

# R Programming For Natural Resource Professionals

Lecture 1: Introduction to R, basic math, objects

Jared Homola  
TNR 163  
jhomola@uwsp.edu



University of Wisconsin  
**Stevens Point**





**Dan Isermann**  
**Unit Leader**

**Aquatic  
Biomonitoring Lab**

**Fisheries Analysis  
Center**



**Jared Homola**  
**Assistant Unit Leader**

**Molecular Conservation  
Genetics Lab**

# Course objectives

## Develop general data science skills

- Ethical data management practices
- Research reproducibility
- Master efficient workflows

## Develop skills in the R programming language

- Read in and process data
- Write functions, loops, conditionals, etc.
- Create wide variety of data visualizations
- Code and interpret common statistical analyses

# What the course is not

## A stats course

- Learn how to do some stats in R
- No evaluations on statistical rigor

## A memorization-based course

- Learning to become self-sufficient
- Approach assignment as you would your research

# Course organization

- Tues 11:00am – 12:15pm lecture
- Thurs 11:00am – 12:15pm lab
- Homework assigned each Thurs, due the next Thurs
- Two larger assignments
- Grades: 100% homework

# Course communication

- Office hours:
  - By appointment
  - By email: [jhomola@uwsp.edu](mailto:jhomola@uwsp.edu)
  - By Teams Chat
- Course website: [jaredhomola.github.io/RforNatRes/](https://jaredhomola.github.io/RforNatRes/)

# Class discussions

- Tuesdays will start with a discussion of assigned readings
- I'll establish discuss groups when you get here each week
- 10 min to discuss and formulate three things:
  - Thought
  - Question
  - Epiphany
- Submit each via Google Docs which will serve as starting point for discussion



# Class norms

- Be respectful & understanding of wide diversity of experiences.
- Ask questions during class. Share your learning.
- Avoid distracting others with what's on your screen.
- You're in a professional program. You'll be treated like a professional.
- Feedback on the course is always welcome.

# Sharing and code reusing policy

- Team learning is key! Help each other.
- Turn in assignments independently, but list people you worked with
  - Do not turn in identical assignments
- Cite code that you found online by providing the url after relevant answers
- Reused code that is not cited will be treated as plagiarism

# AI-assisted coding

- Use of AI such as ChatGPT is new and we're all trying to understand it
- Let's learn together
- You can use it, but credit it
- **You get out what you put in**

# Tips for course success

- Getting the right answer is a minor part of what you'll be graded on.
  - Emphasis on creativity, problem solving, efficiency, and best practices
- There are countless ways to get to the right answer
  - I'll teach you a couple ways, you'll probably find others
  - Strive to write efficient code

# A word on TidyR

- Beginning in week 3, we will transition to writing code using the “TidyR” approach
- Learning some essential base R code in the first couple weeks will serve you well going forward

# What is R?

- R is...
  - computer language
  - an environment for statistical computing
  - platform for generating graphics
  - modeling platform
  - Much, much more!
- Script-based (text computer code) and not GUI-based (point and click with menus)



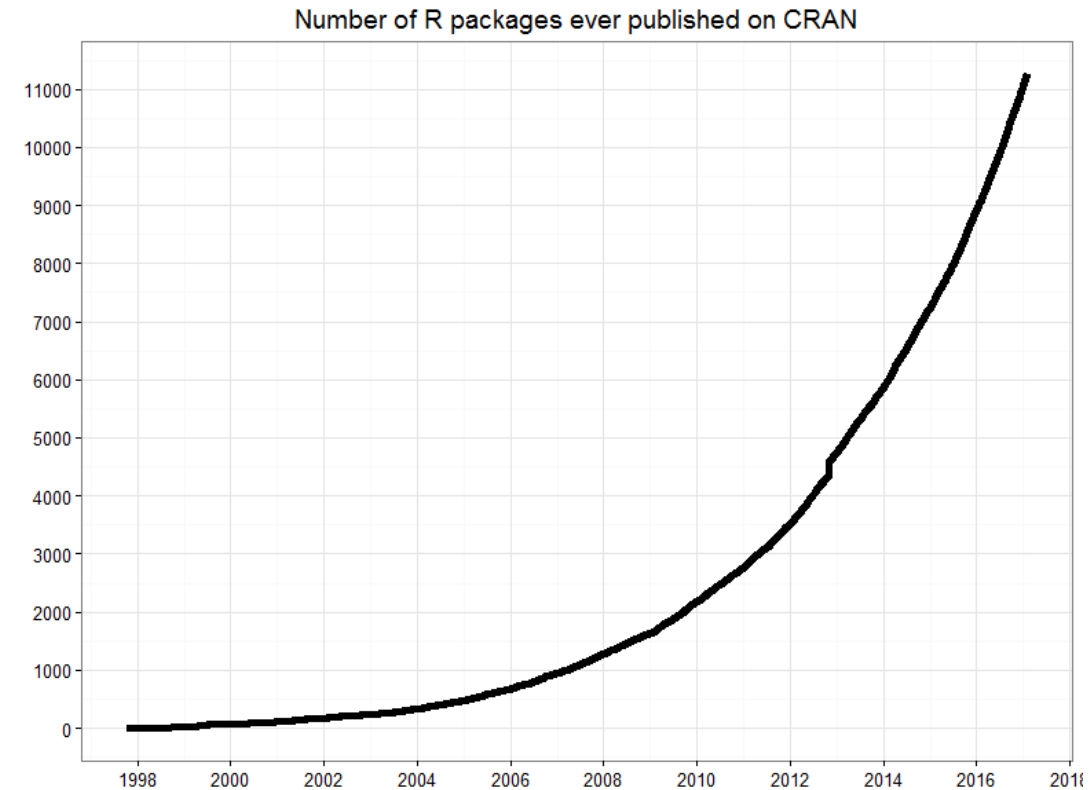
# Why learn R?

- It can get you a job
- Free and open-source
- Powerful plotting tool for generating publication-quality graphics
- Huge community of scientists and developers using R



# Why learn R?

- Incredibly interdisciplinary
- Currently 20,252 R packages on CRAN!
  - Population dynamics
  - Fish stock assessment
  - Telemetry
  - Camera trap analyses
  - Econometrics
  - Baseball analytics
  - Thousands of other things...





# Why learn R?

- Research **reproducibility** and **transparency**: Anyone using any operating system can reproduce your work
  - Read in data
  - Wrangle your data into the right format
  - Exploration of patterns in complex data
  - Apply statistical tests and fit models
  - Produce summary statistics and tables
  - Create final figures
- Easily make changes if your data change, model must be revised, or reviewers ask for revisions, etc.

