

Introduction to R

Using R Studio

An IDE for R



An integrated development environment is software that makes coding easier

- see objects you've imported and created
- autocomplete
- syntax highlighting
- run part or all of your code

The image shows the RStudio interface with several components highlighted and explained:

- CODE, FILE, SCRIPT**: This is where you write code you want to save. The screenshot shows a script file named 'day1-script1.R' with R code comments and a function definition.
- ENVIRONMENT**: This is where you see what objects you've created and data you've loaded. The screenshot shows the 'Global Environment' pane.
- FILES**: This is where you see the files in your project. The screenshot shows a file browser view of a project folder containing several R script files and other files.
- CONSOLE**: This is where results print out and where you write code you don't want to save. The screenshot shows the R console output, including the R version and platform information.

PLOTS and **HELP** are also visible in the interface but not explicitly annotated with text boxes.

Start fresh

- If you have used R previously, an old workspace may still be active when you open RStudio
- You always want to start with a fresh session
- Go to Tools -> Global Options, and under General, change these settings:

Workspace

Restore .RData into workspace at startup

Save workspace to .RData on exit: Never ▾



Tip

Now, you can just quit and restart RStudio if something goes wrong! You can also go to Session -> Restart R to clear your session.

Using R

Running code

You can run...

- code that you type directly in the console
 - code you won't need to run again
- code in an `.R` script
- code in a `.qmd` (Quarto) or `.Rmd` (R Markdown) file
 - code you want to render to an html, word, or pdf file

Running code

Make sure your cursor is on the line you want to run ->
cmd+return (Mac) or ctrl+enter (Windows)

- This will automatically bring your cursor to the next line

You can also highlight a section of code and hit
cmd+return/ctrl+enter or use the “Run” dropdown in
RStudio

Types of documents

- R scripts: `.R` files
 - plain text files with R code
 - can be run line-by-line in RStudio (or all at once)
- Quarto/R Markdown: `.qmd` or `.Rmd` files
 - plain text files with R code and text
 - can be run line-by-line or in “chunks” in RStudio
 - can be rendered to produce a report in HTML, PDF, Word, etc.

Quarto

We'll generally use `.qmd` files (Quarto documents) in this course

- Encourage good coding practices
 - Every time a document is rendered, the code is run from top to bottom
 - When coding line-by-line, sometimes you can get code that only works out of order or depends on something you've only run in the console
- When writing code, run line-by-line/in chunks, but render often

Running code in quarto

Your text here

```
```${r}  
your_code_here
```
```



- Run “line-by-line” by putting your cursor in the chunk and hitting cmd+return/ctrl+enter
- Run the whole chunk by clicking the green arrow
- Run all chunks above this one by clicking the grey arrow with green bar
 - This mimics what will happen when you render (start a new R session first)

Demo

R objects

Dataframes

The most common object we'll be using R is a *dataframe*

- this is the equivalent to a dataset in Stata, but you can have many in your environment at the same time
- we'll also use other objects: in R, a function is an object, a figure is an object, a single number can be an object, etc.

We'll talk briefly about some of these

- Also see the optional reading ([Chapter 2](#) of *Hands-On Programming with R*) for more practice

The biggest difference between R and Stata is that R can have many different objects in its environment

- datasets, numbers, figures, etc.
- you have to be explicit about storing and retrieving objects
 - e.g., what dataset a variable belongs to

R uses `<-` to store objects in the environment

I call this the “assignment arrow”

```
1 # create values
2 vals <- c(1, 645, 329)
```

Now `vals` holds those values

- this is called a vector

Warning

No assignment arrow means that the object will be printed to the console (and lost forever!)

Objects

We can retrieve those values by running just the name of the object

```
1 vals
```

```
[1] 1 645 329
```

We can also perform operations on them using functions like `mean()`

```
1 mean(vals)
```

```
[1] 325
```

If we want to keep the result of that operation, we need to use `<-` again

```
1 mean_val <- mean(vals)
```

Types of data (*classes*)

We could also create a character *vector*.

```
1 chars <- c("dog", "cat", "rhino")  
2 chars
```

```
[1] "dog" "cat" "rhino"
```

Or a *logical* vector:

```
1 logs <- c(TRUE, FALSE, FALSE)  
2 logs
```

```
[1] TRUE FALSE FALSE
```



We'll see more options as we go along!

Types of objects

We created *vectors* with the `c()` function (`c` stands for concatenate)

We could also create a *matrix* of values with the `matrix()` function:

```
1 # turn the vector of numbers into a 2-row matrix
2 mat <- matrix(c(234, 7456, 12, 654, 183, 753), nrow = 2)
3 mat
```

```
      [,1] [,2] [,3]
[1,]  234   12  183
[2,] 7456  654  753
```

Indices

The numbers in square brackets are *indices*, which we can use to pull out values:

```
1 # extract second animal  
2 chars[2]
```

```
[1] "cat"
```

We can pull out rows or columns from matrices:

```
1 # extract second row  
2 mat[2, ]
```

```
[1] 7456 654 753
```

```
1 # extract first column  
2 mat[, 1]
```

```
[1] 234 7456
```

Demo

Packages

- Some functions are built into R
 - `mean()`, `lm()`, `table()`, etc.
- They actually come from built-in packages
 - `base`, `stats`, `graphics`, etc.
- Anyone (yes, *anyone*) build their own package to add to the functionality of R
 - `{ggplot2}`, `{dplyr}`, `{data.table}`, `{survival}`, etc.



1

1. Image from Zhi Yang

Packages

- You have to **install** a package once¹

```
1 install.packages("survival")
```

- You then have to **load** the package every time you want to use it

```
1 library(survival)
```

1. Actually, with every new major R release, but we won't worry about that.

Packages

“You only have to buy the book once, but you have to go get it out of the bookshelf every time you want to read it.”

```
1 install.packages("survival")  
2 library(survival)  
3 survfit(...)
```

SEVERAL DAYS LATER..

```
1 library(survival)  
2 coxph(...)
```

Package details

- When you use `install.packages`, packages are downloaded from **CRAN** (The Comprehensive R Archive Network)
 - This is also where you downloaded R
- Packages can be hosted lots of other places, such as **Bioconductor** (for bioinformatics), and **Github** (for personal projects or while still developing)
- The folks at CRAN check to make things “work” in some sense, but don’t check on the statistical methods...
 - But because R is open-source, you can always read the code yourself
- Two functions from different packages can have the same name... if you load them both, you may have some trouble

Demo