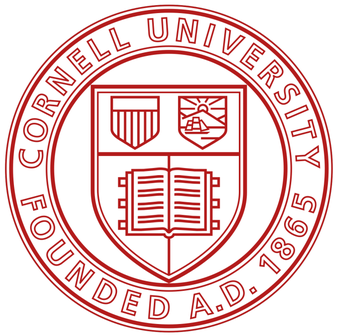
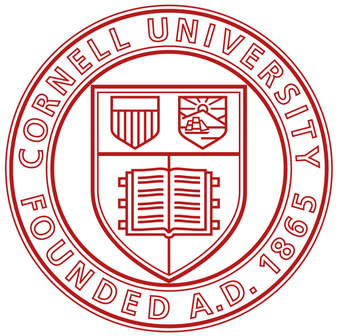


Introduction to R programming

STSCI 2120

Nayel Bettache - Fall 2024



Personal Presentation

Nayel Bettache

Visiting Assistant Professor

Nayel Bettache

Visiting Assistant Professor

- PhD in mathematical statistics from Institut Polytechnique de Paris

🇫🇷

Nayel Bettache

Visiting Assistant Professor

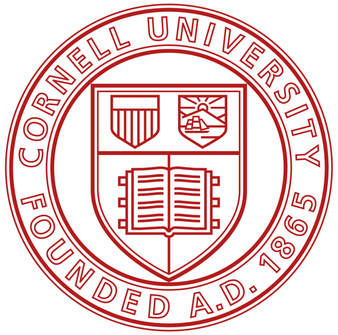
- PhD in mathematical statistics from Institut Polytechnique de Paris
- MSc in DataScience from Ecole Polytechnique

🇫🇷

Nayel Bettache

Visiting Assistant Professor

- PhD in mathematical statistics from Institut Polytechnique de Paris
- MSc in DataScience from Ecole Polytechnique
- MSc in Statistics from ENSAE Paris



Overview of the course

What will you learn ?

Objectives

What will you learn ?

Objectives

- Help you learn the most important tools in R

What will you learn ?

Objectives

- Help you learn the most important tools in R
- Allow you to do data science efficiently and reproducibly

What will you learn ?

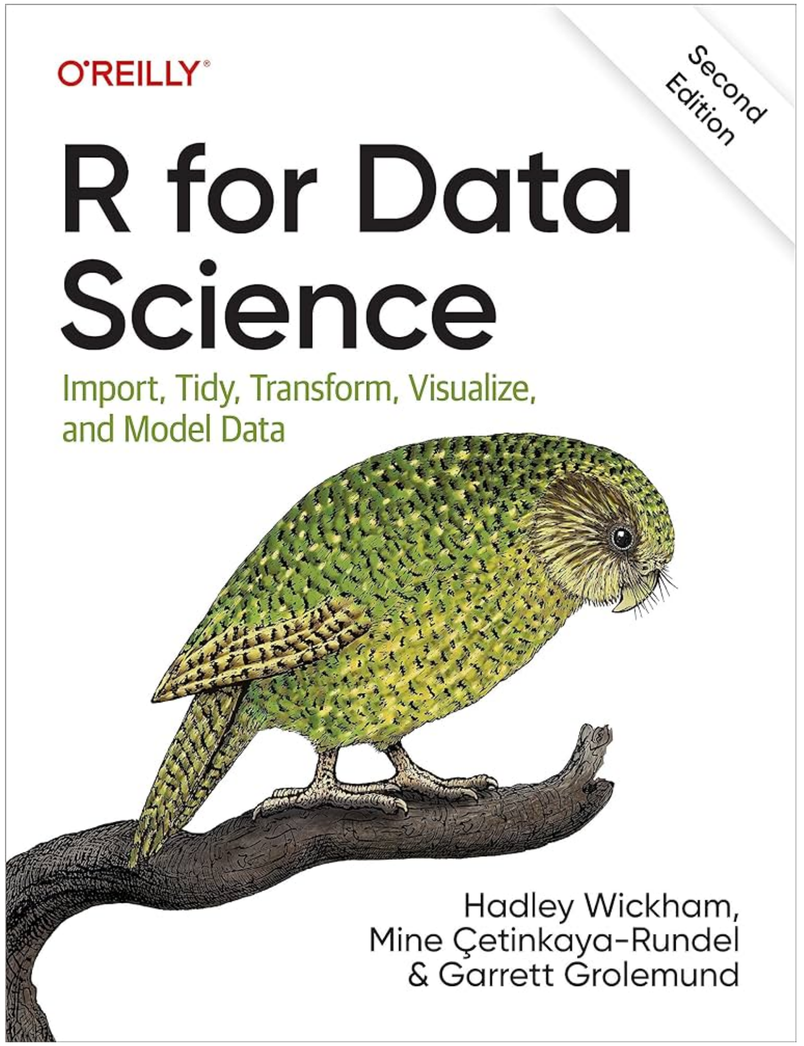
Objectives

- Help you learn the most important tools in R
- Allow you to do data science efficiently and reproducibly
- Have some fun

What will you learn ?

Ressources

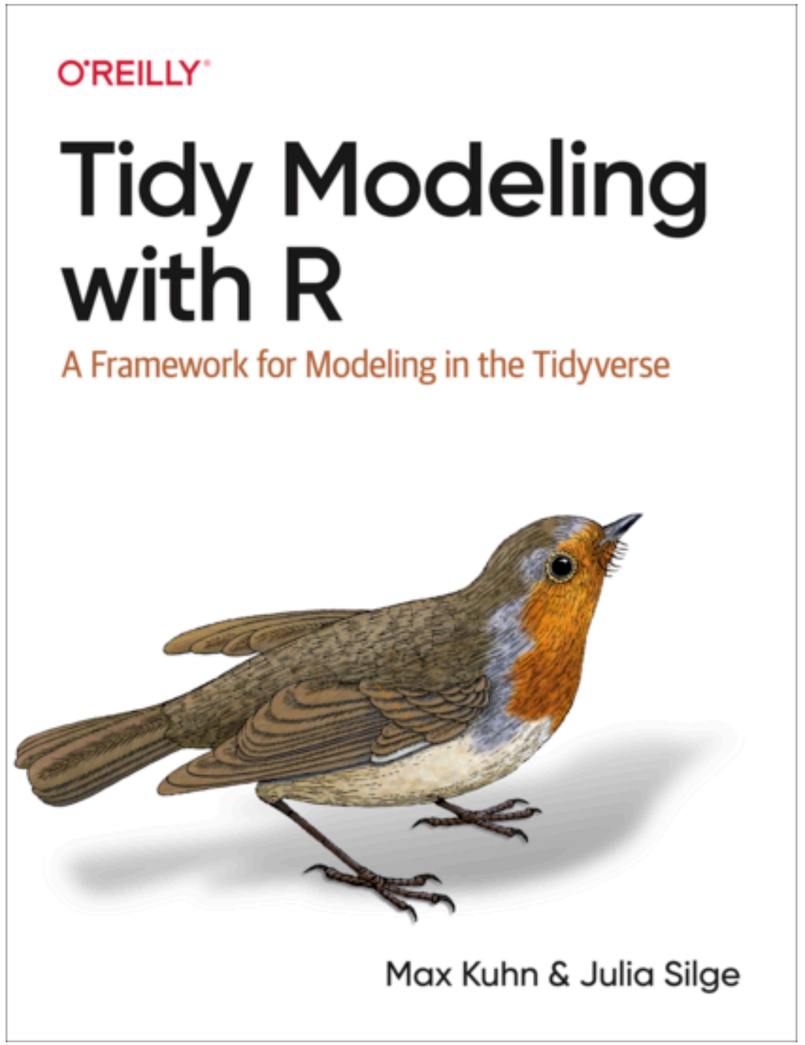
- R for Data Science



What will you learn ?

Ressources

- R for Data Science
- Tidy modeling with R



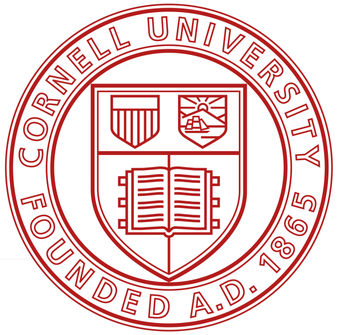
What will you learn ?

Ressources

- R for Data Science
- Tidy modeling with R
- R programming tutorial

What will you learn ?

Ressources



- R for Data Science
- Tidy modeling with R
- R programming tutorial
- An introduction to R

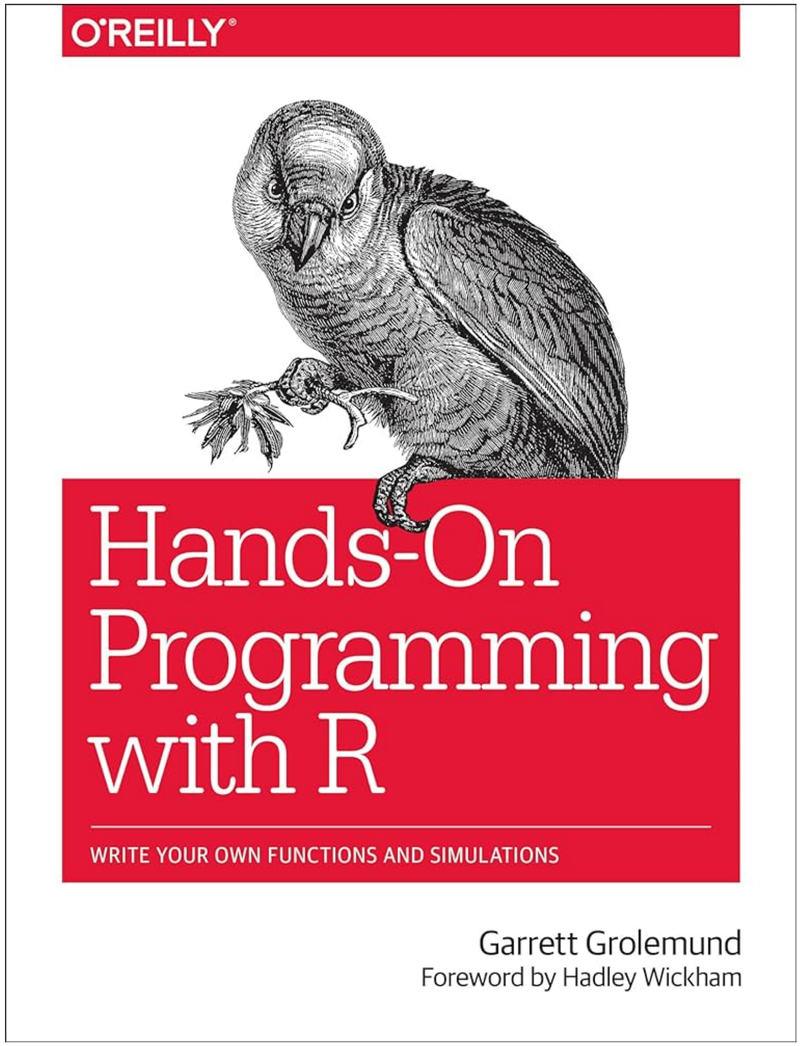
An Introduction to R

Notes on R: A Programming Environment for Data Analysis and Graphics
Version 4.4.1 (2024-06-14)

What will you learn ?

Ressources

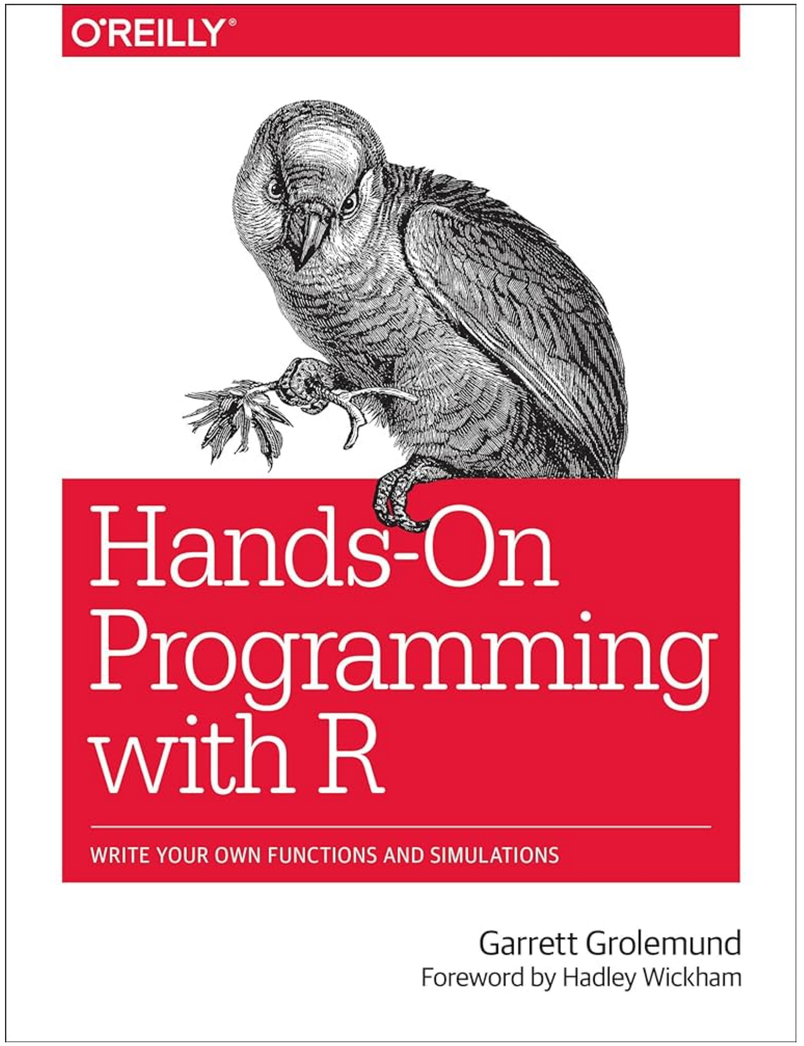
- R for Data Science
- Tidy modeling with R
- R programming tutorial
- An introduction to R
- Hands on Programming with R



What will you learn ?

Main ressource

- Hands on Programming with R



What will you learn ?

Details

What will you learn ?

Details

- https://nayelbettache.github.io/STSCI2120.html

What will you learn ?

Details

- https://nayelbettache.github.io/STSCI2120.html
- Info about the course, homework, syllabus.

What will you learn ?

Details

- https://nayelbettache.github.io/STSCI2120.html
- Info about the course, homework, syllabus.
- Lectures on Tuesdays and Thursdays, 10:10am-11:25am, Emerson Hall 135.

What will you learn ?

Details

- https://nayelbettache.github.io/STSCI2120.html
- Info about the course, homework, syllabus.
- Lectures on Tuesdays and Thursdays, 10:10am-11:25am, Emerson Hall 135.
- Office hours: Thursdays 2:30 - 3:30

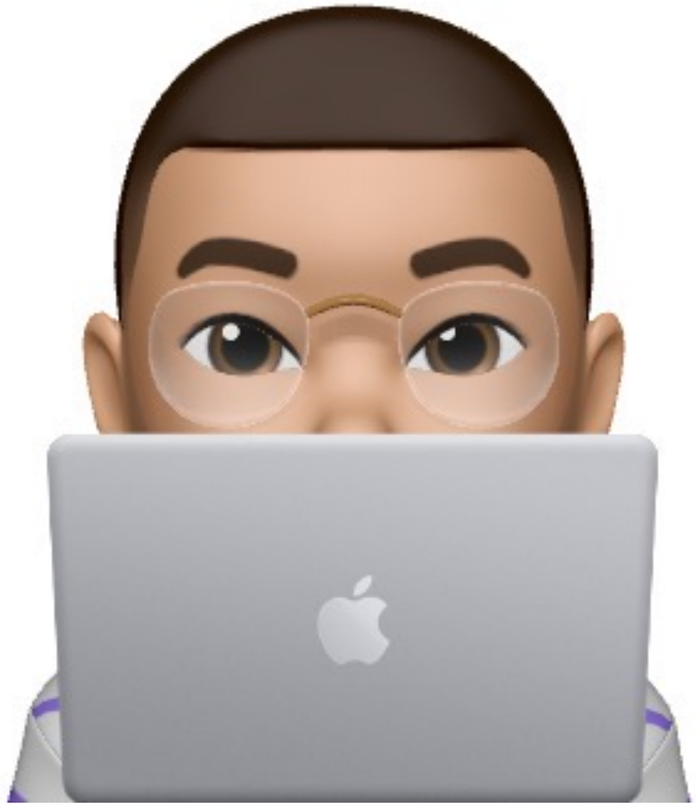
What won’t you learn ?