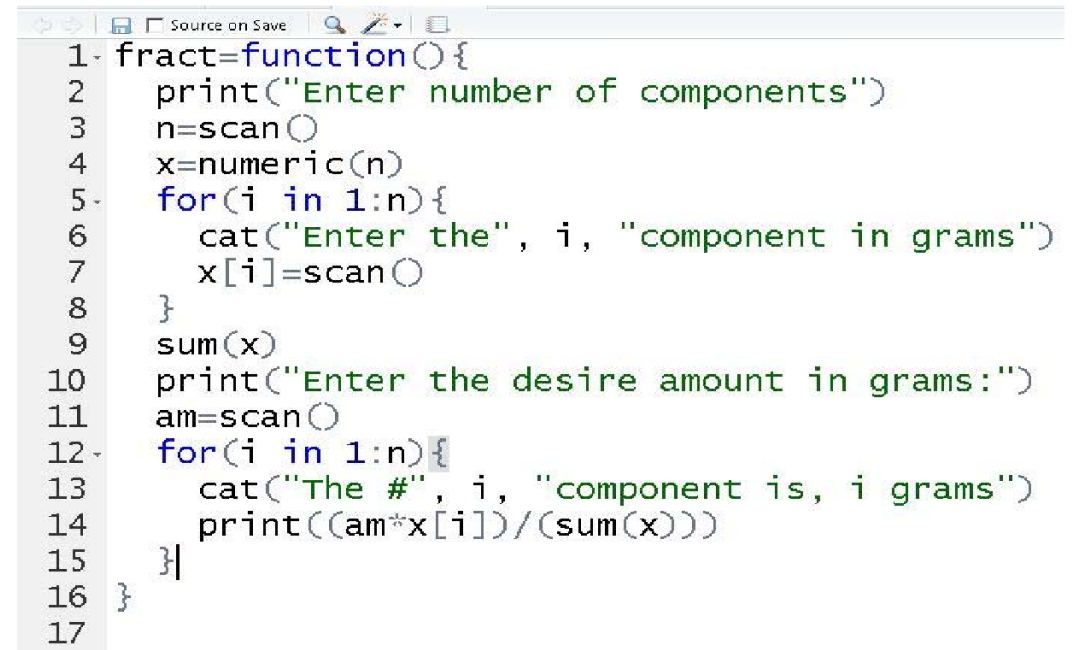
# What about Excel?

- Excel allows quick prototyping
- Data manipulation is easy
- Can see what is happening
- Looping is hard
- Limited statistical packages
- Inflexible
- Hard to repeat analyses



# How to learn R

- R is a language, the learning curve can be steep
- Be patient and creative
- Keep your motivation in mind
- Lots of help files, online sources, books
- Work with other students
- Reach out to instructor
  - Might just point you toward help

A screenshot of an R script editor window. The window has a title bar with standard icons and a 'Source on Save' checkbox. The script is a function named 'fract' that takes no arguments. It prompts the user to enter the number of components, then enters the component amounts in grams, then prompts for a desired amount, and finally prints the percentage of each component relative to the total. The code is as follows:

```
1- fract=function() {  
2-   print("Enter number of components")  
3-   n=scan()  
4-   x=numeric(n)  
5-   for(i in 1:n){  
6-     cat("Enter the", i, "component in grams")  
7-     x[i]=scan()  
8-   }  
9-   sum(x)  
10-  print("Enter the desire amount in grams:")  
11-  am=scan()  
12-  for(i in 1:n){  
13-    cat("The #", i, "component is, i grams")  
14-    print((am*x[i])/(sum(x)))  
15-  }  
16- }  
17-
```

# History of R

S

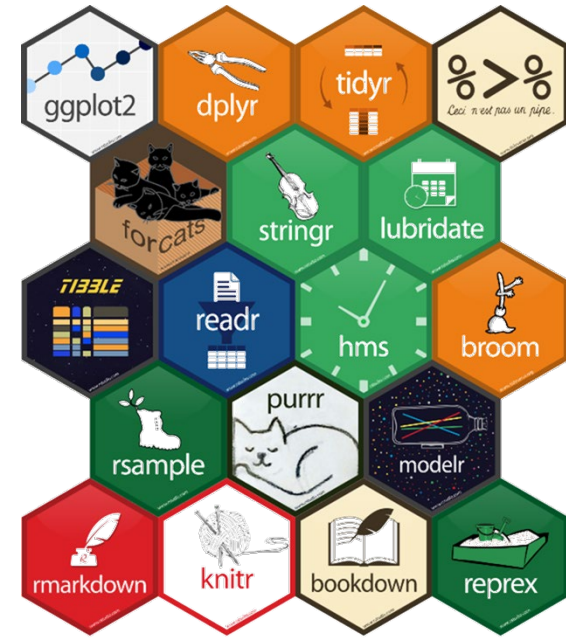
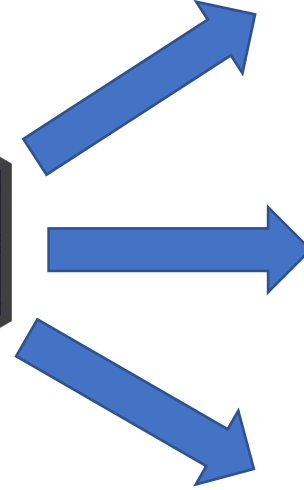
1976



1996

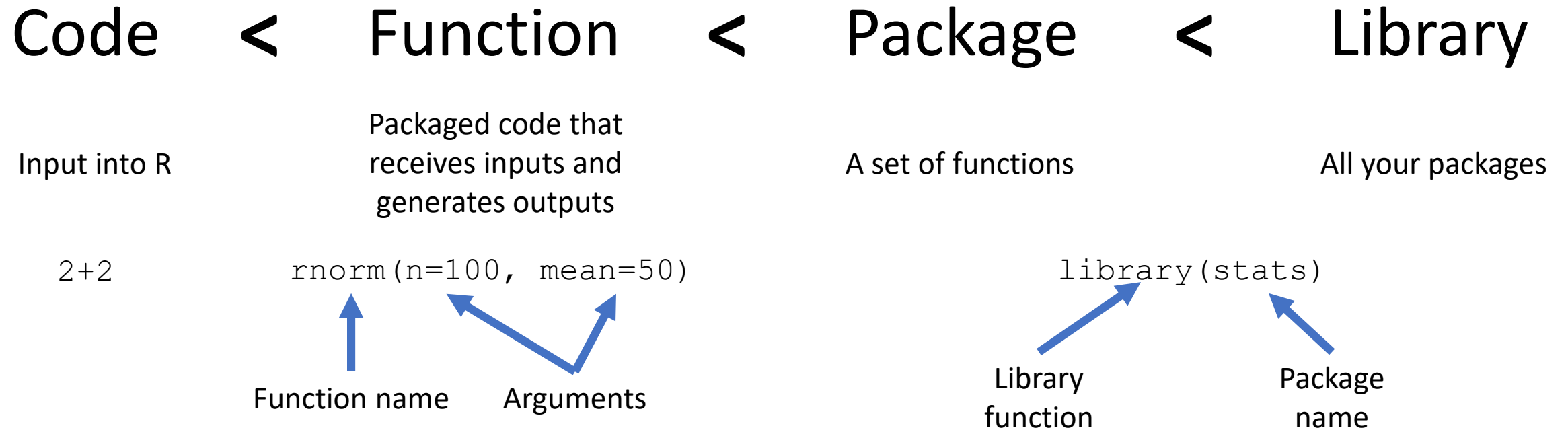


~2005



- S: language for data analysis developed at Bell Labs circa 1976
- R: initially written & released as an open-source software by Ross Ihaka and Robert Gentleman at U Auckland during 90s
- Since 1997: international R-core team ~15 people & 1000s of code writers and statisticians happy to share their libraries

