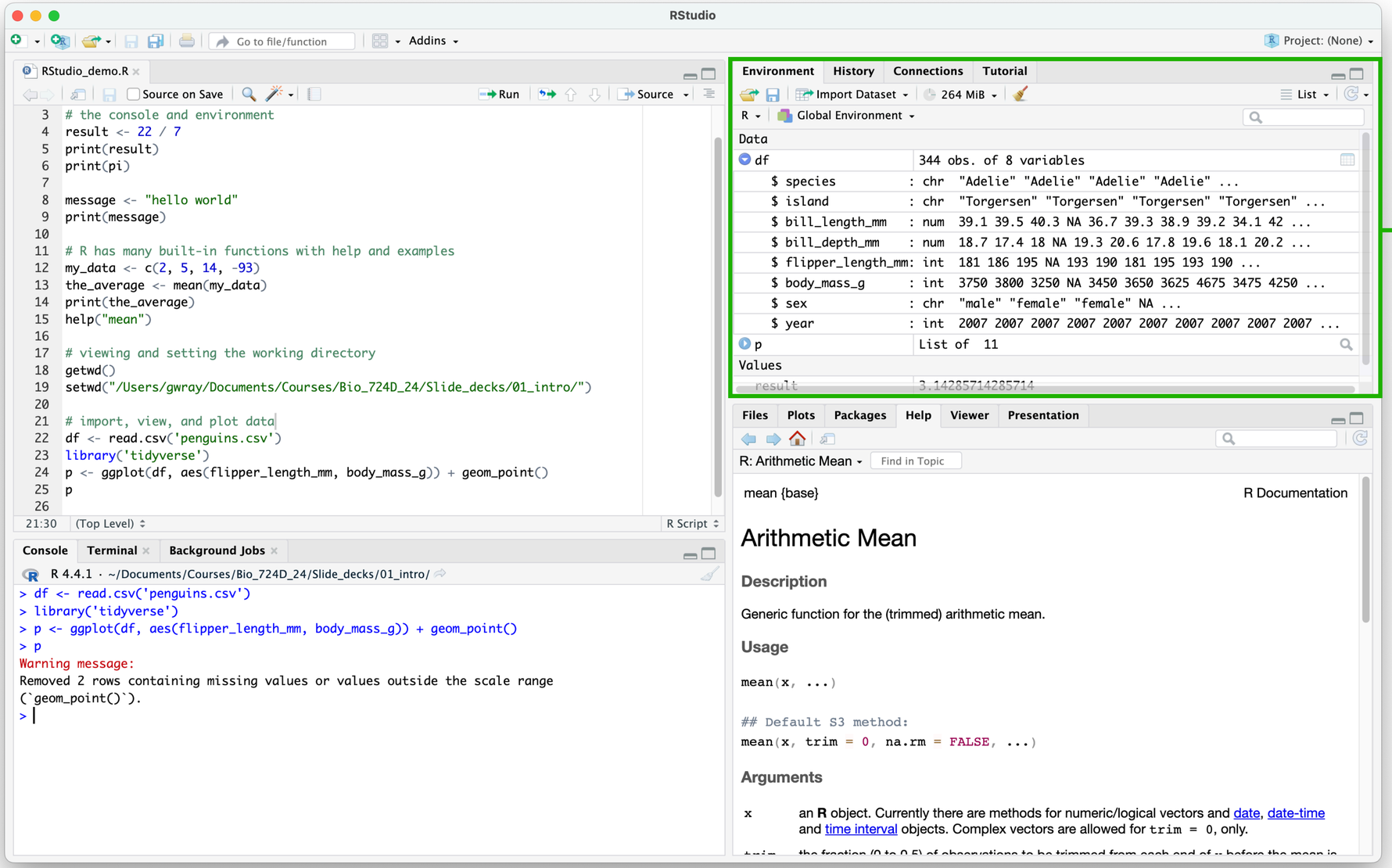
console

The RStudio interface

program

