Foundations of Data Science for Biologists

Class Orientation and Introduction to R

BIO 724D

25-AUG-2025

Instructors: Greg Wray, Paul Magwene, and Kayla Wilhoit

Orientation to BIO724D

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

- Drawing on data sets generated by current and former biology grad students
- Fostering discussions with current biology trainees, staff, and faculty

Not

- An introduction to statistics (though we will often use statistical methods)
- An introduction to computer science (though it might feel like it occasionally)

Course structure

Lecture / lab: Mondays 3:05-5:05pm, 4233 FFSC

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

Class sessions mostly active learning with some traditional lecture

Hands-on component a mix of individual follow-along and group exercises

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

Presentations by trainees, staff, and faculty over lunch

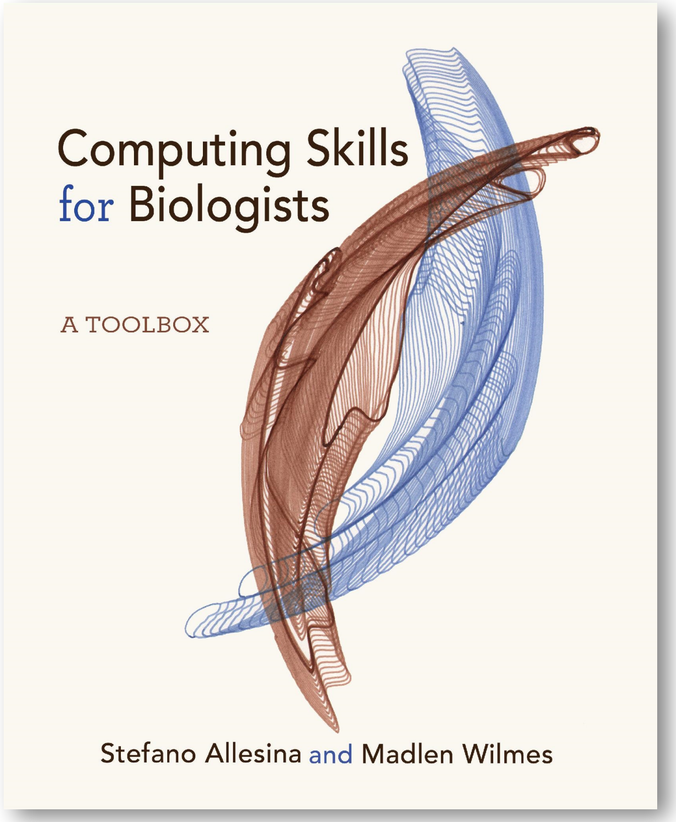
Discussions / questions / interaction encouraged