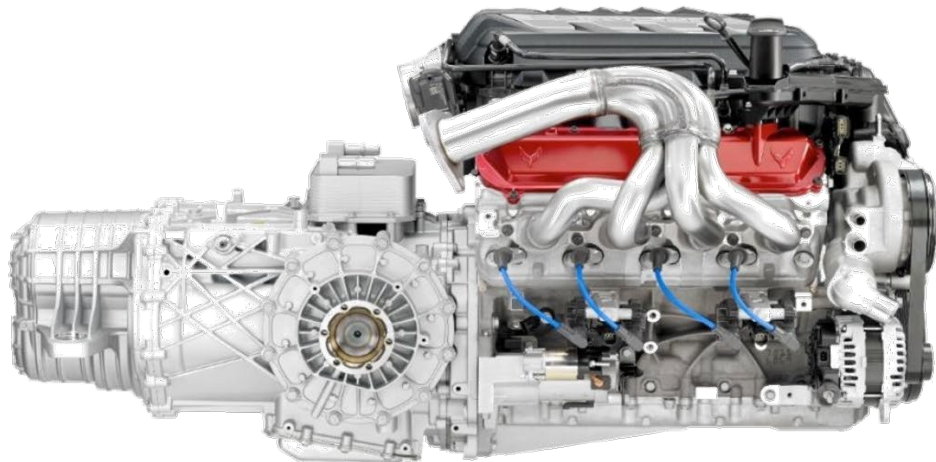
# R lingo



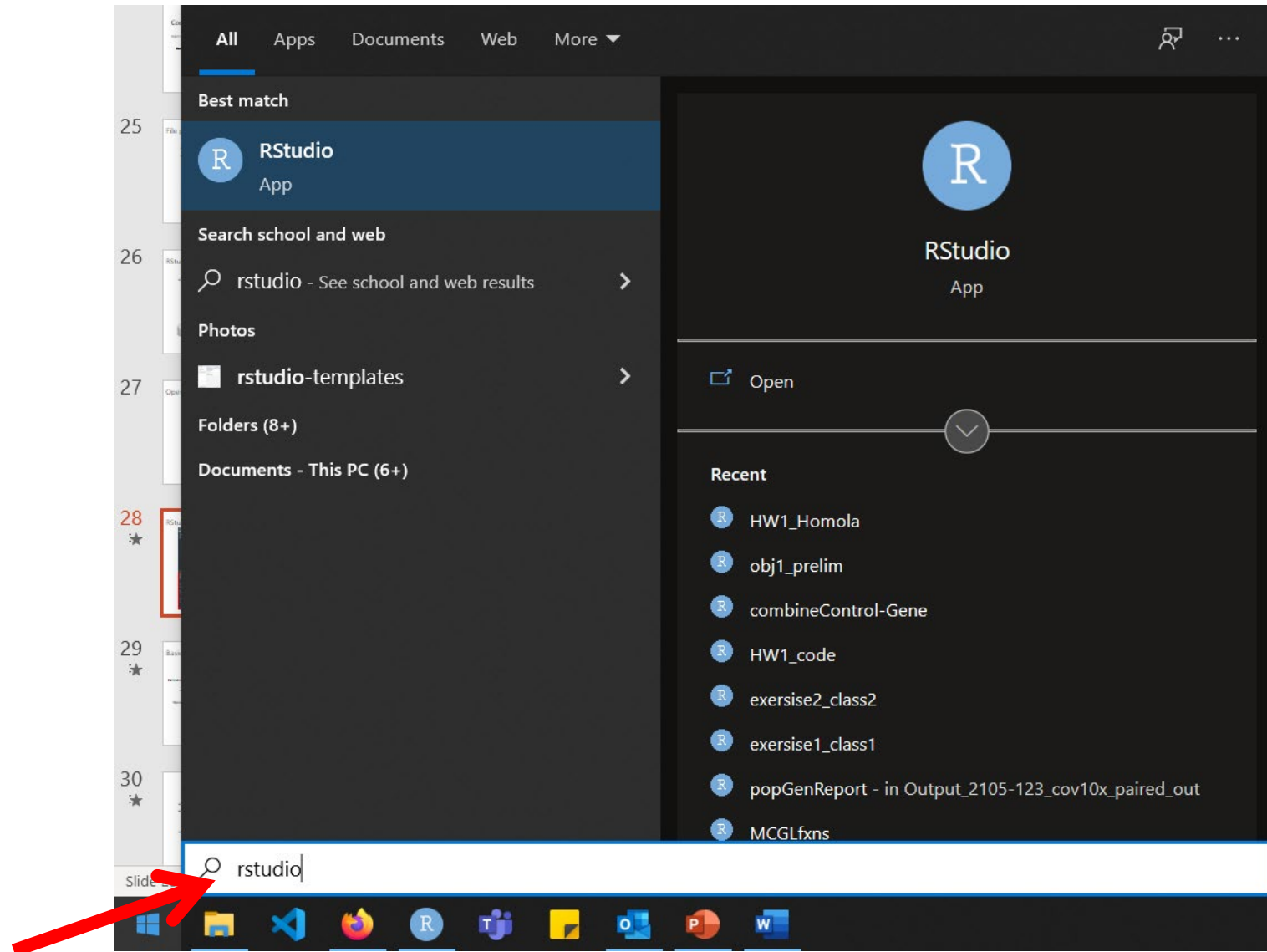


# RStudio

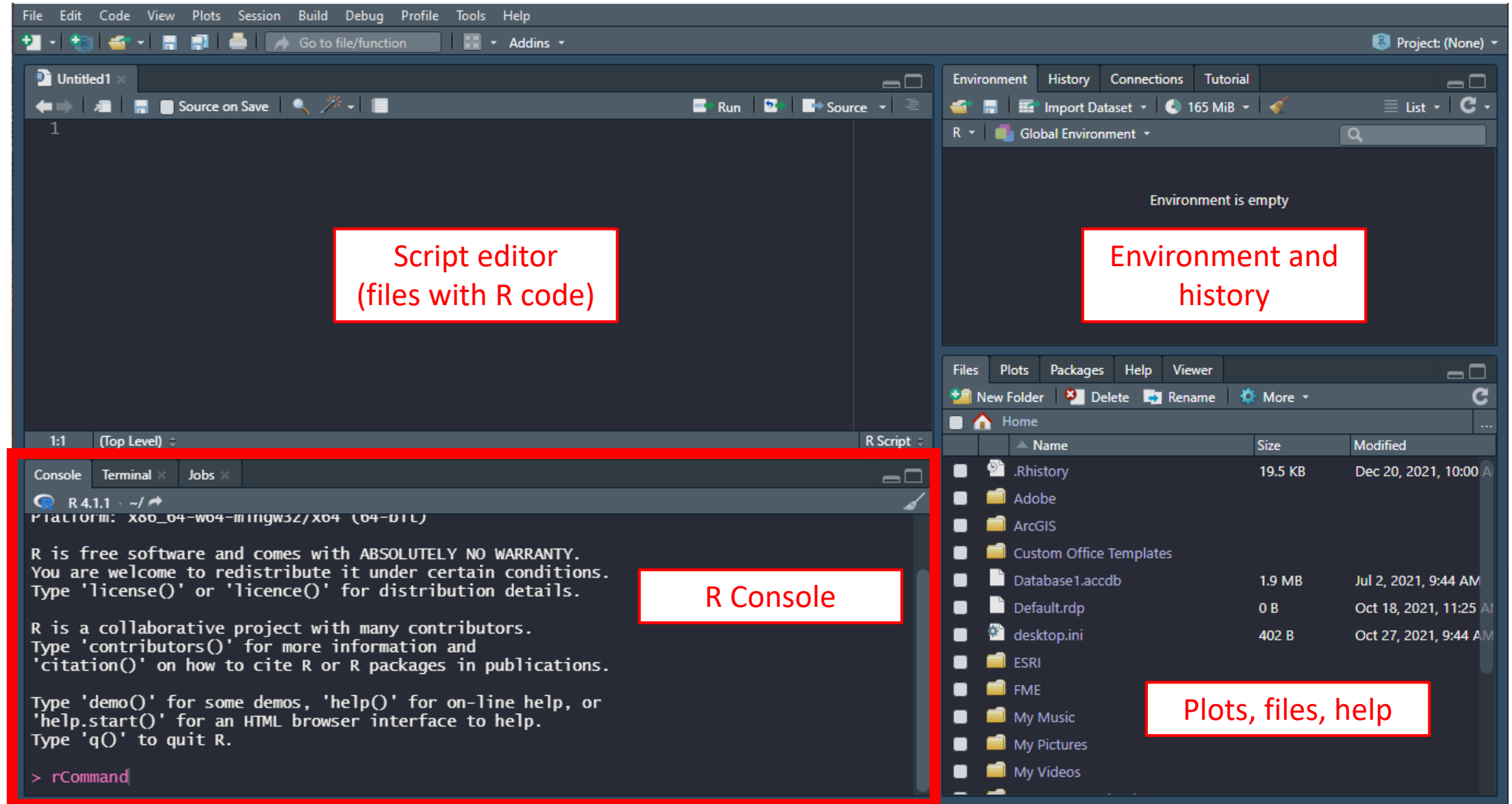
- RStudio is an integrated development environment (IDE)



# Open R studio



# RStudio





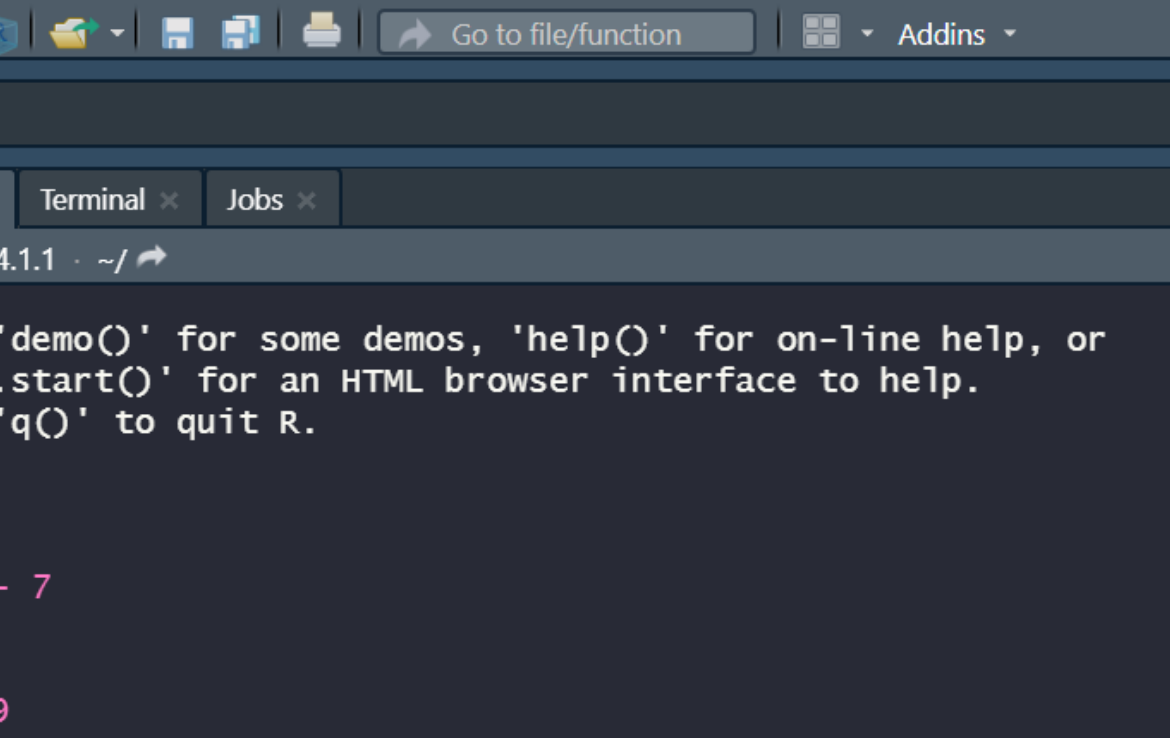
# Basic syntax

## R command prompt

**[1]** indicates the first element of a vector

## "<-" assign a value to a variable

## Operation of adding to variable's value



The screenshot displays the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar labeled 'Go to file/function'. The main workspace area is divided into three tabs: Source, Console, and Jobs. The Console tab is active, showing the R prompt and the following commands and output:

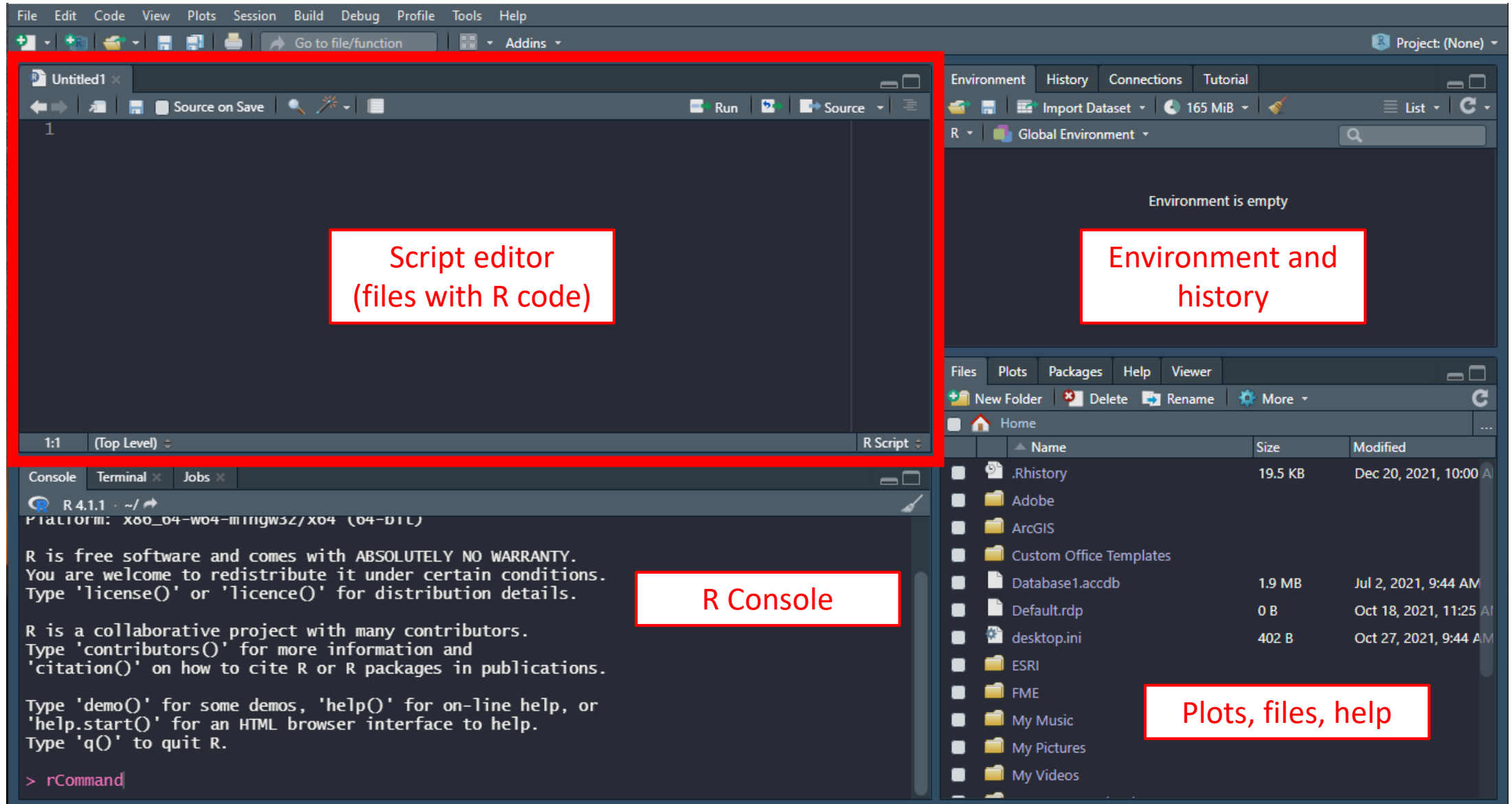
```
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.  
  
> 2+4  
[1] 6  
> x <- 7  
> x  
[1] 7  
> x+19  
[1] 26  
> |
```

Red arrows point to the input lines of the commands: `> 2+4`, `> x <- 7`, and `> x+19`.

**Note: R is case sensitive!**

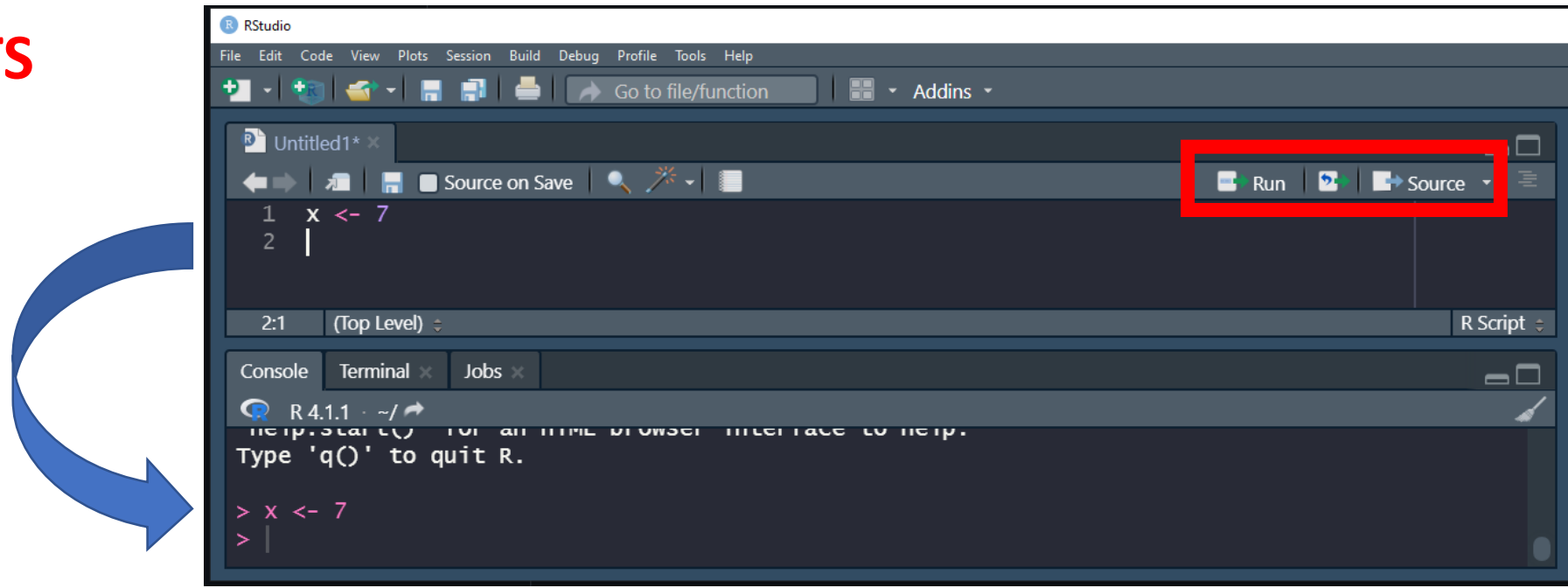
```
> x <- 10
> x
[1] 10
> X
Error: object 'X' not found
> |
```