Details

- Modeling
- Big data
- Python, Julia and others

7 weeks

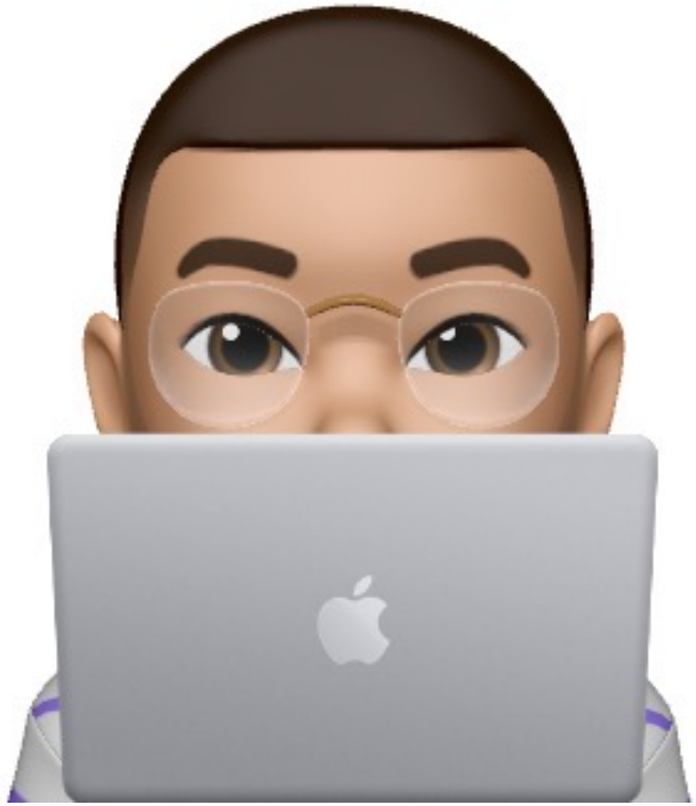
Overview



7 weeks

Overview

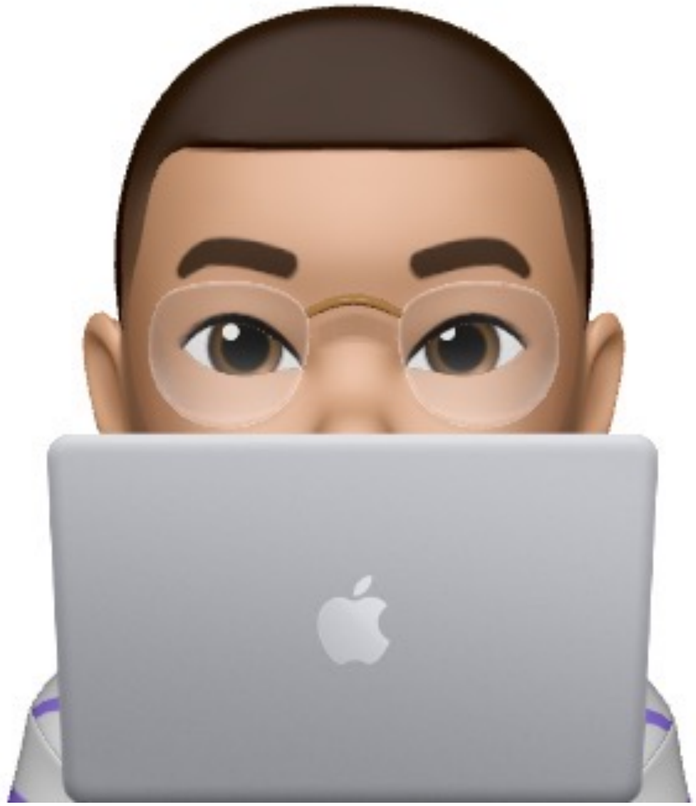
- W1: Introduction to R



7 weeks

Overview

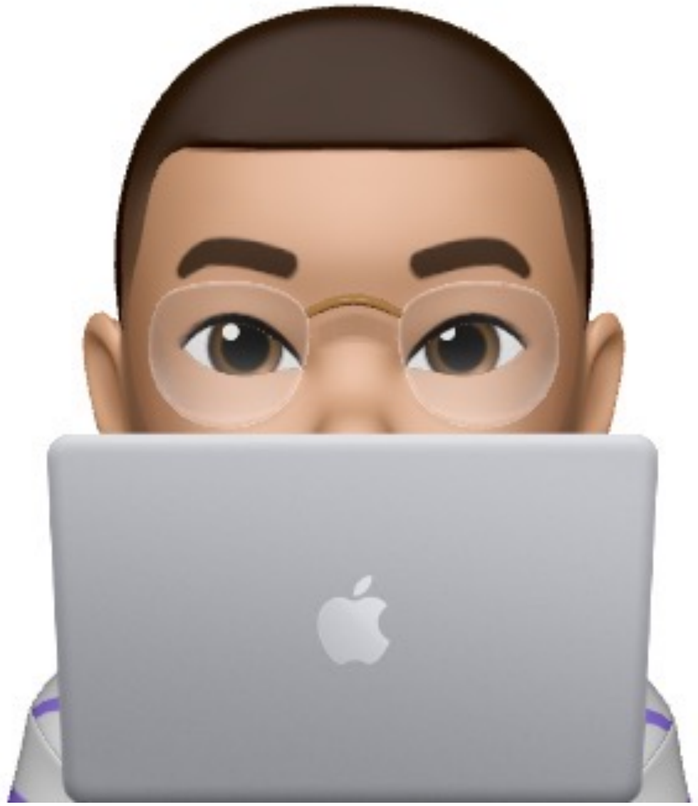
- W1: Introduction to R
- W2: R Objects and R Notation



7 weeks

Overview

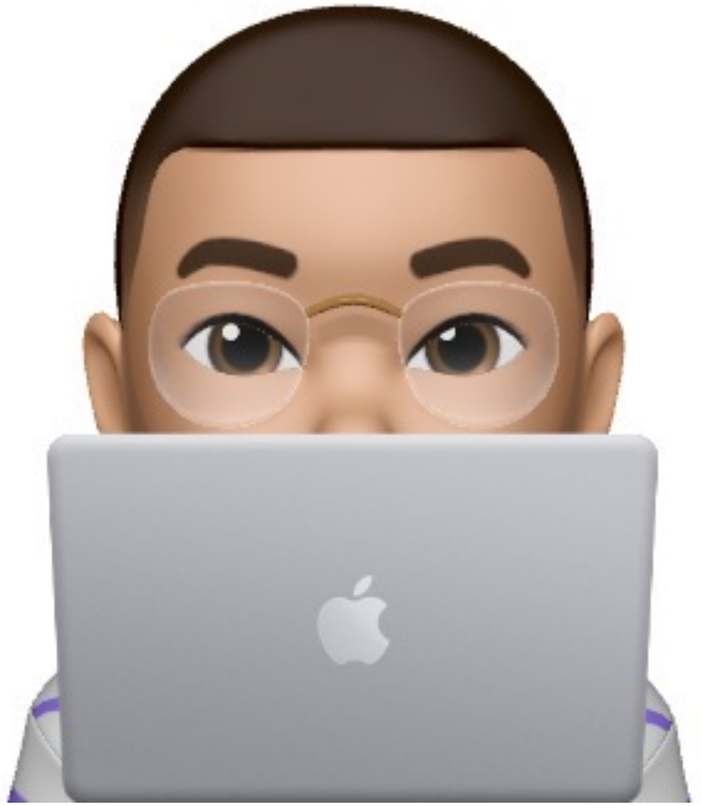
- W1: Introduction to R
- W2: R Objects and R Notation
- W3: Modifying values and Environments



7 weeks

Overview

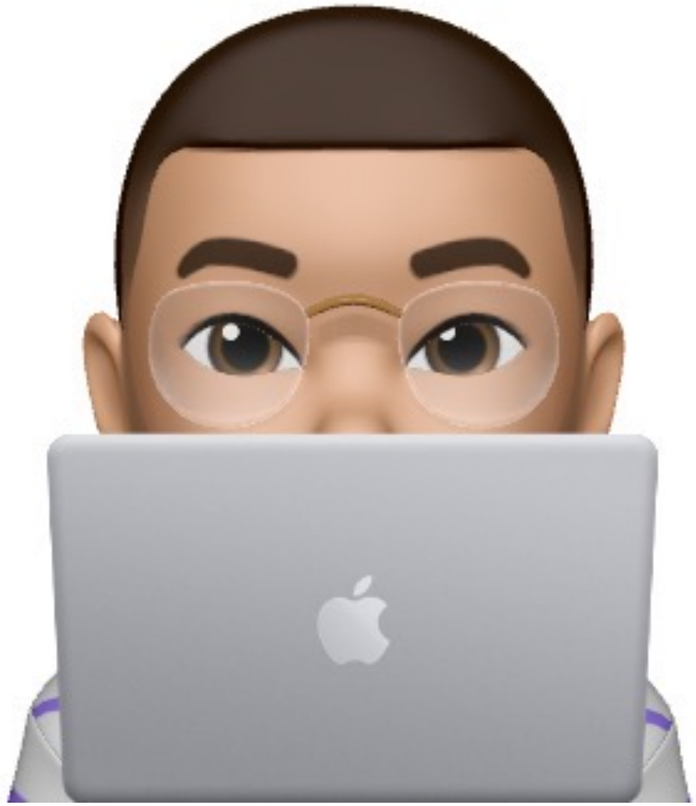
- W1: Introduction to R
- W2: R Objects and R Notation
- W3: Modifying values and Environments
- W4: Programs and S3



7 weeks

Overview

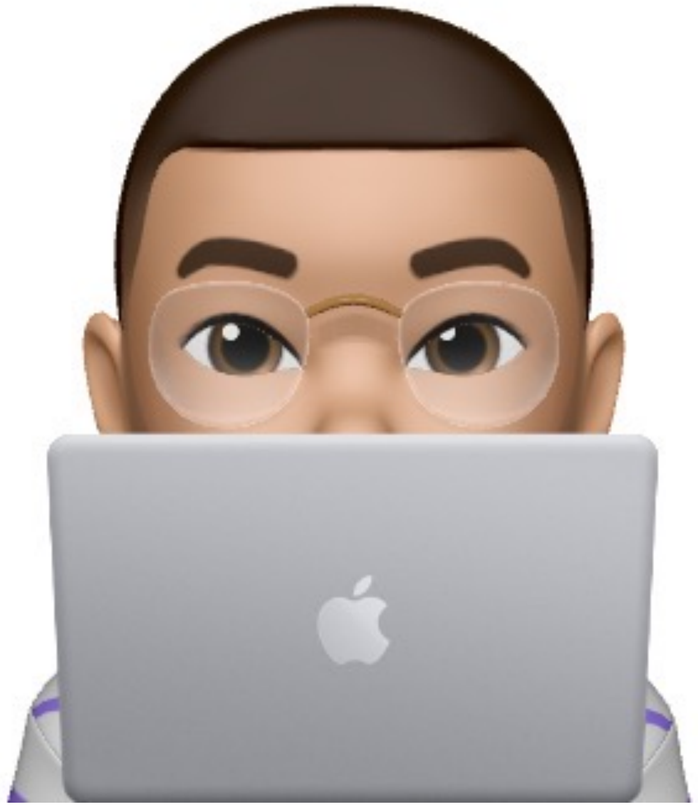
- W1: Introduction to R
- W2: R Objects and R Notation
- W3: Modifying values and Environments
- W4: Programs and S3
- W5: Programs and S3



7 weeks

Overview

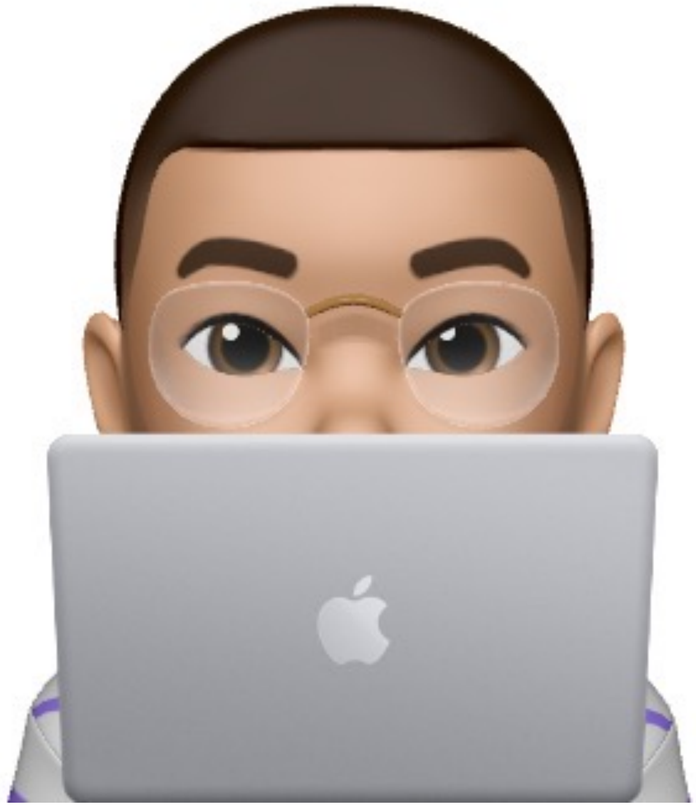
- W1: Introduction to R
- W2: R Objects and R Notation
- W3: Modifying values and Environments
- W4: Programs and S3
- W5: Programs and S3
- W6: Working with data

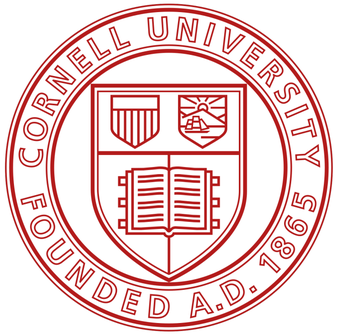


7 weeks

Overview

- W1: Introduction to R
- W2: R Objects and R Notation
- W3: Modifying values and Environments
- W4: Programs and S3
- W5: Programs and S3
- W6: Working with data
- W7: Final Project





Introduction to R and RStudio

Install R

What is it ?

Install R

What is it ?

- R is a computer language

Install R

What is it ?

- R is a computer language
- It isn’t a program you can open and start using

Install R

What is it ?

- R is a computer language
- It isn’t a program you can open and start using
- R is maintained by an international team of developers

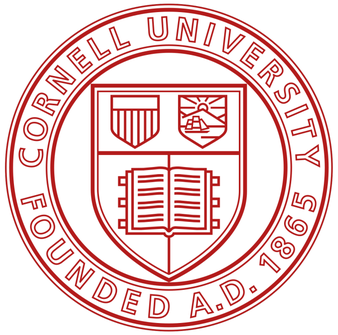
Install R

What is it ?

- R is a computer language
- It isn’t a program you can open and start using
- R is maintained by an international team of developers
- Info: [The Comprehensive R Archive Network]

Install R

How to install it ?