be respectful

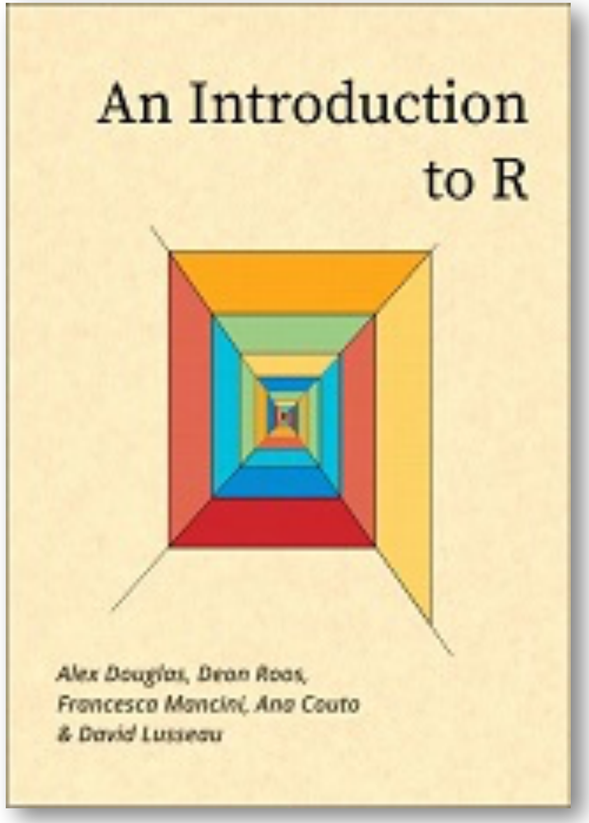
be polite

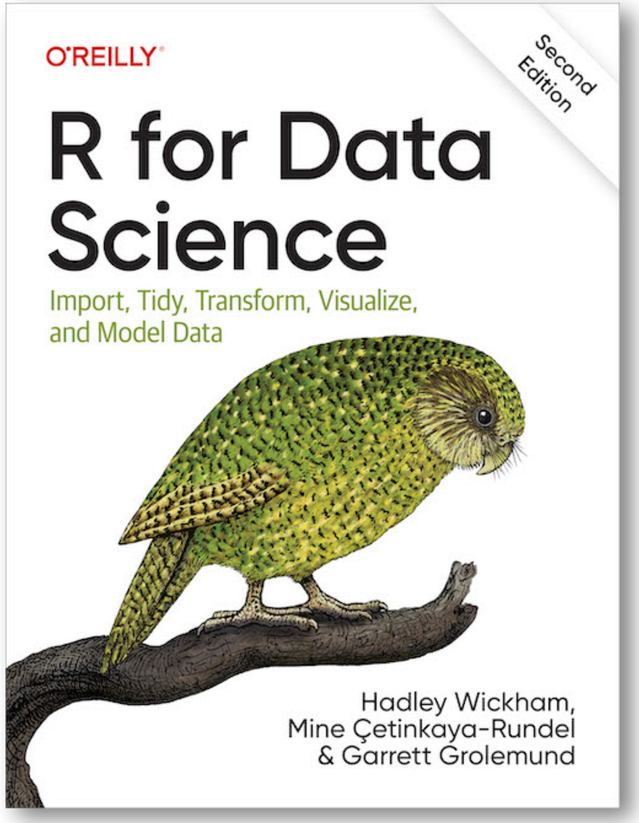
Course materials

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 lab/lecture and lunch topics
- Information about course logistics, grading, and policies
- Links to preparation materials: readings and videos
- Links to slide decks
- Assignments: weekly problem sets and project rubric
- Links to resources: help with learning, reference material, deeper dives

GitHub: *where you turn in assignments*

- Upload files into your folder under the “Assignments” tab
- Note the file naming convention in assignment instructions
- We will cover how to set up your GitHub repository on Thursday

Expectations and keys to success

Adhere to the Duke Community Standard

Attend every class session and actively engage

Come to Monday sessions prepared

Submit homework and projects on time

Stay on top of the material and ask questions

Familiarize yourself with the information on the course wiki

Assignments

Problem sets (90%)

- Posted by class time each Monday; due the following Sunday at midnight
- Graded for completion: code must run or include an explanation

Data lunch reflections (5%)

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

Learning notebook (5%)

- Markdown file of notes, code snippets, links, etc.; due at end of the semester

Projects

- Posted on 13 Nov, due 1 Dec
- Graded for completion; counts as 2 problem sets

Seeking help

Seek help when you are stuck!

- Textbooks, Google, YouTube
- On-line forums: StackOverflow and others
- AI resources: ChatGPT, Claude, BingGPT, GPT4/5, etc.
- Built-in help resources, R vignettes, code snippets from official documentation
- Classmates (class Slack channel), friends, lab-mates, other grad students

When turning in your homework, cite your sources and explain how they helped you

- E.g., Code for removing empty rows modified from a post on StackOverflow
- E.g., Use of Itertools library for permutation suggested by a lab mate
- For humans: use generic attributions rather than personal names
- For AI tools: name the specific model and include the prompt you used

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 what you tried to do to fix the problems