# RStudio



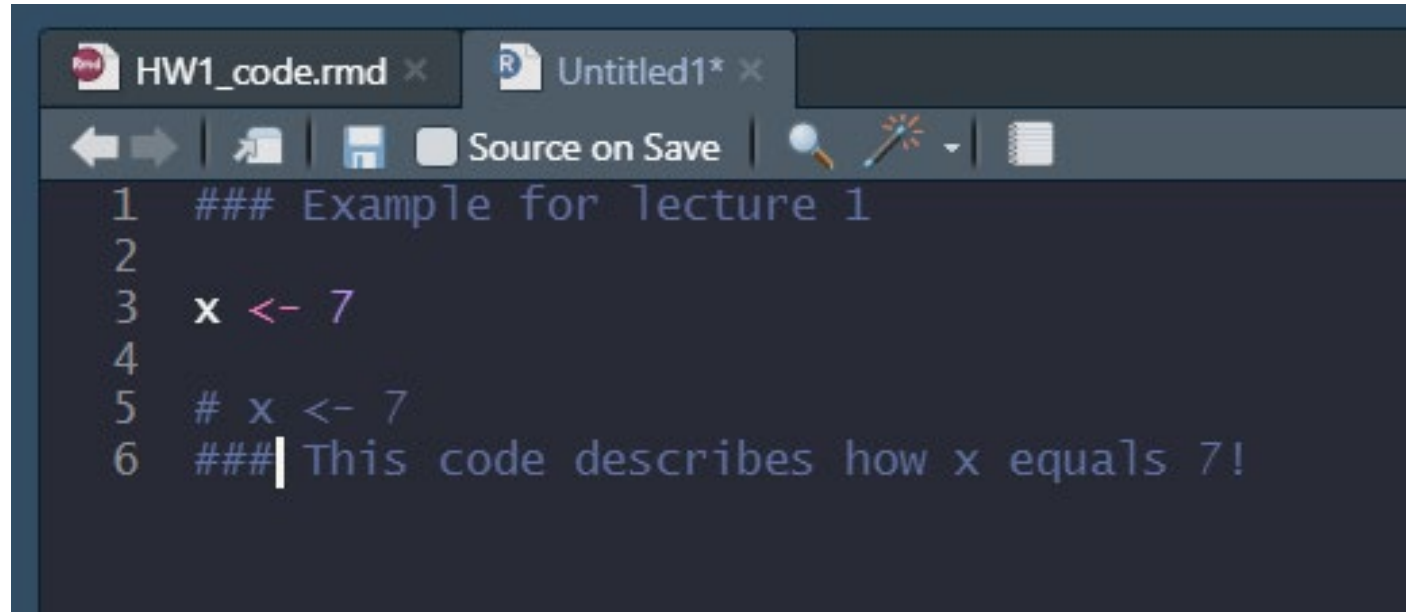
# R scripts

- A text file that contains your R code
- Reproducible: rerunning your code is easy for you or someone else
- Easily modified and rerun
- In RStudio, type `<ctrl+enter>` to run the code in the R console
- **SAVE YOUR SCRIPTS**



# R scripts

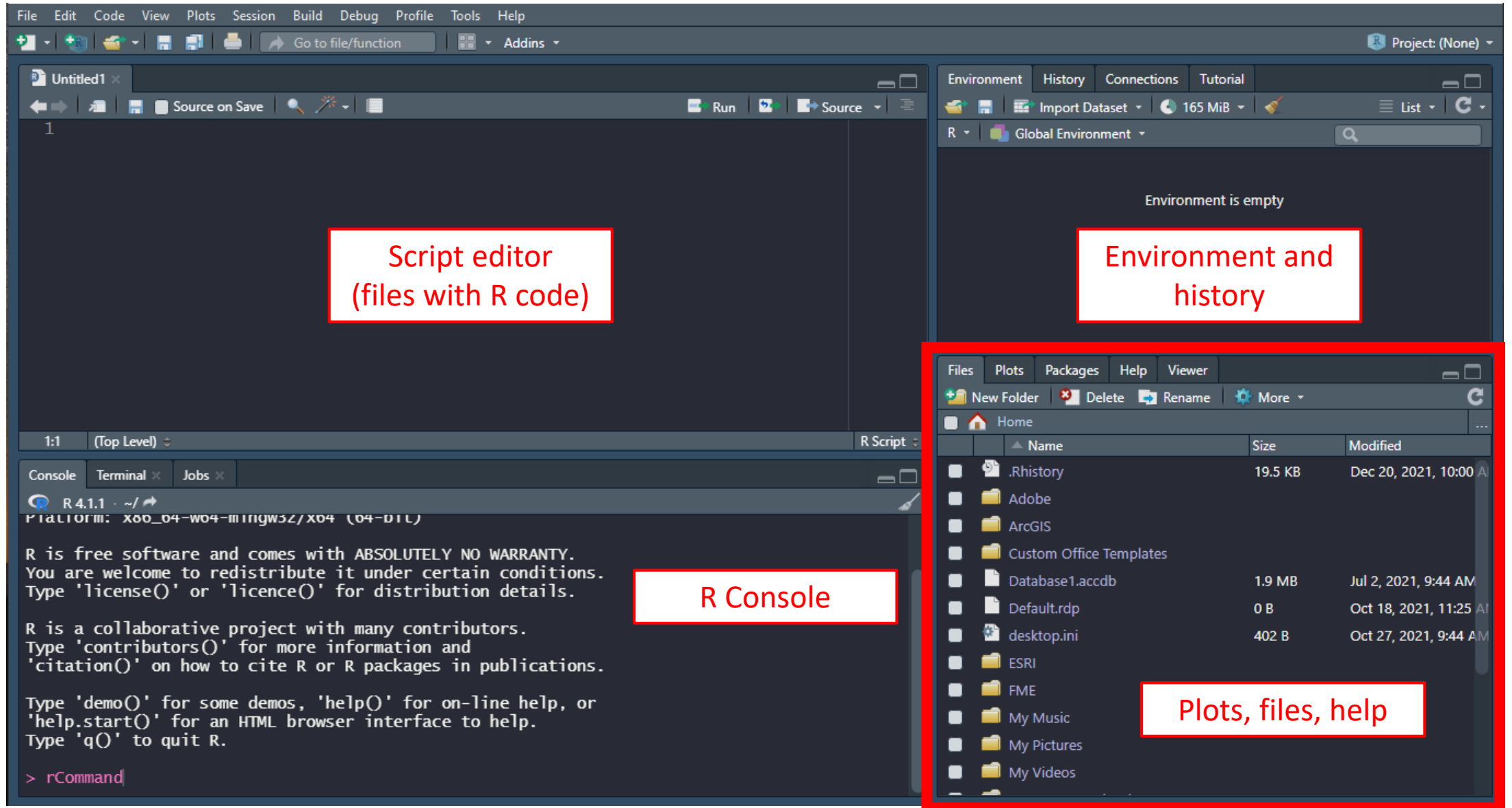
- Comment your code using `#`
- Useful for temporally removing code without deleting it
- Make notes to future users of the script (especially yourself!)
- Useful for adding headers to the top of scripts



The screenshot shows the RStudio editor interface. The top toolbar includes icons for navigation, file operations, and a 'Source on Save' button. The editor window displays the following R script:

```
1  ### Example for lecture 1
2
3  x <- 7
4
5  # x <- 7
6  ### This code describes how x equals 7!
```