The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

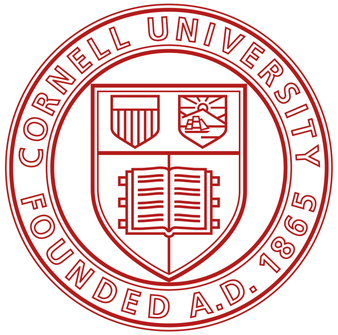
Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2024-06-14, Race for Your Life) R-4.4.1.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Install R

How to install it ?

- Go on: [The Comprehensive R Archive Network]



The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

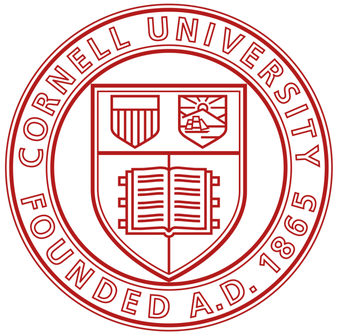
Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2024-06-14, Race for Your Life) R-4.4.1.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Install R

How to install it ?

- Go on: The Comprehensive R Archive Network
- Follow the link that describes your operating system: Windows, Mac, or Linux.



The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2024-06-14, Race for Your Life) R-4.4.1.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Install RStudio

What is it ?

Install RStudio

What is it ?

- RStudio is an application like Microsoft Word.

Install RStudio

What is it ?

- RStudio is an application like Microsoft Word.
- Instead of helping you write in English, RStudio helps you write in R.

Install RStudio

What is it ?

- RStudio is an application like Microsoft Word.
- Instead of helping you write in English, RStudio helps you write in R.
- The RStudio interface looks the same for Windows, Mac OS, and Linux.

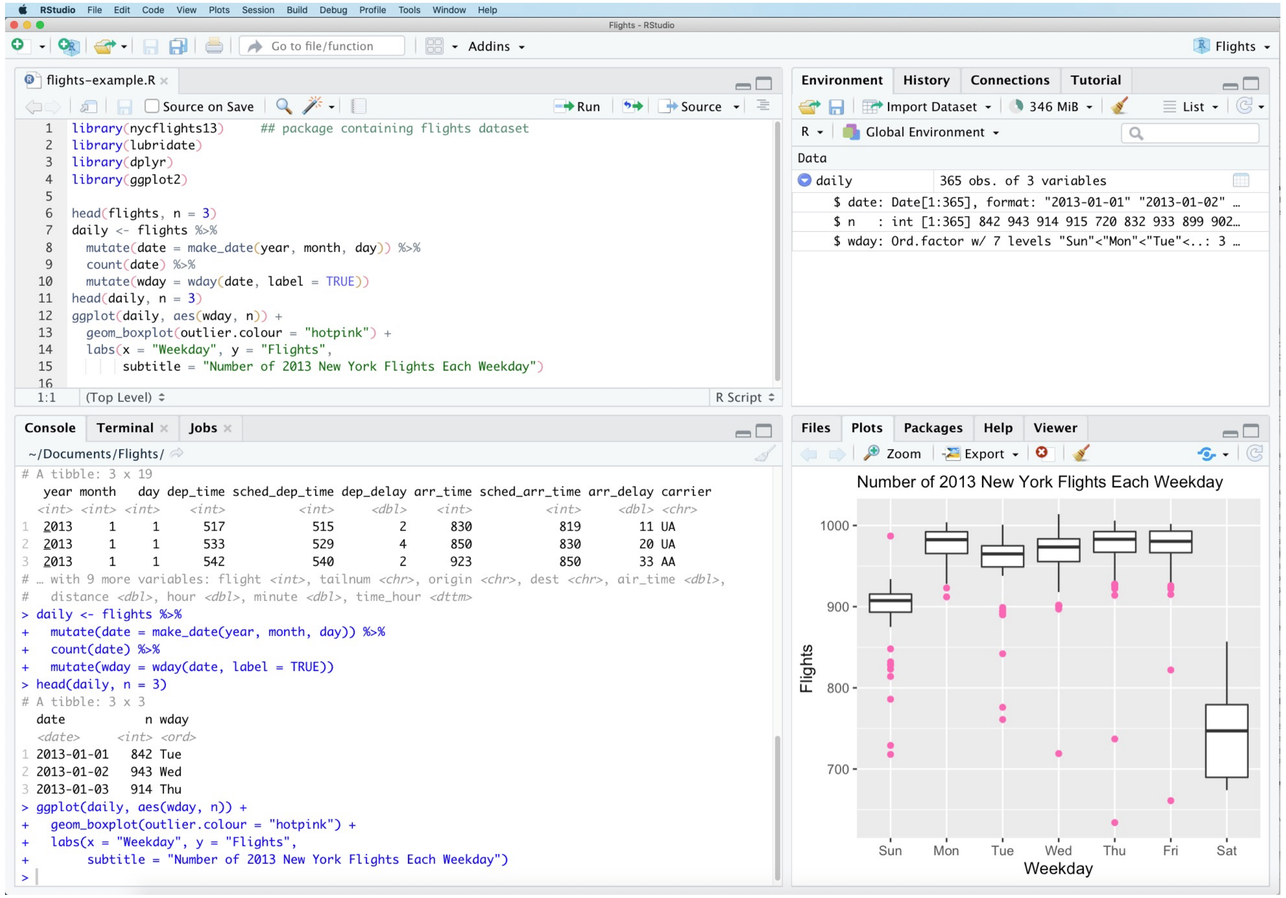
Install RStudio

What is it ?

- RStudio is an application like Microsoft Word.
- Instead of helping you write in English, RStudio helps you write in R.
- The RStudio interface looks the same for Windows, Mac OS, and Linux.
- It is an IDE

Install RStudio

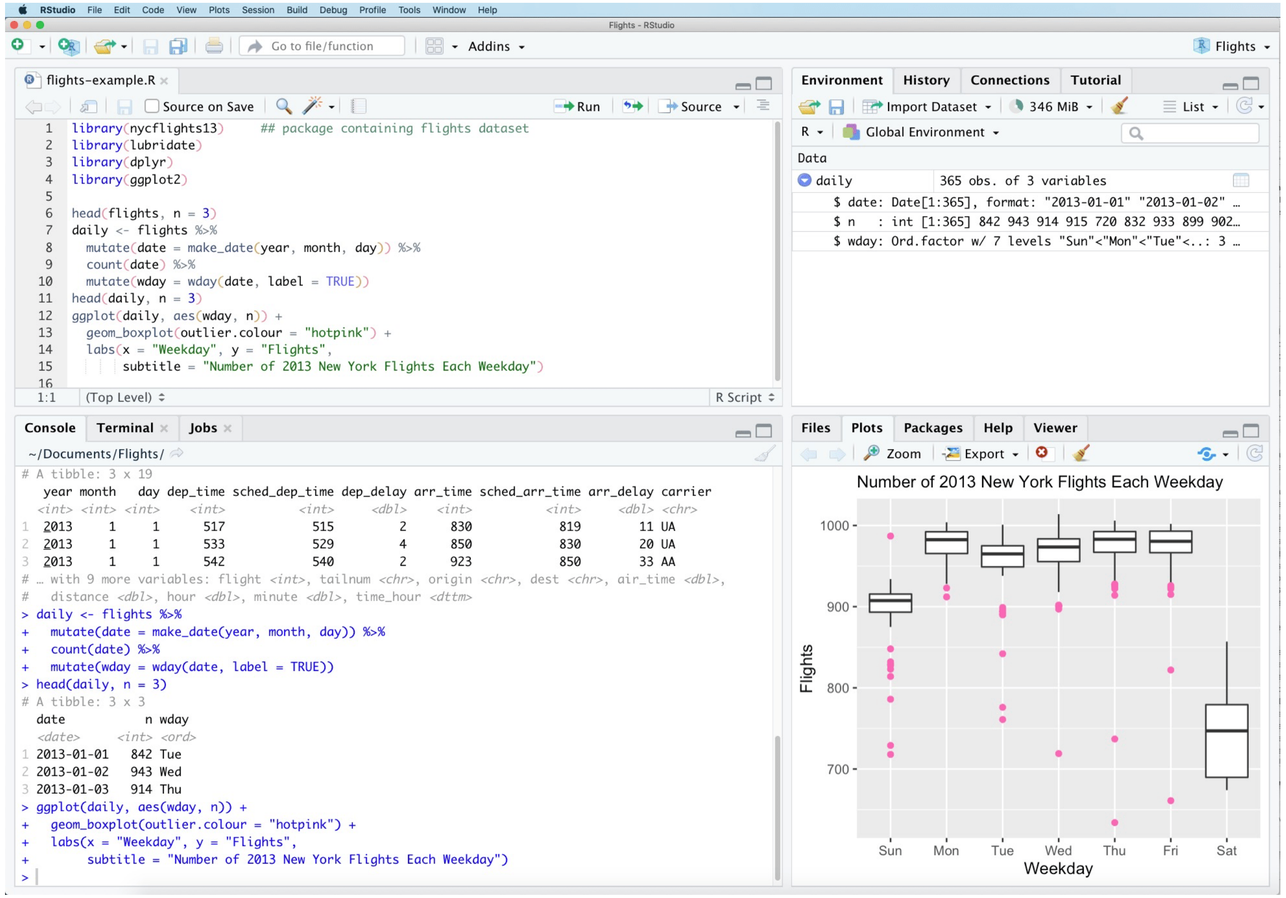
How to install it ?



Install RStudio

How to install it ?

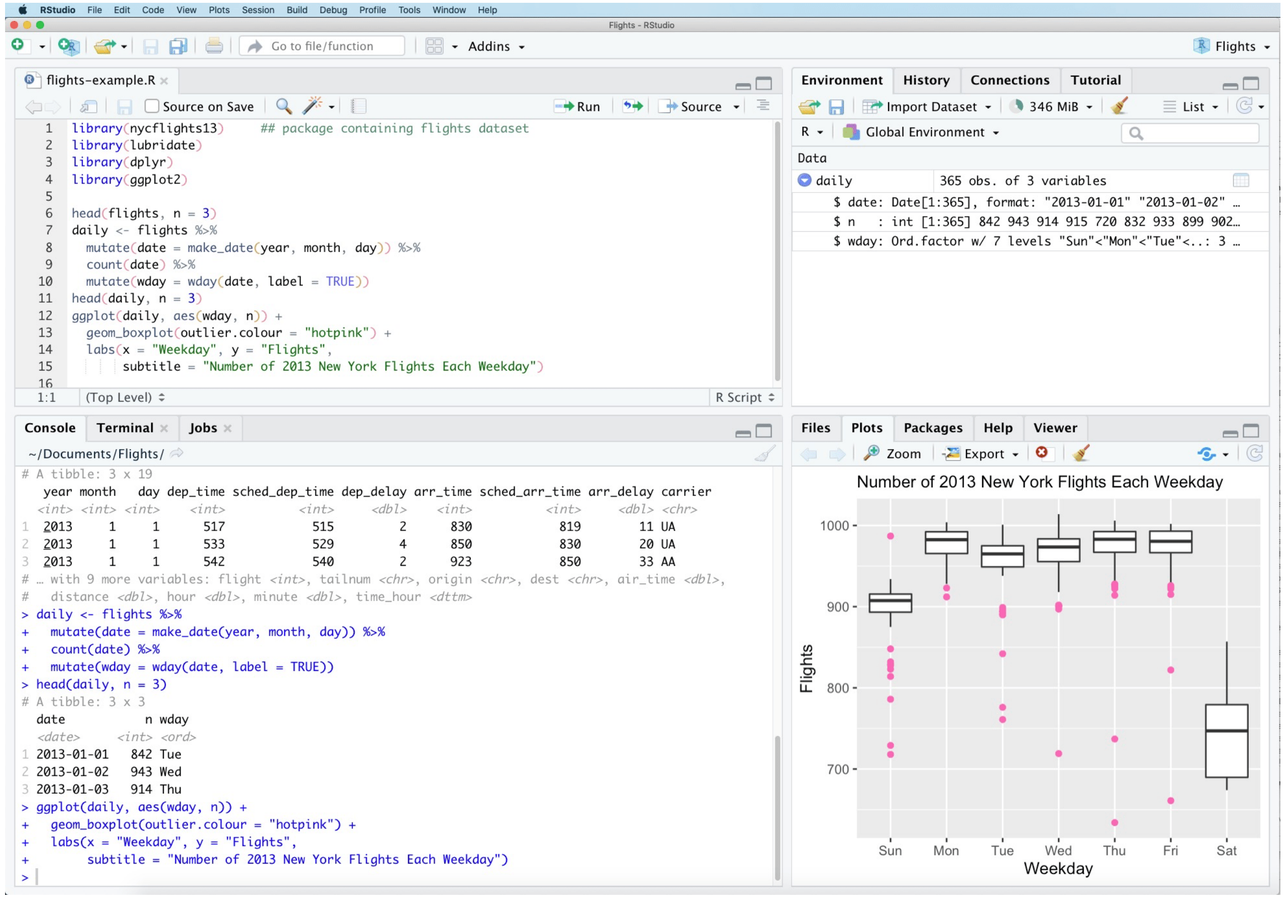
- Download RStudio for free.



Install RStudio

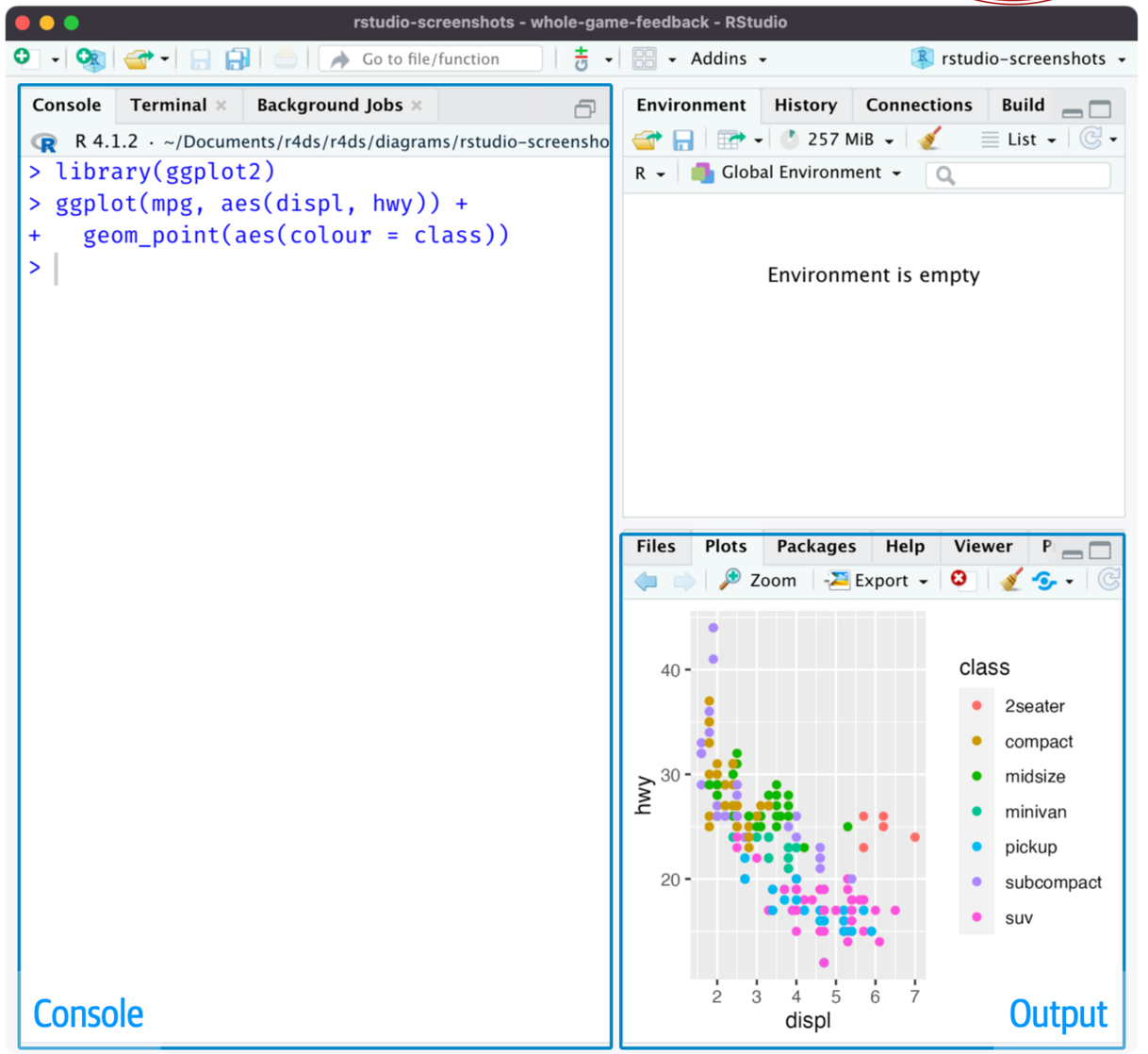
How to install it ?