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

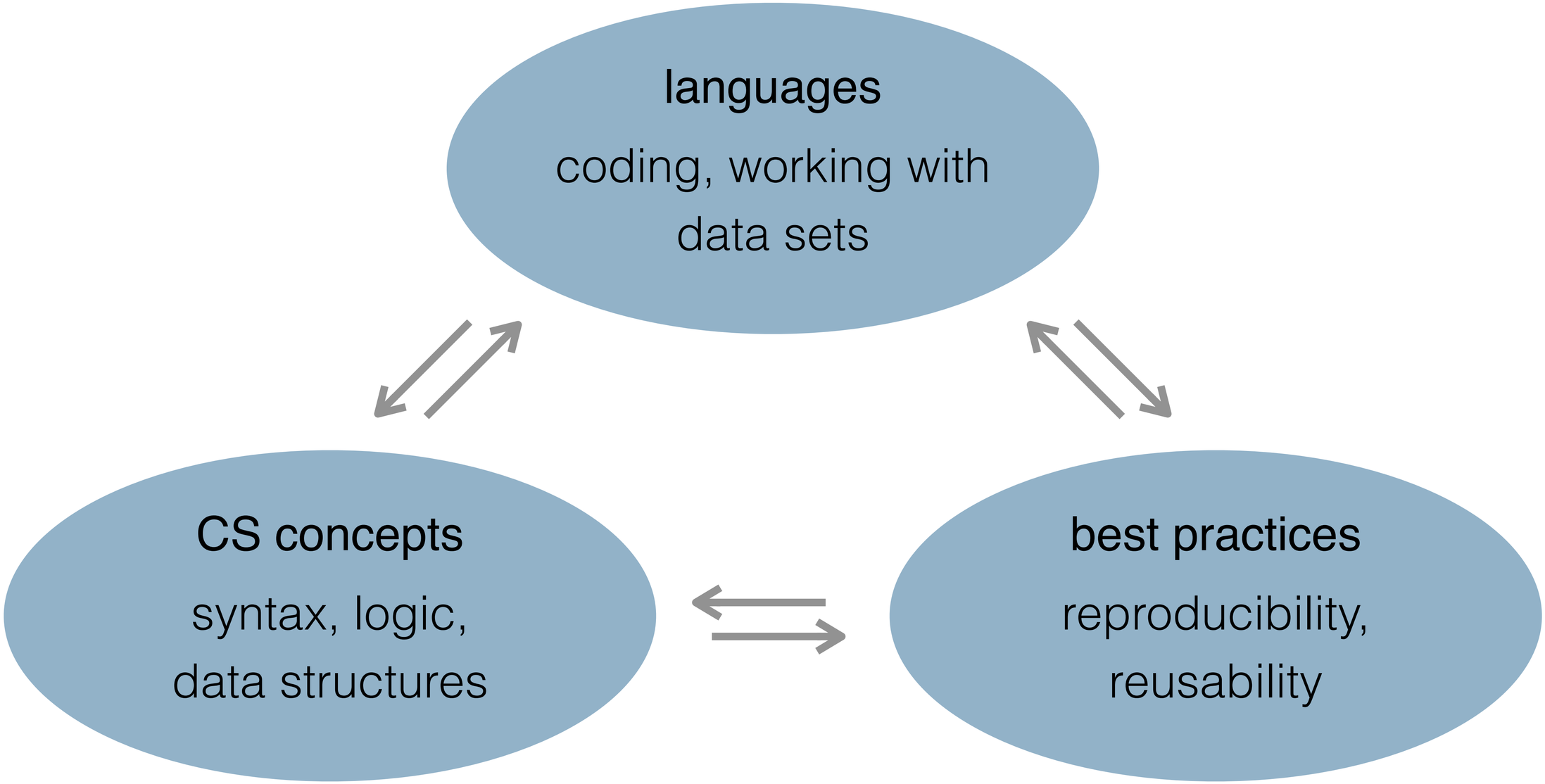
This comes at a cost

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

We will strive to make this process as smooth as possible

- Maintain a safe, respectful, and constructive learning environment
- Encourage questions and foster curiosity (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



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 that:

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

How to produce visuals that:

- Reveal trends, organization, and anomalies within complex data sets
- Portrays your data clarity, accuracy, and integrity
- Is customized to your precise specifications

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 Library ecosystem can be confusing
	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 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	“Second-best at everything” Library ecosystem can be confusing

Preview of topics

session	topics
1 - 11	R: importing, transforming, querying, and displaying data
12 - 13	SQL: designing and querying relational databases
	winter break
14 - 19	Unix: working with files at the command line; regular expressions
20 - 26	Python: general-purpose programming