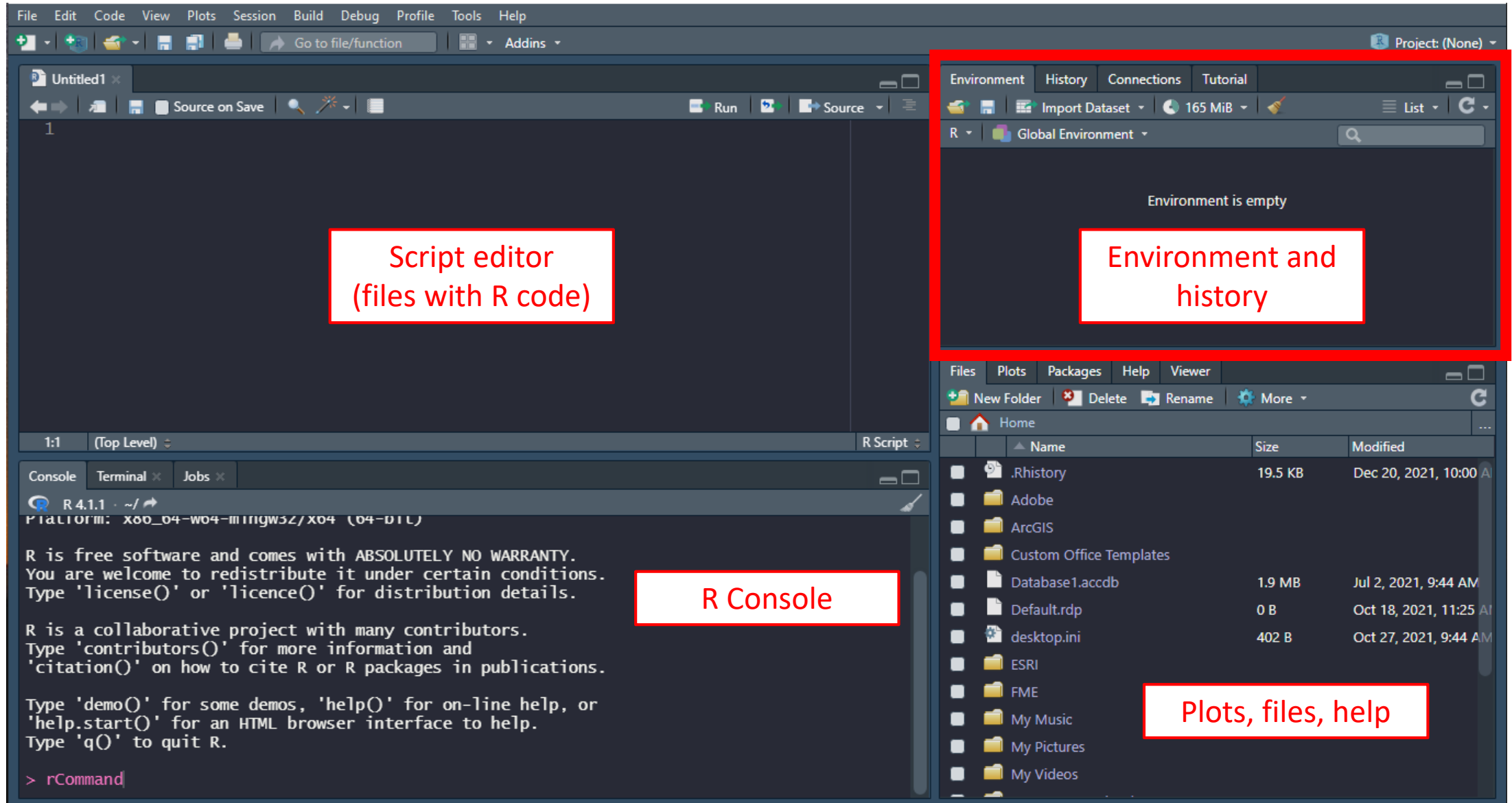
# RStudio



# R's built-in help

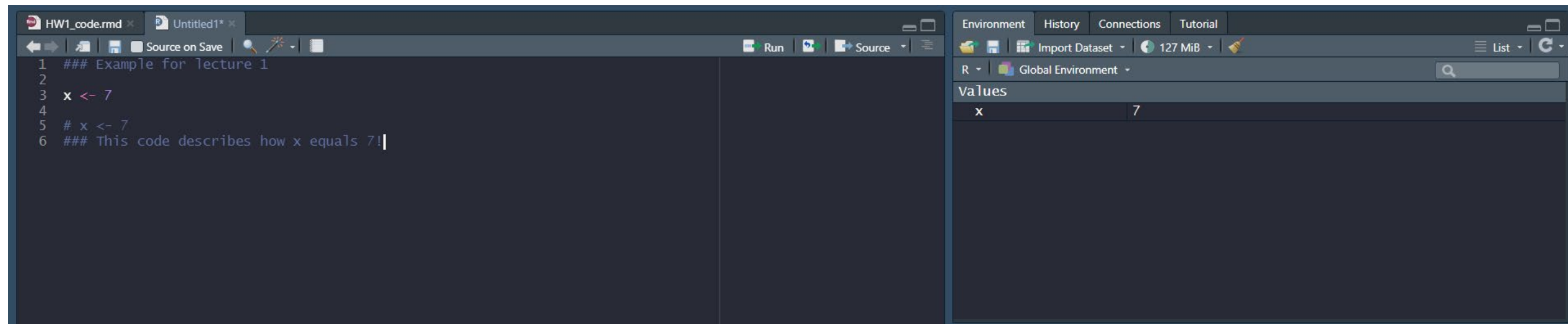
- `?mean`
  - I know this is the right function, but I can't remember how to use it
- `??mean`
  - I think this is the right function, but I could be wrong...