- Download RStudio for free.
- Use Studio on Posit Cloud.



Install RStudio

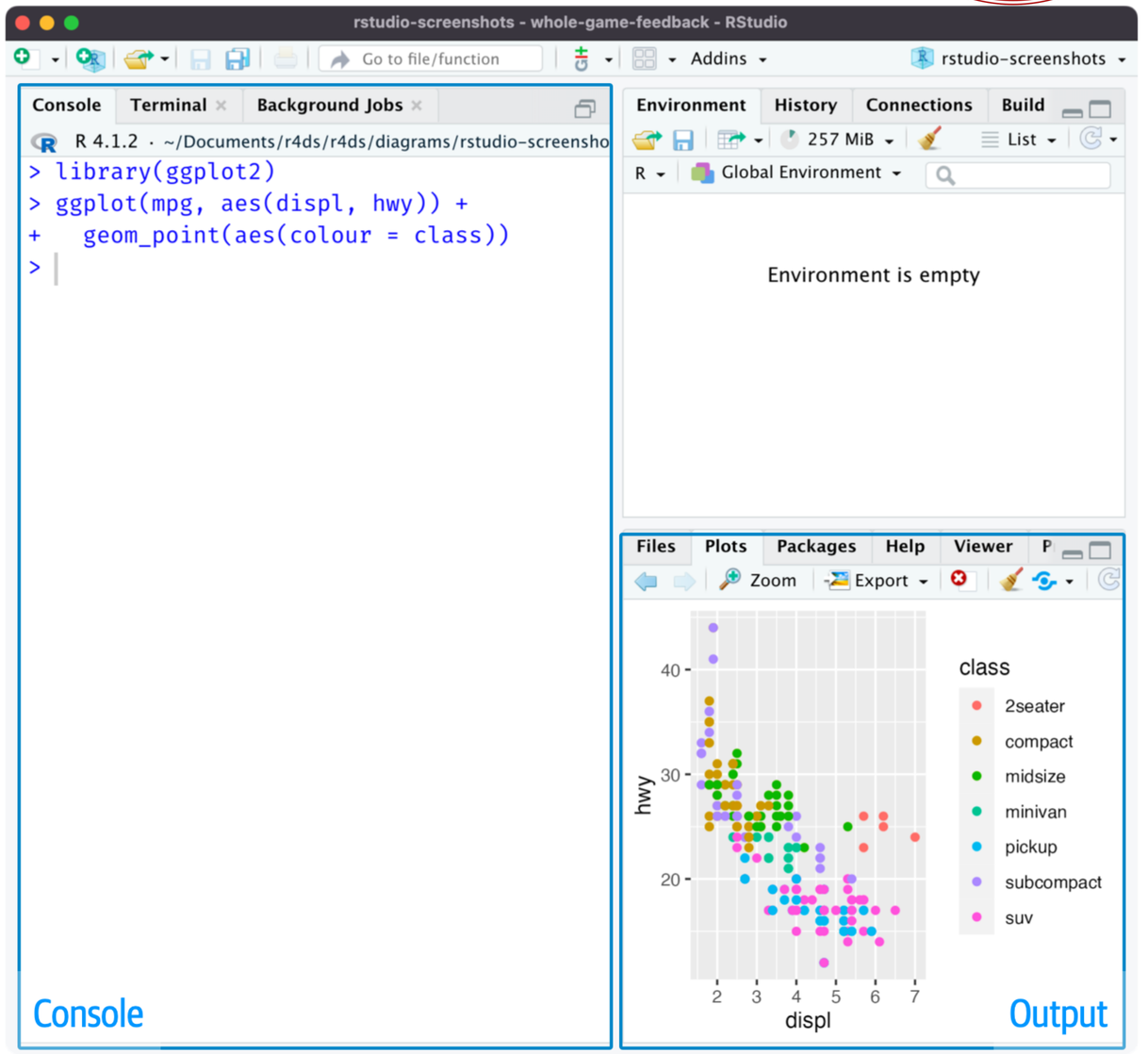
Interface



Install RStudio

Interface

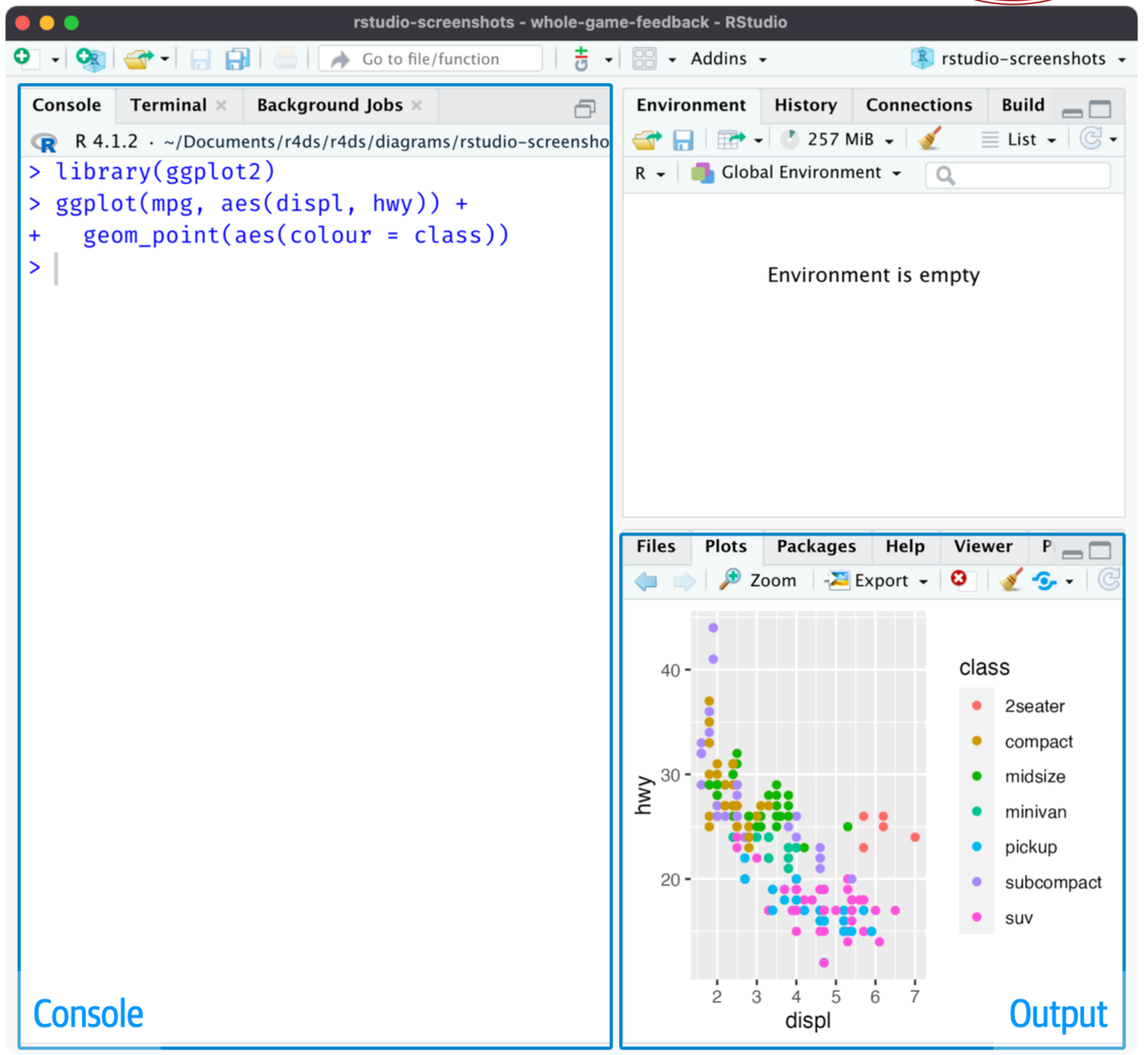
- Two key regions in the interface: the console pane and the output pane.



Install RStudio

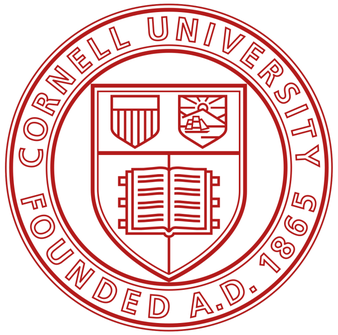
Interface

- Two key regions in the interface: the console pane and the output pane.
- You type the R code in the console pane and press enter to run it.



R packages

What is it ?




```
install.packages("<package name>")
```

R packages

What is it ?

- An R package is a collection of functions, data, and documentation that extends the capabilities of base R.

```
install.packages("<package name>")
```

R packages

What is it ?

- An R package is a collection of functions, data, and documentation that extends the capabilities of base R.
- Using packages is key to the successful use of R.

```
install.packages("<package name>")
```

R packages

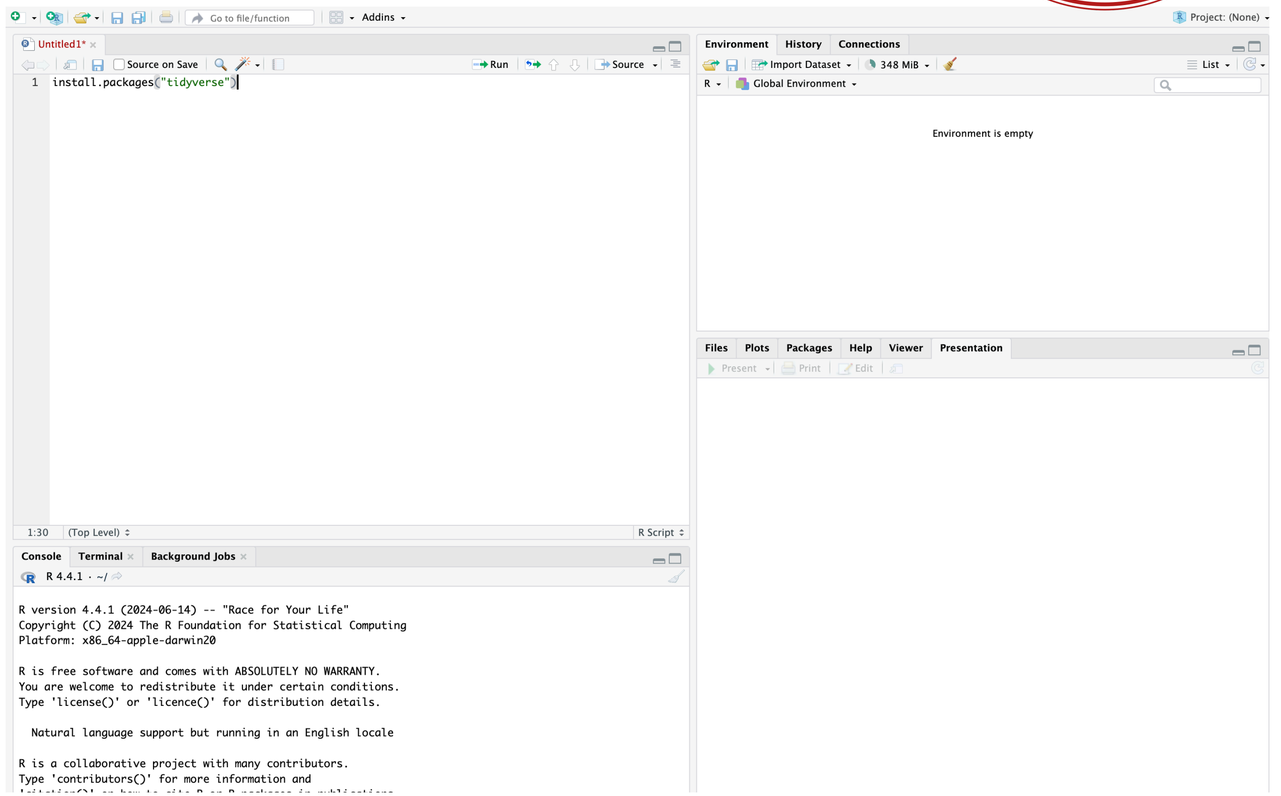
What is it ?

- An R package is a collection of functions, data, and documentation that extends the capabilities of base R.
- Using packages is key to the successful use of R.
- You can install the complete tidyverse package with a single line of code.

```
install.packages("<package name>")
```