Introduction to R and R Studio

Why learn 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

"engine"

programming environment

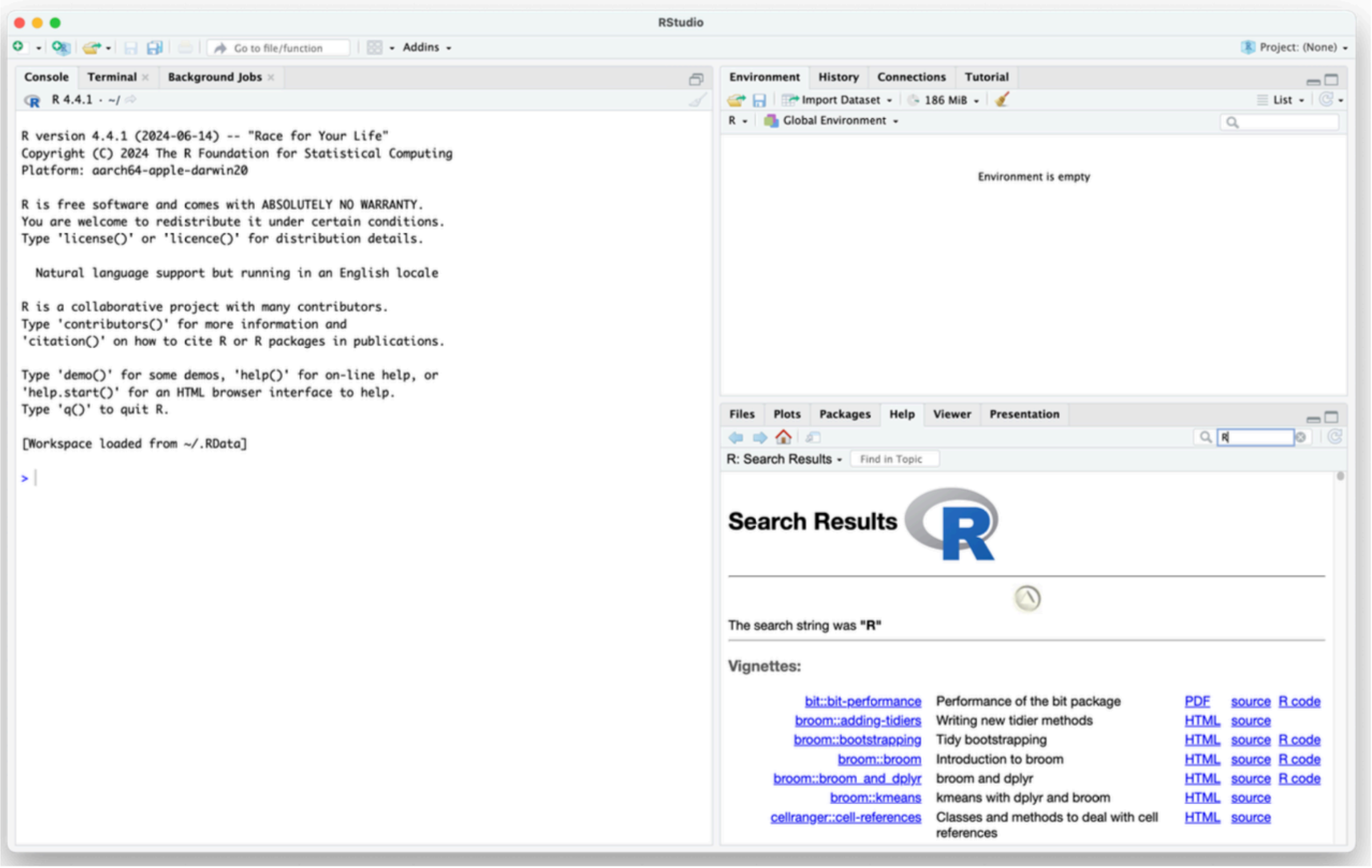
provides an interface

"dashboard"

RStudio

R version →

prompt →



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
[2] 3
```

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

```
> my_result <- 6 * 7                     # <- is the assignment operator
> print(my_result)                       # displays the value stored in my_result
[3] 42
```

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

Store your code in a script

Let's write some code we want to save:

```
# 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 file

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

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!