# RStudio



# Environment

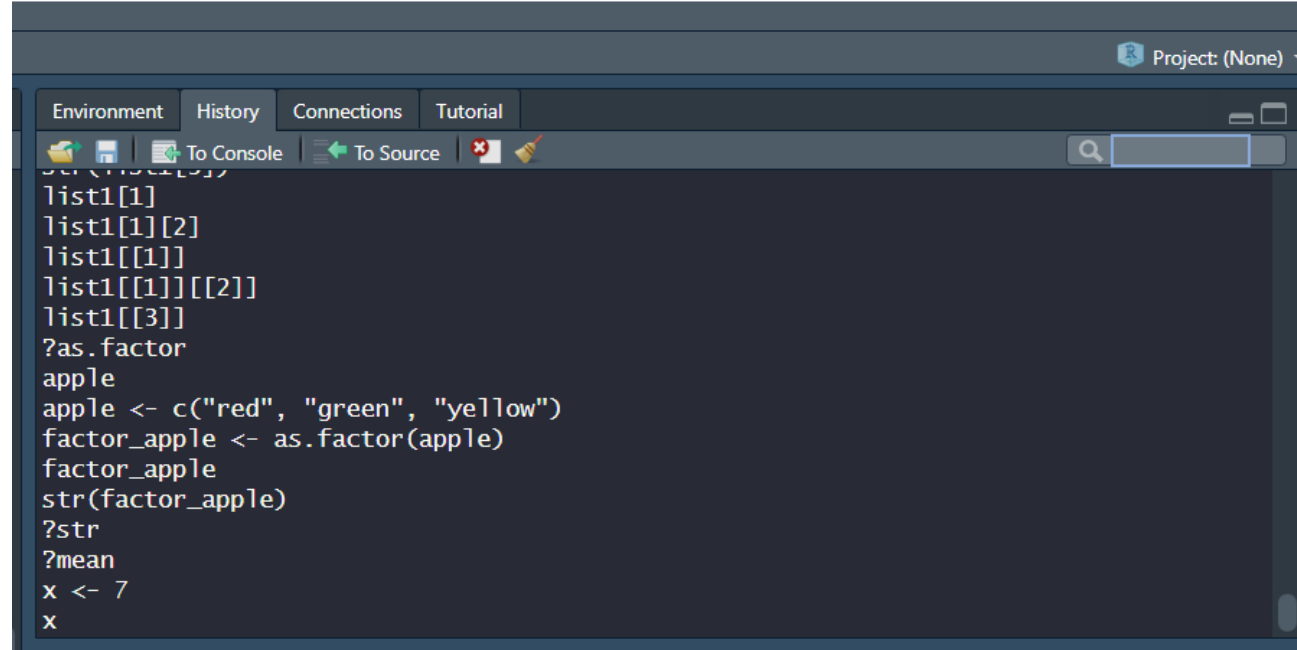
- Objects currently stored in the computer's memory
- Clean up using the broom icon





# History

- List of commands that have been recently run
- Double click to re-run something
- Also accessible using up or down arrows in the R console



The screenshot shows the RStudio interface with the 'History' pane active. The pane displays a list of commands that have been recently executed in the R console. The commands are listed in a dark-themed editor with syntax highlighting. The commands include: `list1[1]`, `list1[1][2]`, `list1[[1]]`, `list1[[1]][[2]]`, `list1[[3]]`, `?as.factor`, `apple`, `apple <- c("red", "green", "yellow")`, `factor_apple <- as.factor(apple)`, `factor_apple`, `str(factor_apple)`, `?str`, `?mean`, `x <- 7`, and `x`. The 'History' pane is located at the top of the RStudio window, below the 'Environment', 'Connections', and 'Tutorial' panes. The 'History' pane has a search bar and a 'To Console' button. The 'Environment' pane is currently selected, showing the 'Project: (None)' dropdown.

```
list1[1]
list1[1][2]
list1[[1]]
list1[[1]][[2]]
list1[[3]]
?as.factor
apple
apple <- c("red", "green", "yellow")
factor_apple <- as.factor(apple)
factor_apple
str(factor_apple)
?str
?mean
x <- 7
x
```

# Basic R coding

First, experiment with R as a calculator:

```
> 2+2*4
```

```
> 2^5+7
```

```
> 2^(5+7)
```

```
> 0.05/1E6 #note 1E6 = 1,000,000
```

# Basic R coding

Experiment with variables. For instance...

```
> name1 <- "Jared" # Note: variables cannot start with number
> name2 <- "Homo1a"
> c(name1, name2) # c = concatenate (combine)
> paste(name1, name2)
> paste0(name1, name2)
> x = 1
> y = 2
> x+y
```

# Basic R coding

Experiment with functions. For instance...

```
> sqrt(16)  
> round(3.1459)  
> ?round
```

How to round to 2 decimal points?

What if you don't state the name of the argument?

# Setting working directory

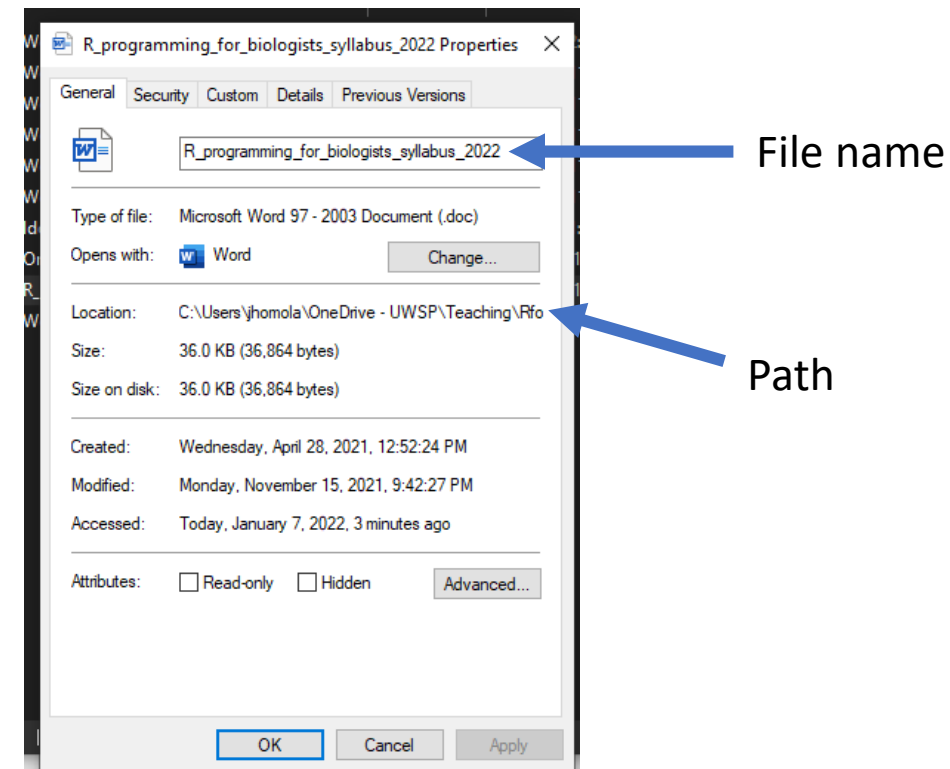
Directory: A location on your computer's hard drive or in a cloud drive

- E.g., C:\Users\jhomola\OneDrive - UWSP\Teaching\RforNatRes\_2021

The working directory becomes the default place that R looks for files.

**This will not always be encouraged!**

```
> setwd("working/directory/")
```



# Data frames

Data frame: two-dimensional array-like structure (e.g., table) with variables arranged in columns and values in each row.

All variables must have the same number of values

Make your own:

```
> var1 <- 1:4
```

```
> var2 <- 5:8
```

```
> data.frame(var1, var2)
```

# Basic subsetting

Subsetting data: isolating certain values and/or variables

Subsetting a column:

```
> dat <- data.frame(var1, var2)  
> dat$var1
```

Subsetting a column, row, or single value

```
> dat[1,] #First slot is for complete rows  
> dat[,1] #Second slot is for complete columns  
> dat[1,2] #Combining them gives a single value  
> dat[1] #No comma gives values in column as vector  
> dat[1:3,] #Colons indicate ranges
```

# Basic summary statistics

```
> sum(dat$var1)
> min(dat$var2)
> max(dat$var1)
> sd(dat$var2) #Standard deviation
```