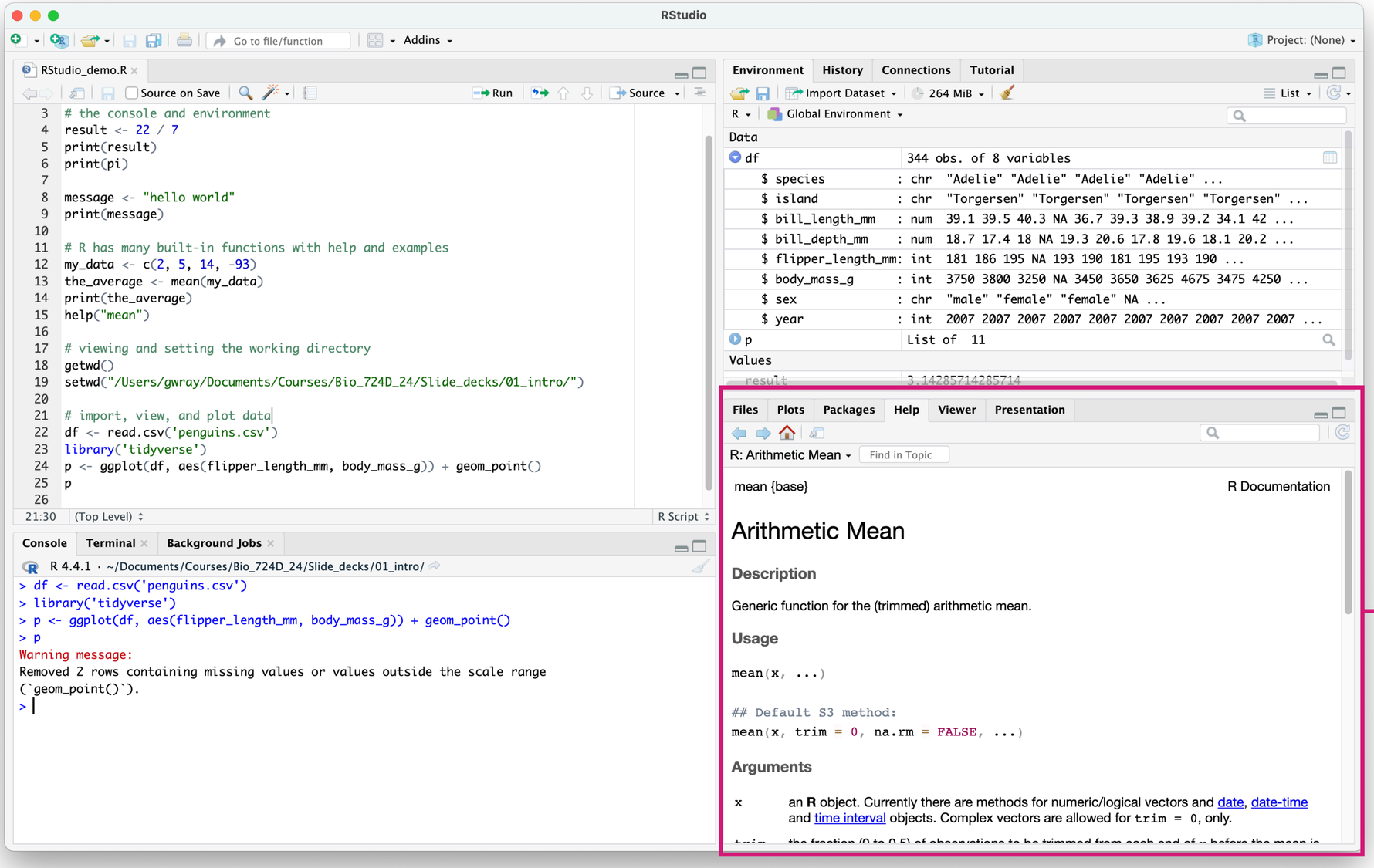


The RStudio interface

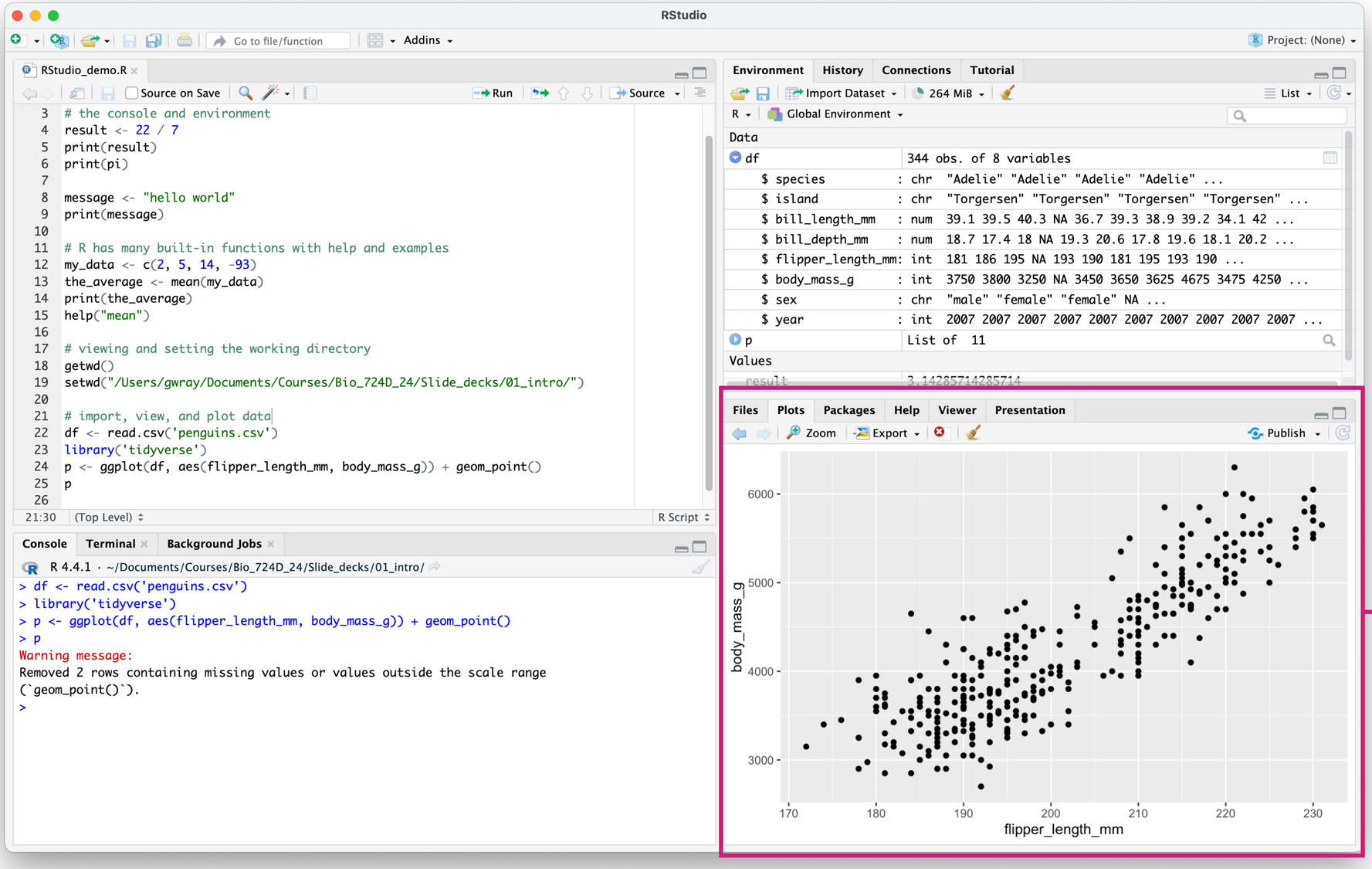


environment

The RStudio interface



The RStudio interface



plots