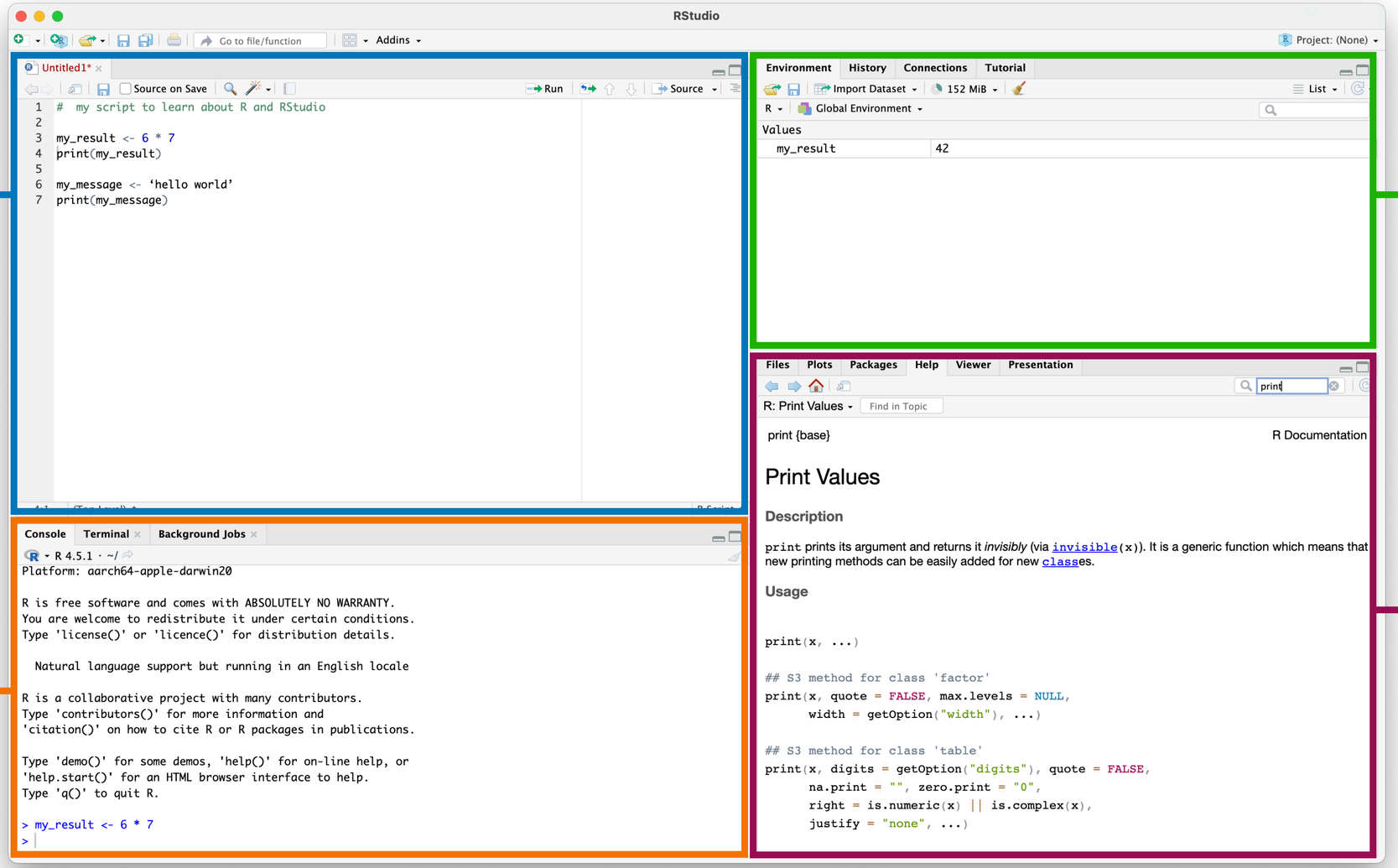
Use `tab` to autocomplete names of functions and variables

- 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

Quadrants of the RStudio interface



code editor

environment

console

help, plots, file browser