R packages

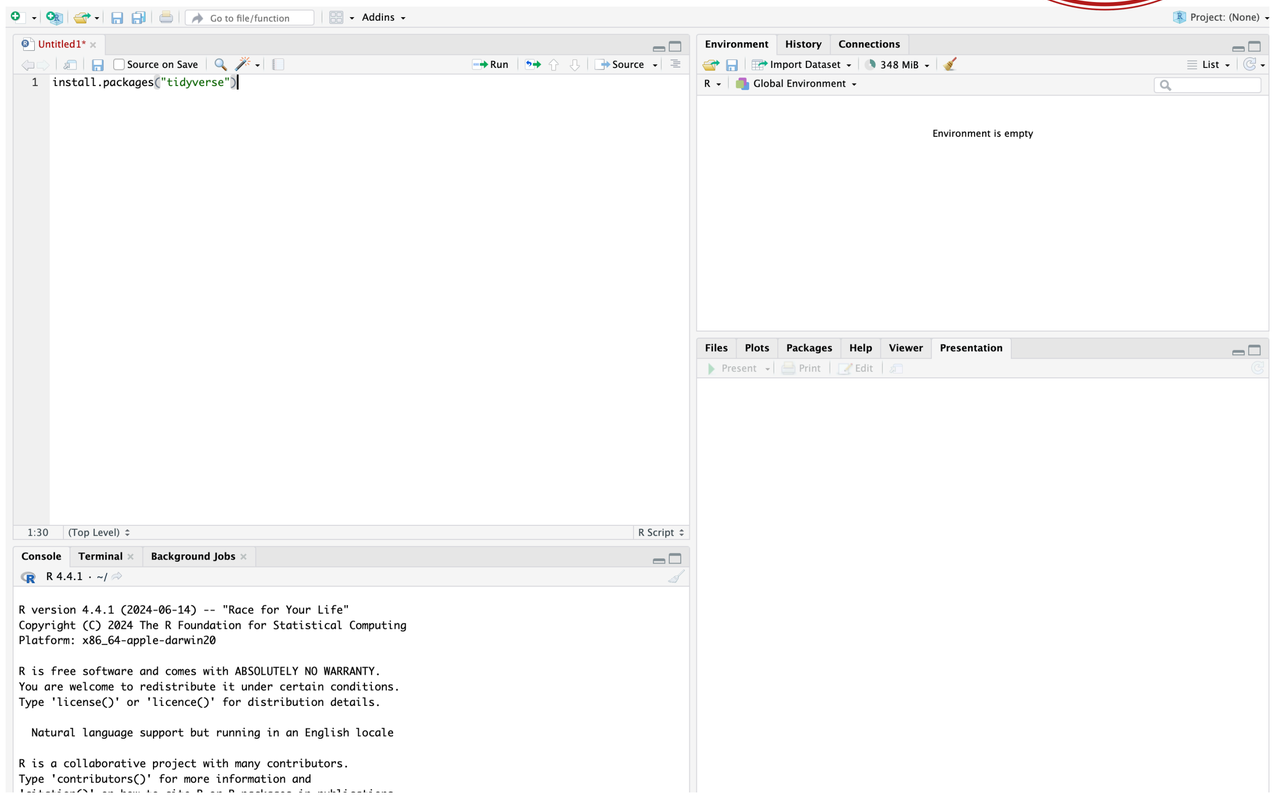
Installing packages



R packages

Installing packages

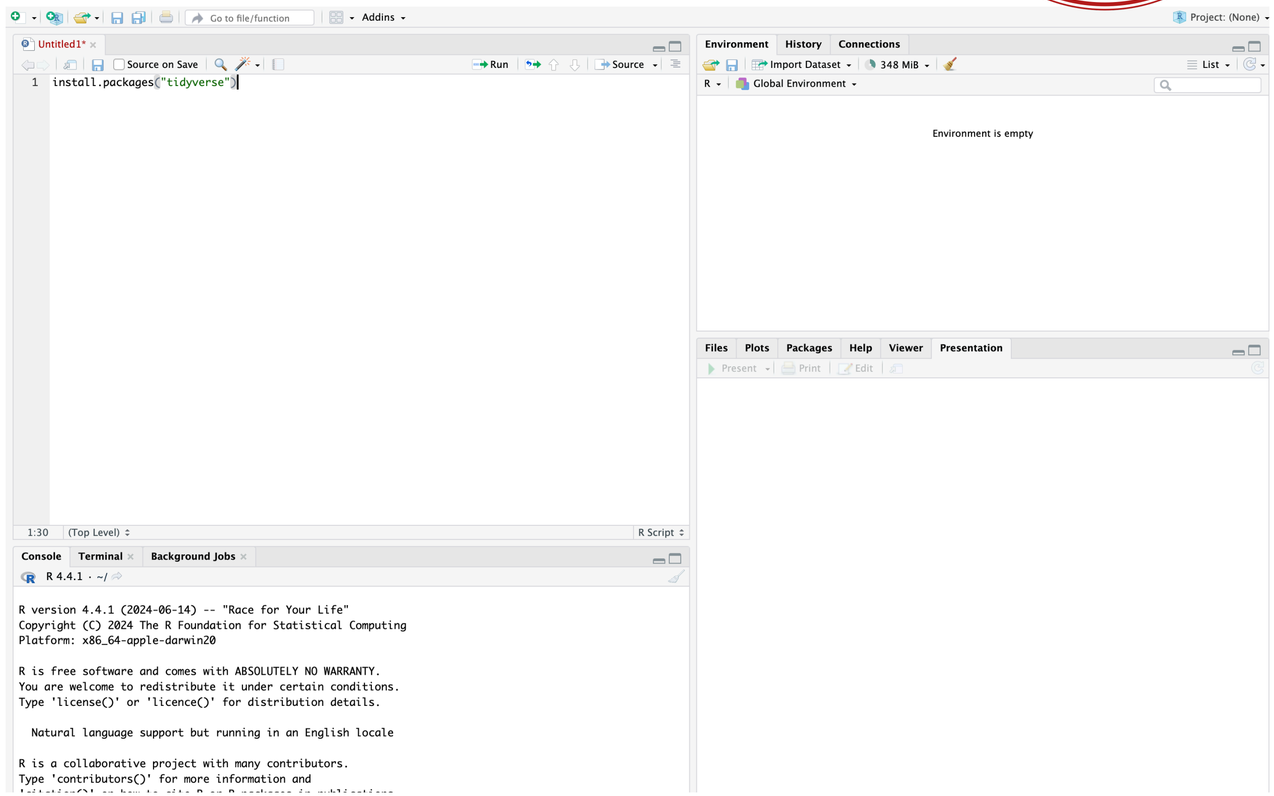
- Open a new R script.



R packages

Installing packages

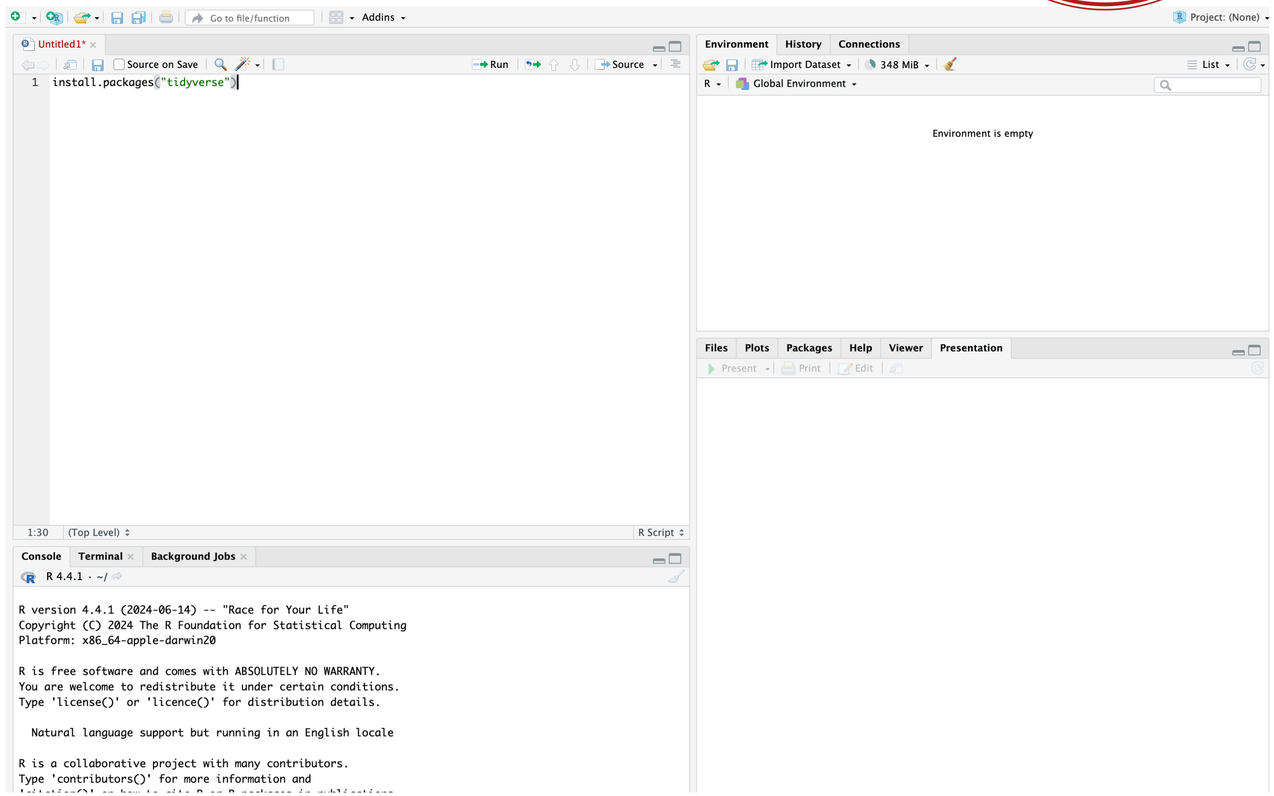
- Open a new R script.
- Type the install.packages line of code.



R packages

Installing packages

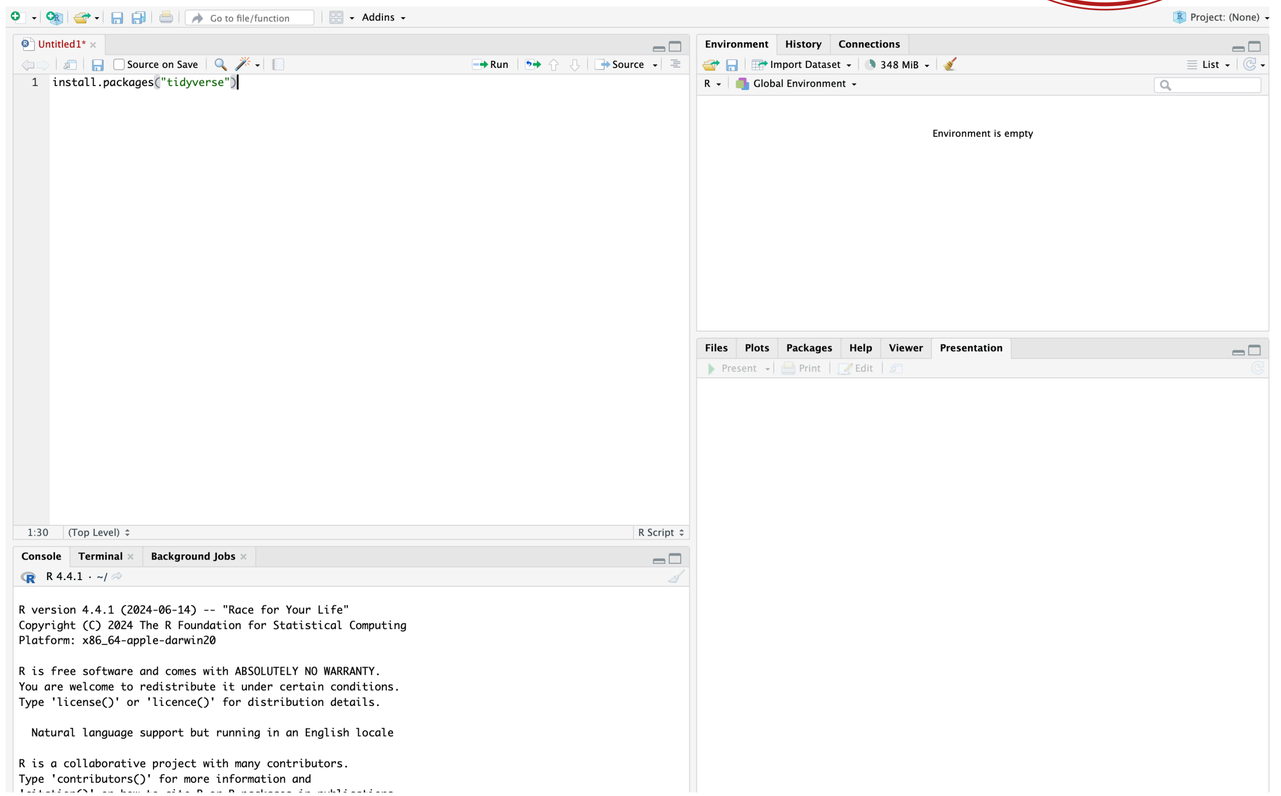
- Open a new R script.
- Type the install.packages line of code.
- Run it.



R packages

Installing packages

- Open a new R script.
- Type the install.packages line of code.
- Run it.
- R will download the packages from CRAN and install them on your computer.



Go to file/function | Addins | Project: (None)

Untitled1*

Source on Save | Run | Source

```
1  install.packages("tidyverse")
2
```

2:1 (Top Level) R Script

Environment | History | Connections

Import Dataset | 369 MiB | List

R | Global Environment

Environment is empty

Files | Plots | Packages | Help | Viewer | Presentation

Present | Print | Edit

Console | Terminal | Background Jobs

R 4.4.1 · ~/

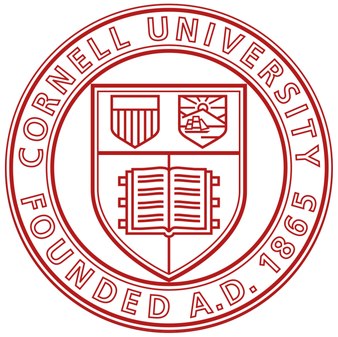
```
Content type 'application/x-gzip' length 527797 bytes (515 KB)
==================================================
downloaded 515 KB

trying URL 'https://cran.rstudio.com/bin/macosx/big-sur-x86_64/contrib/4.4/tidyverse_2.0.0.tgz'
Content type 'application/x-gzip' length 428830 bytes (418 KB)
==================================================
downloaded 418 KB


The downloaded binary packages are in
        /var/folders/2r/9nmfdty958xcz1d179s1fxgw0000gn/T//RtmpKNCkL4/downloaded_packages
>
```

R packages

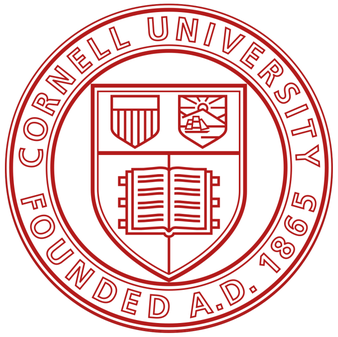
Loading packages




```
library(<package name>)
```

R packages

Loading packages



- Installing a package doesn't immediately place its functions at your fingertips.

```
library(<package name>)
```

R packages

Loading packages

- Installing a package doesn't immediately place its functions at your fingertips.
- It just places them on your computer.

```
library(<package name>)
```

R packages

Loading packages

- Installing a package doesn't immediately place its functions at your fingertips.
- It just places them on your computer.
- To use an R package, you next have to load it in your R session with the command library.

```
library(<package name>)
```

R packages

Loading packages

- Installing a package doesn't immediately place its functions at your fingertips.
- It just places them on your computer.
- To use an R package, you next have to load it in your R session with the command library.
- Notice that the quotation marks have disappeared.

```
library(<package name>)
```

Updating R

R language

Updating R

R language

- New versions of R are released several times a year.

Updating R

R language

- New versions of R are released several times a year.
- The easiest way to stay current with R is to periodically check the CRAN website.

Updating R

R language

- New versions of R are released several times a year.
- The easiest way to stay current with R is to periodically check the CRAN website.
- The process is the same as when you first installed R.

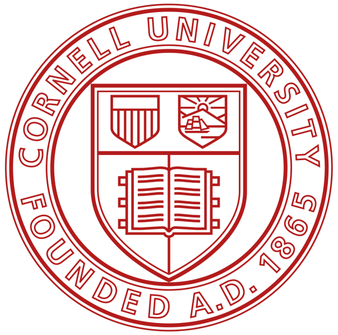
Updating R

R language

- New versions of R are released several times a year.
- The easiest way to stay current with R is to periodically check the CRAN website.
- The process is the same as when you first installed R.
- Updating to the current version of R is a good place to start if you ever encounter a bug that you can't explain.

Updating R

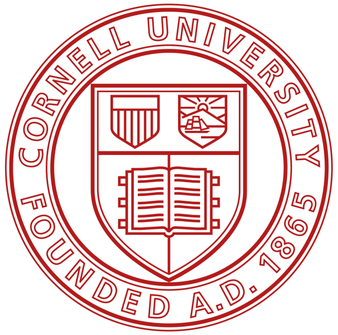
R packages




```
update.packages(c("ggplot2", "reshape2", "dplyr"))
```

Updating R

R packages

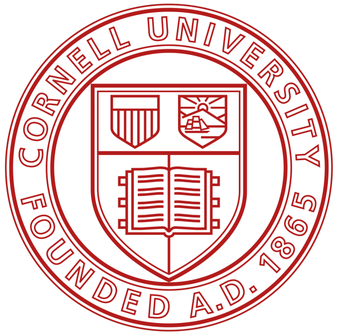


- Package authors occasionally release new versions of their packages to add functions, fix bugs, or improve performance.

```
update.packages(c("ggplot2", "reshape2", "dplyr"))
```

Updating R

R packages

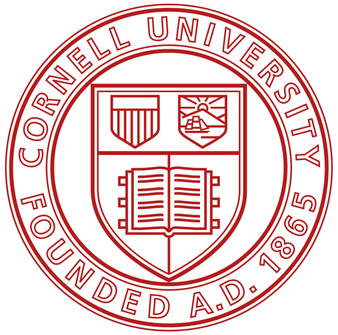


- Package authors occasionally release new versions of their packages to add functions, fix bugs, or improve performance.
- The `update.packages` command checks whether you have the most current version of a package and installs the most current version if you do not.

```
update.packages(c("ggplot2", "reshape2", "dplyr"))
```

Updating R

R packages



- Package authors occasionally release new versions of their packages to add functions, fix bugs, or improve performance.
- The `update.packages` command checks whether you have the most current version of a package and installs the most current version if you do not.
- You should start a new R session after updating packages.

```
update.packages(c("ggplot2", "reshape2", "dplyr"))
```

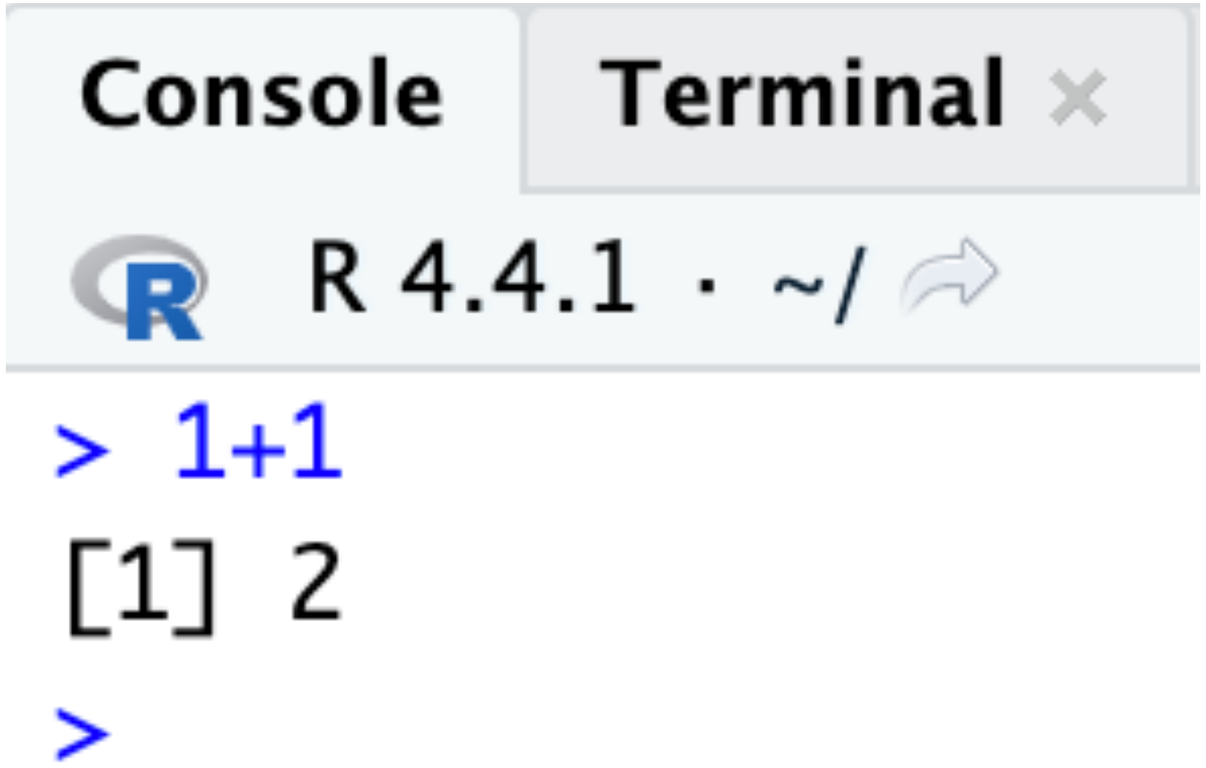
User guide

Comprehensive overview

- If you'd like a comprehensive overview of all of RStudio's feature: https://docs.posit.co/ide/use.

Running R code

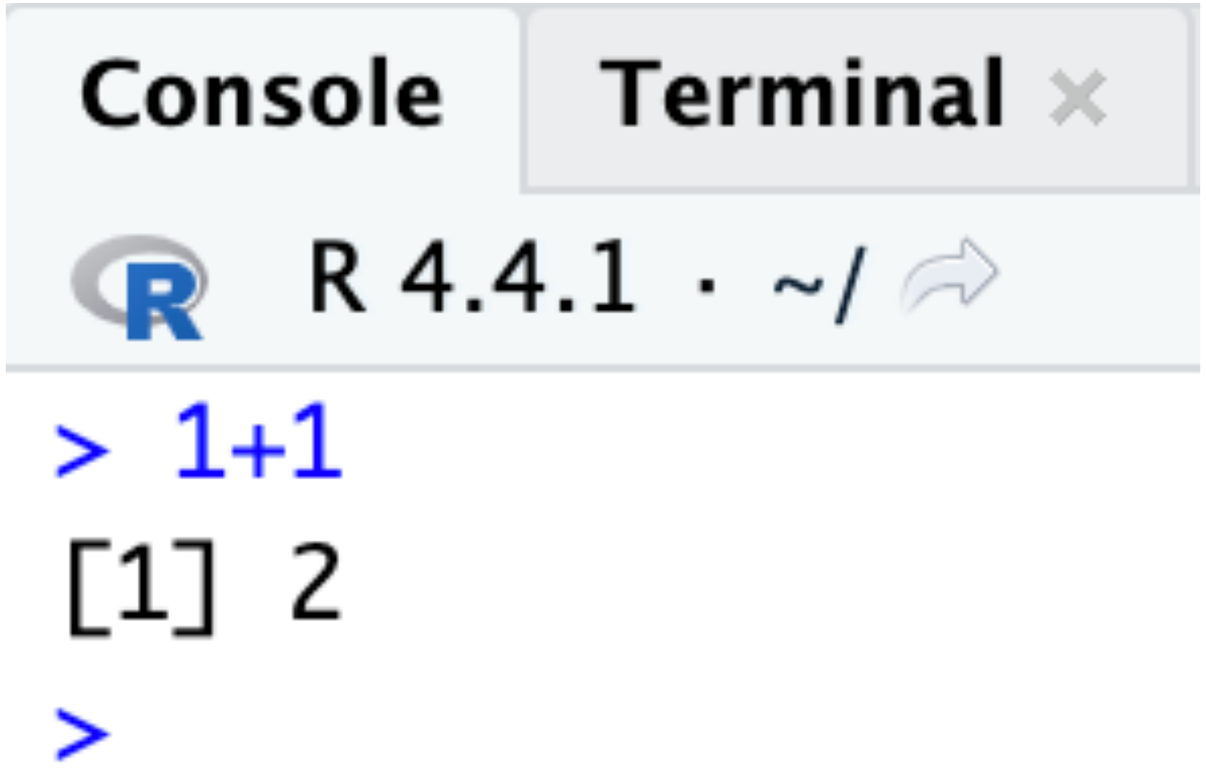
Basic command



Running R code

Basic command

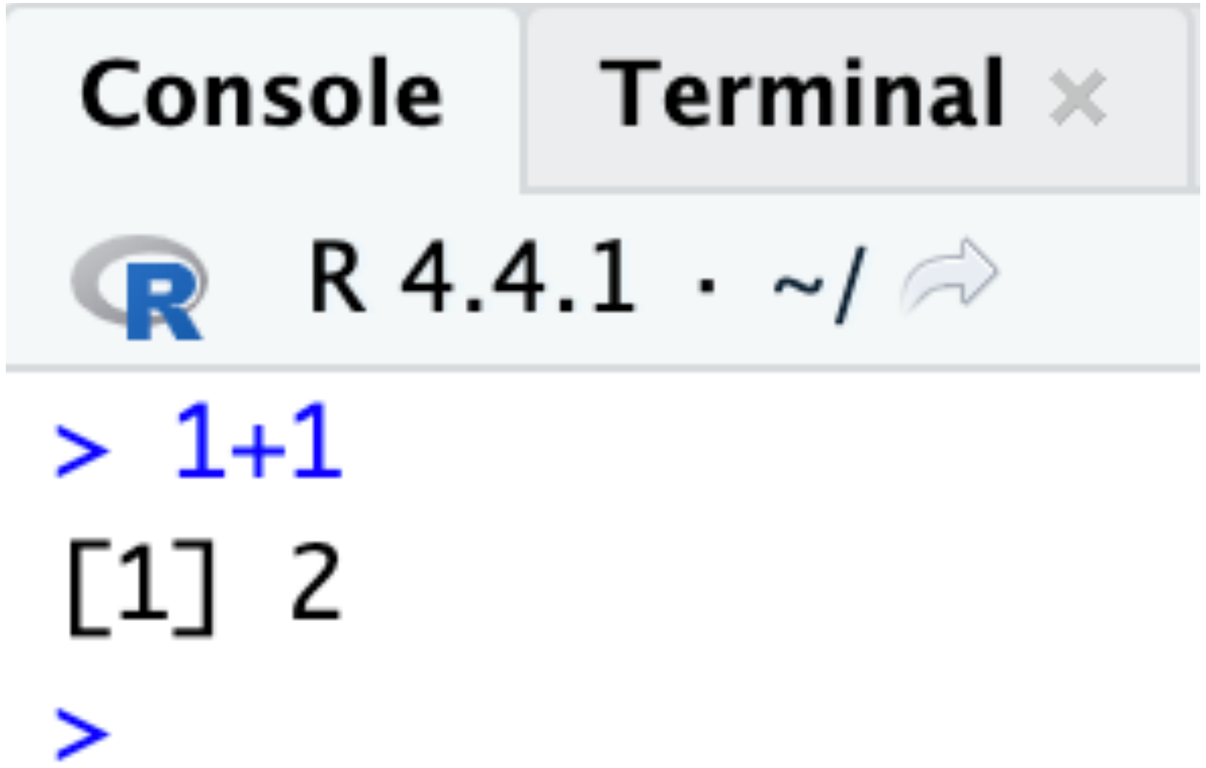
- Open RStudio.



Running R code

Basic command

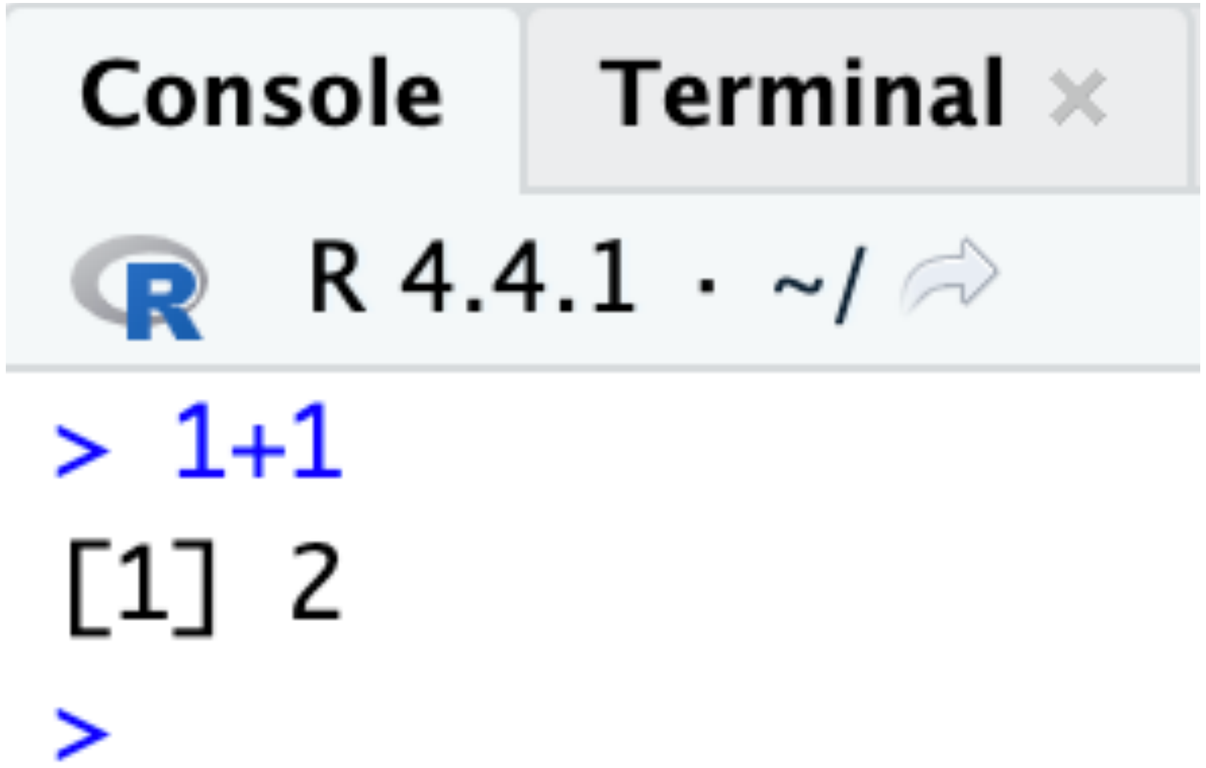
- Open RStudio.
- Type R code into the bottom line of the RStudio console pane.



Running R code

Basic command

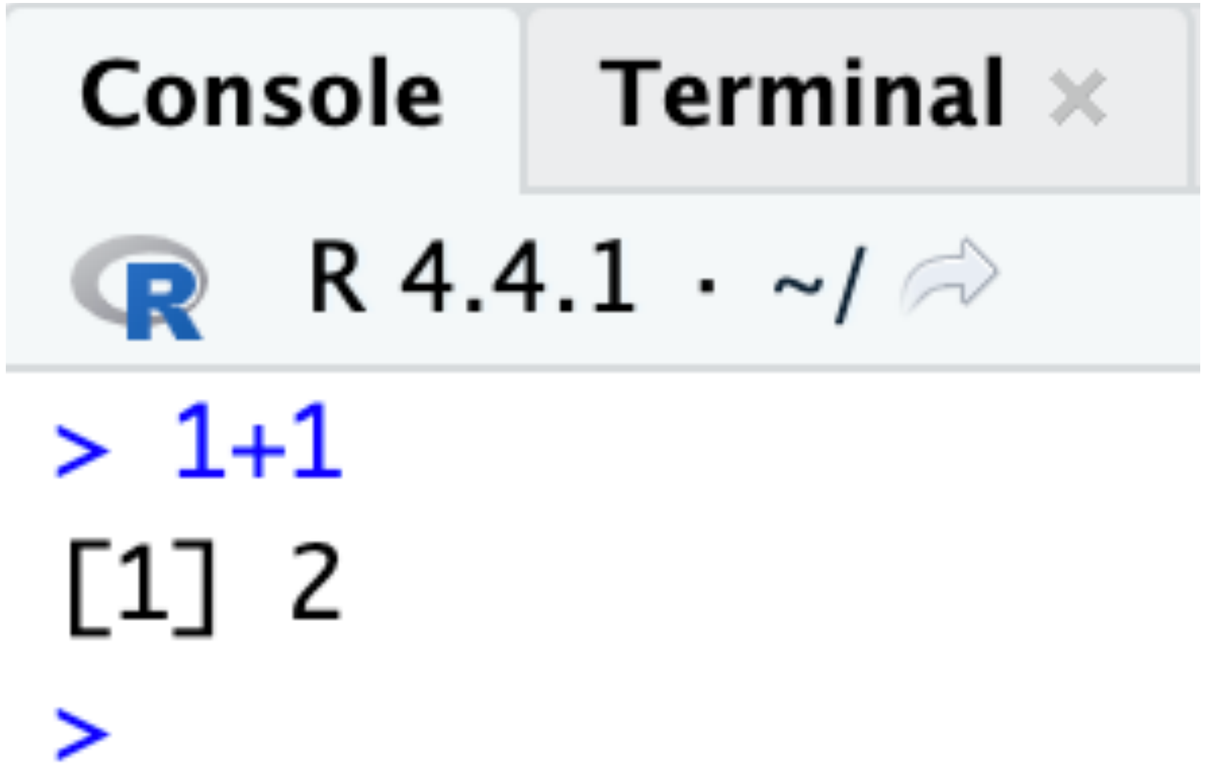
- Open RStudio.
- Type R code into the bottom line of the RStudio console pane.
- Click Enter.



Running R code

Basic command

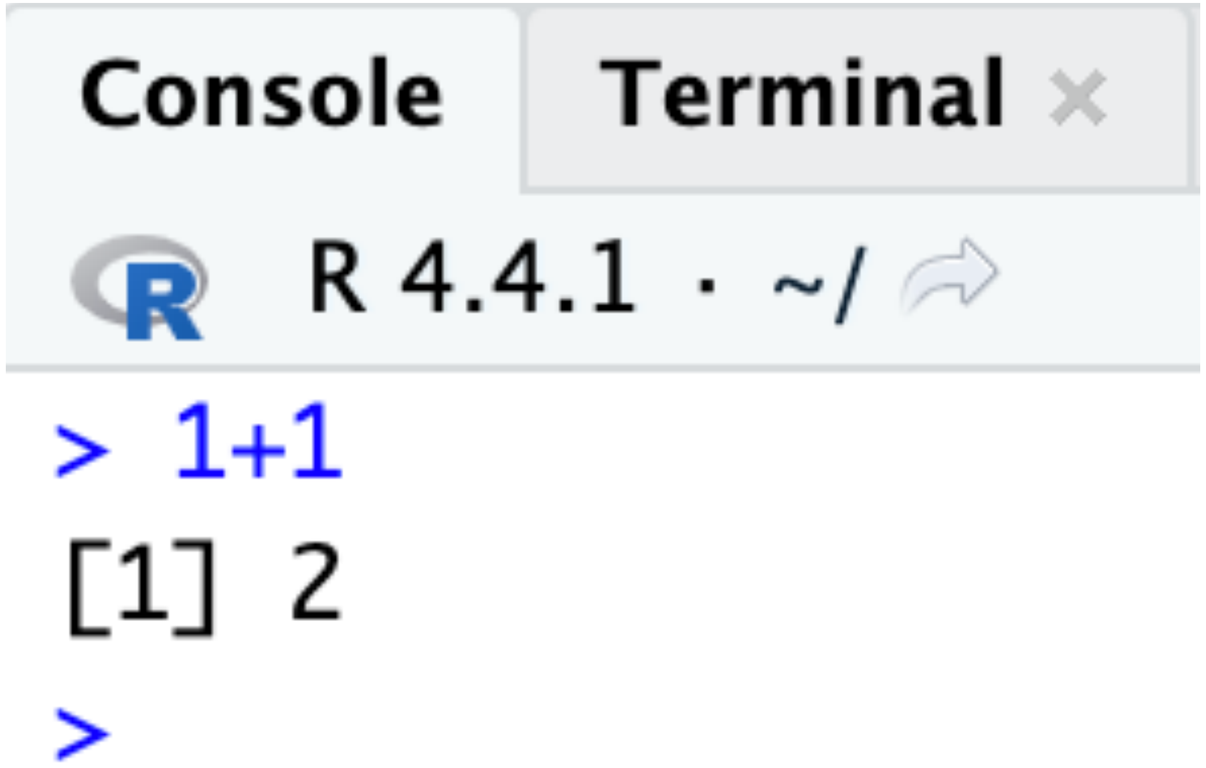
- Open RStudio.
- Type R code into the bottom line of the RStudio console pane.
- Click Enter.
- The code you type is called a *command*.



Running R code

Basic command

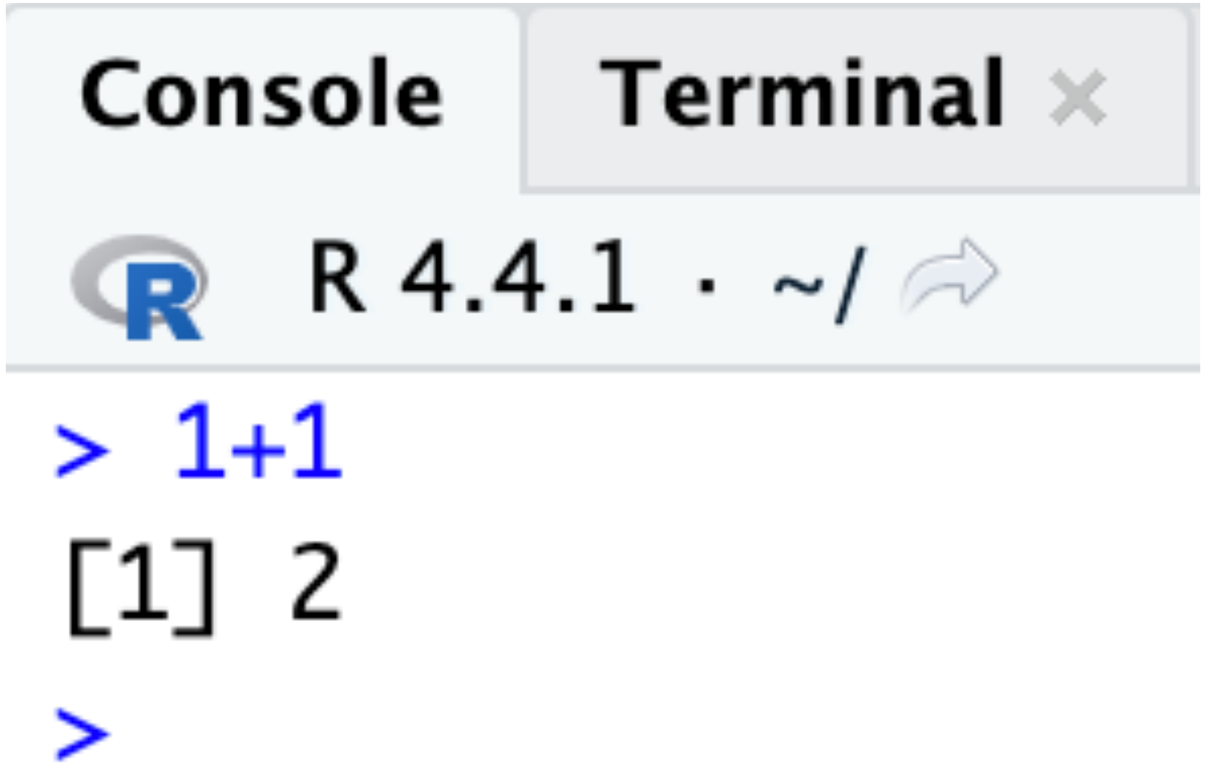
- Open RStudio.
- Type R code into the bottom line of the RStudio console pane.
- Click Enter.
- The code you type is called a *command*.
- The line you type it into is called the *command line*.



Running R code

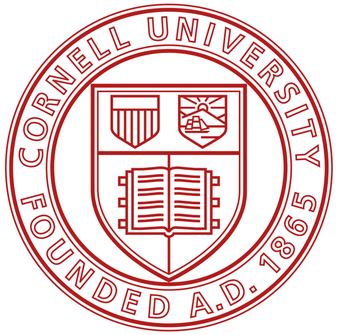
Basic command

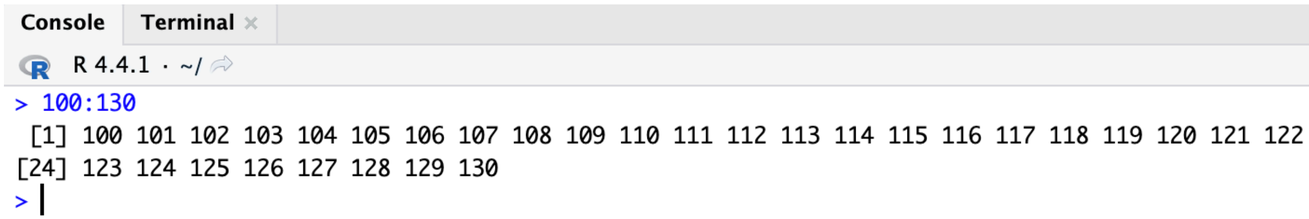
- Open RStudio.
- Type R code into the bottom line of the RStudio console pane.
- Click Enter.
- The code you type is called a *command*.
- The line you type it into is called the *command line*.
- For example, if you type `1 + 1` and hit Enter, RStudio will display:



Basics

Sequence of numbers

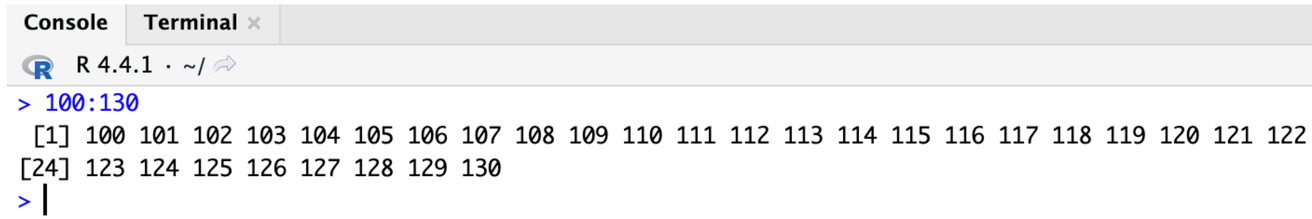




Basics

Sequence of numbers

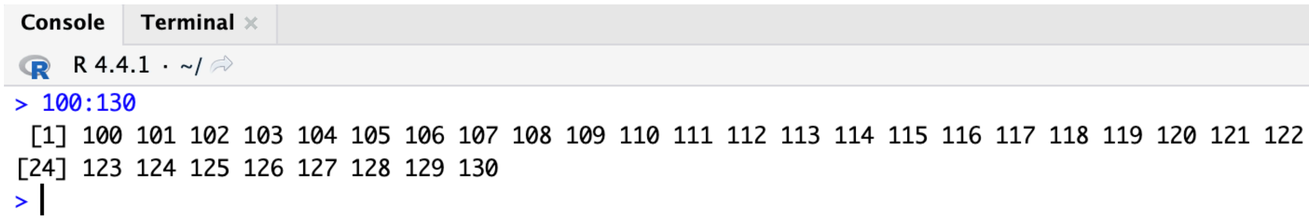
- You'll notice that a `[1]` appears next to your result.



Basics

Sequence of numbers

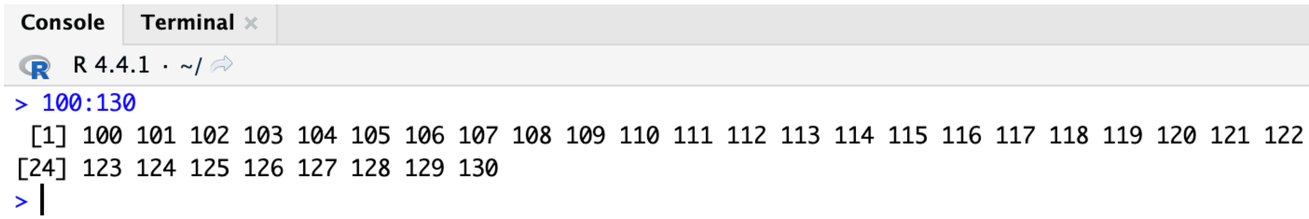
- You'll notice that a `[1]` appears next to your result.
- R is just letting you know that this line begins with the first value in your result.



Basics

Sequence of numbers

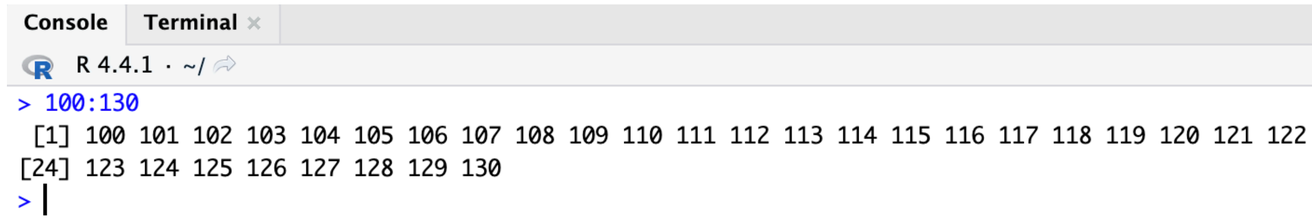
- You'll notice that a `[1]` appears next to your result.
- R is just letting you know that this line begins with the first value in your result.
- Some commands return more than one value, and their results may fill up multiple lines.



Basics

Sequence of numbers

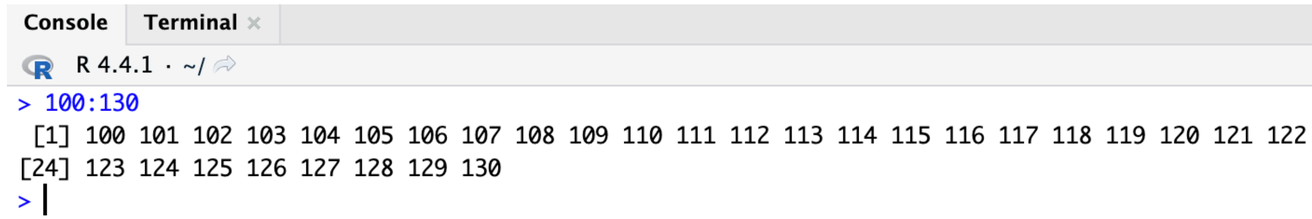
- You'll notice that a `[1]` appears next to your result.
- R is just letting you know that this line begins with the first value in your result.
- Some commands return more than one value, and their results may fill up multiple lines.
- For example, the command `100:130` returns 31 values; it creates a sequence of integers from 100 to 130.



Basics

Sequence of numbers

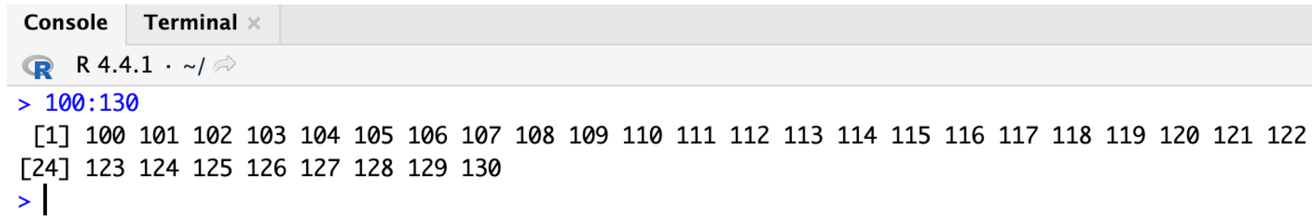
- You'll notice that a `[1]` appears next to your result.
- R is just letting you know that this line begins with the first value in your result.
- Some commands return more than one value, and their results may fill up multiple lines.
- For example, the command `100:130` returns 31 values; it creates a sequence of integers from 100 to 130.
- Notice that new bracketed numbers appear at the start of the second line of output.



Basics

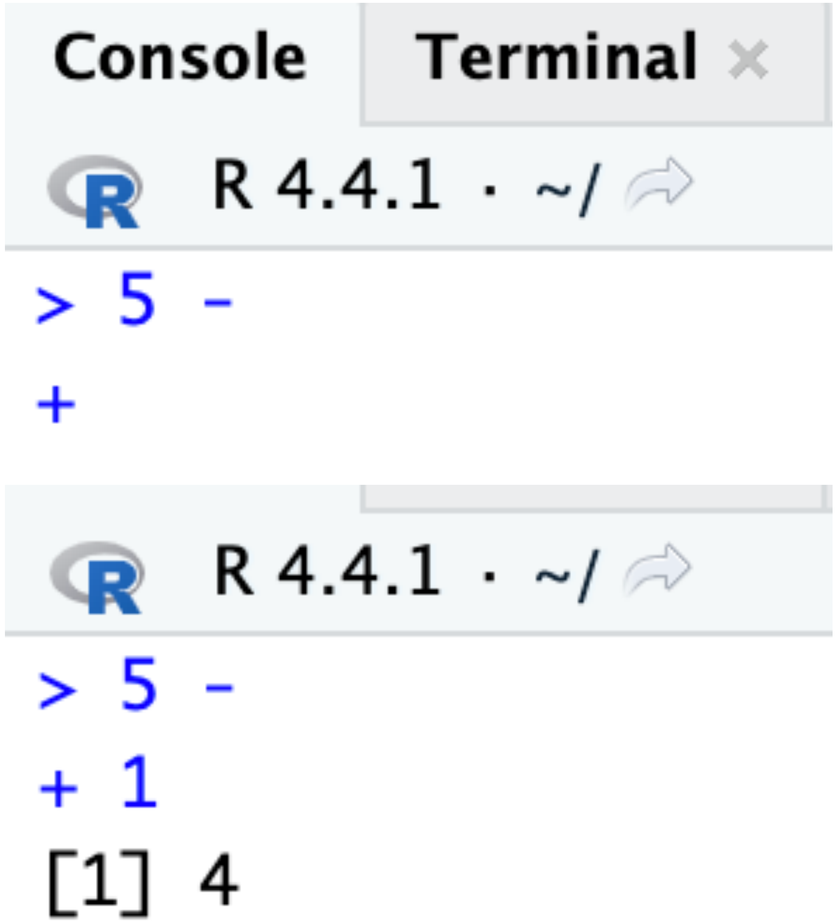
Sequence of numbers

- You'll notice that a `[1]` appears next to your result.
- R is just letting you know that this line begins with the first value in your result.
- Some commands return more than one value, and their results may fill up multiple lines.
- For example, the command `100:130` returns 31 values; it creates a sequence of integers from 100 to 130.
- Notice that new bracketed numbers appear at the start of the second line of output.
- It means that the second line begins with the 24th value in the result.



Basics

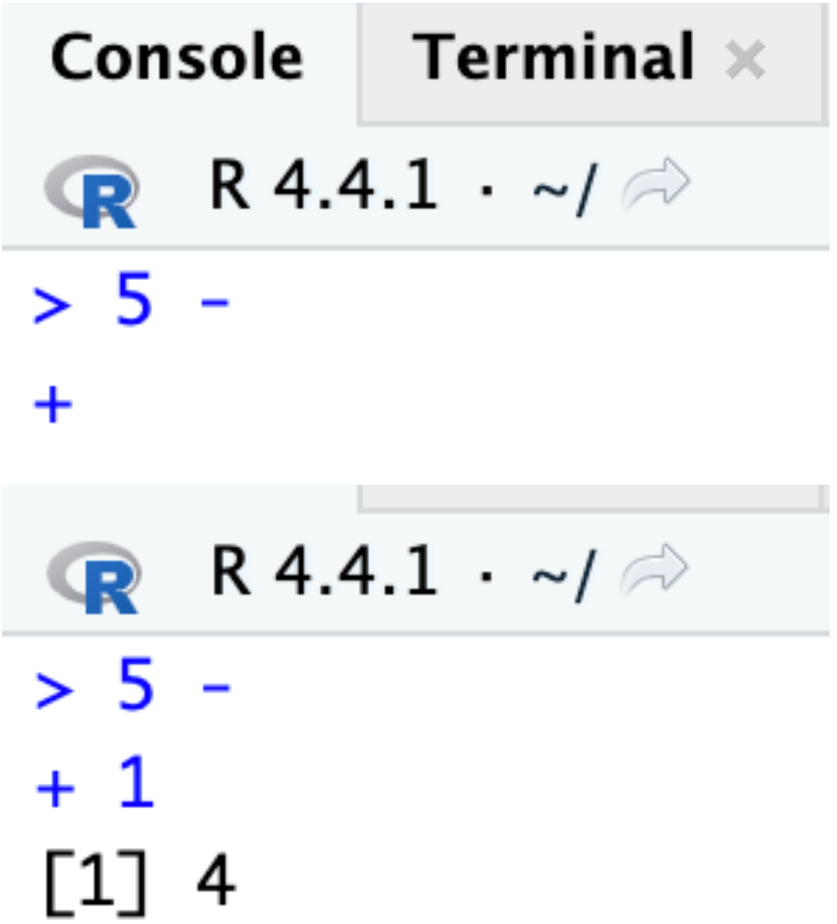
Incomplete command



Basics

Incomplete command

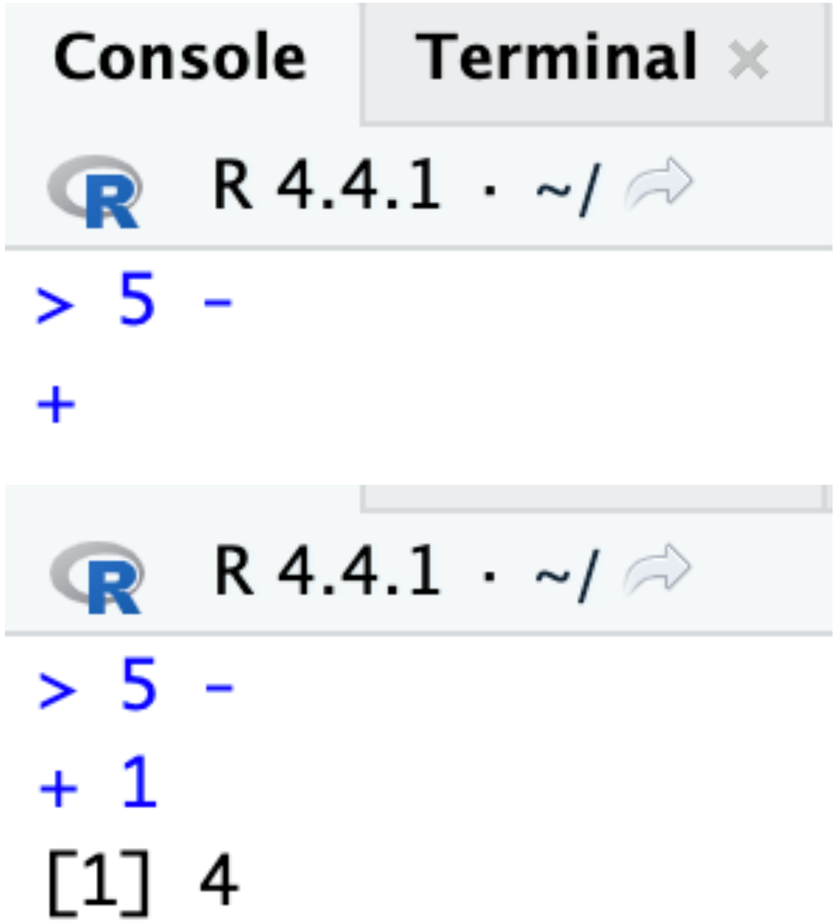
- If you type an incomplete command and press Enter, R will display a `+` prompt.



Basics

Incomplete command

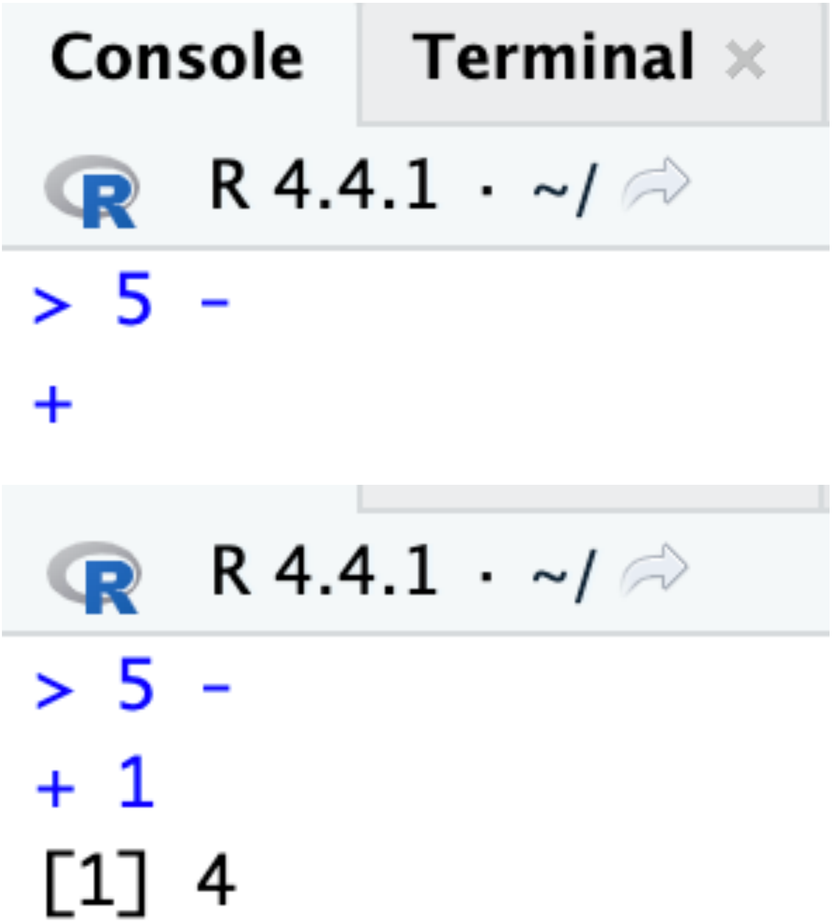
- If you type an incomplete command and press Enter, R will display a `+` prompt.
- It means R is waiting for you to type the rest of your command.



Basics

Incomplete command

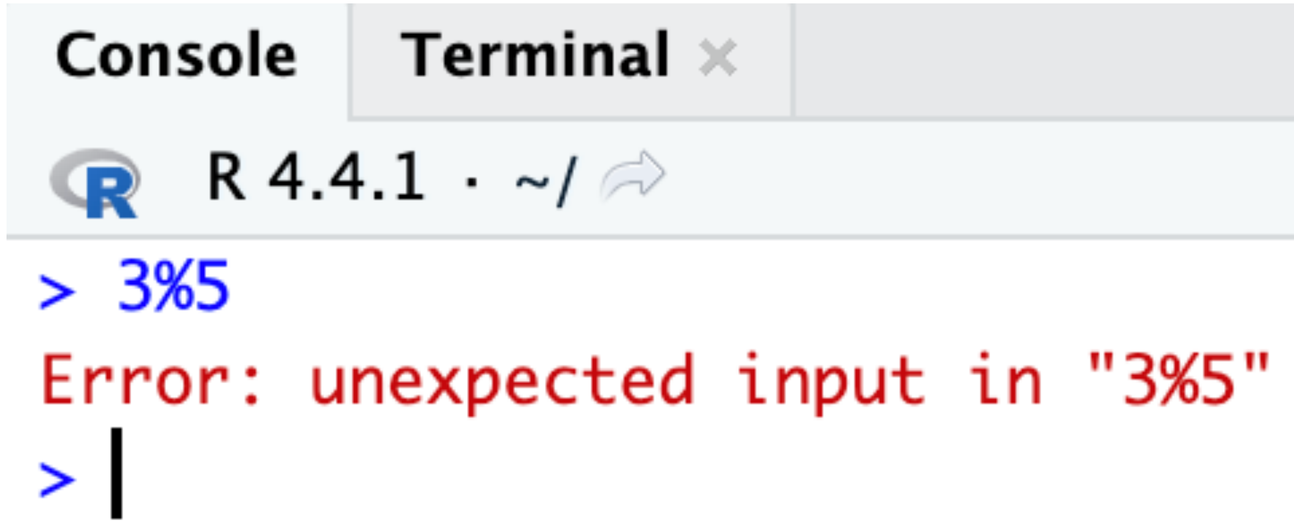
- If you type an incomplete command and press Enter, R will display a `+` prompt.
- It means R is waiting for you to type the rest of your command.
- Either finish the command or hit Escape to start over.



Basics

Error message

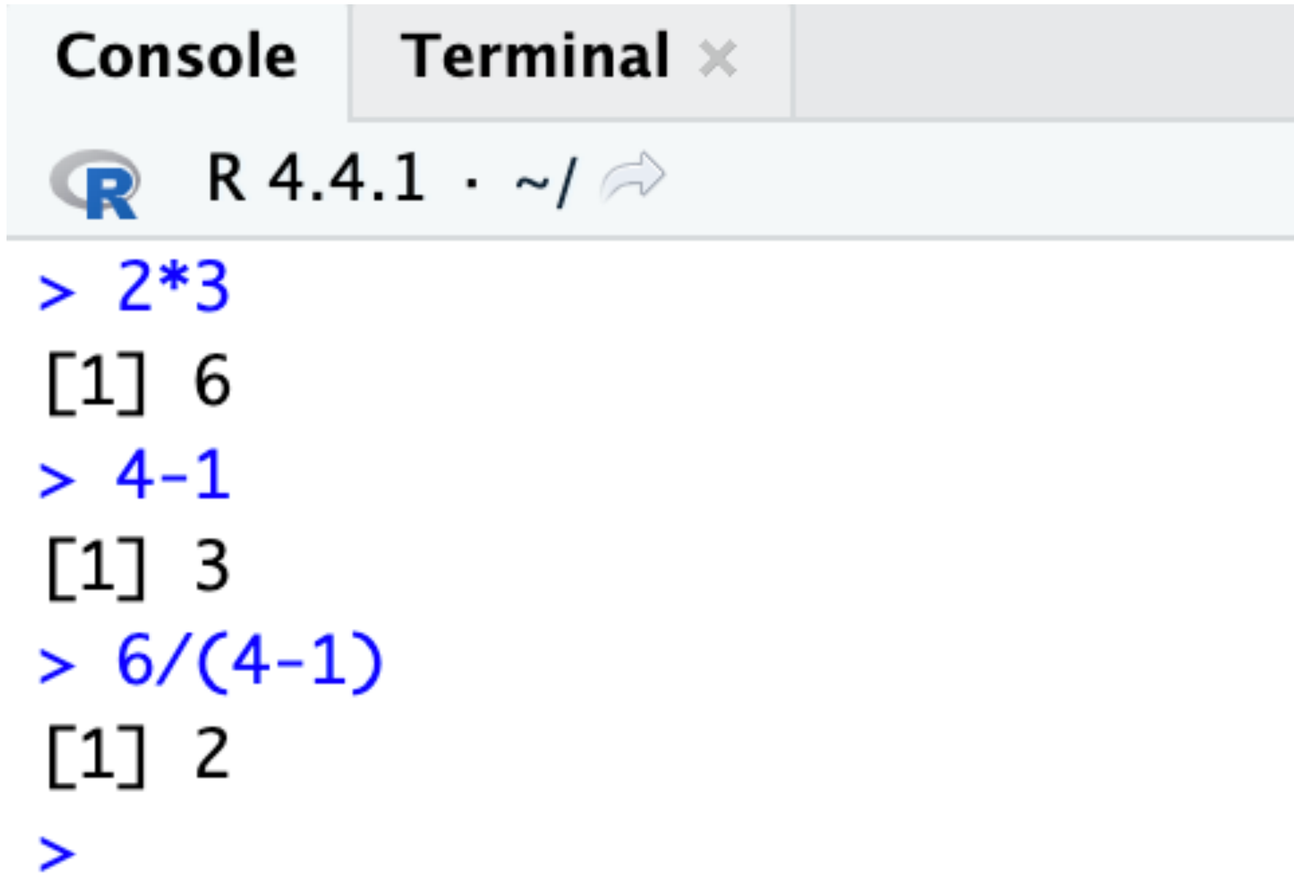
- If you type a command that R doesn't recognize, R will return an error message.



Basics

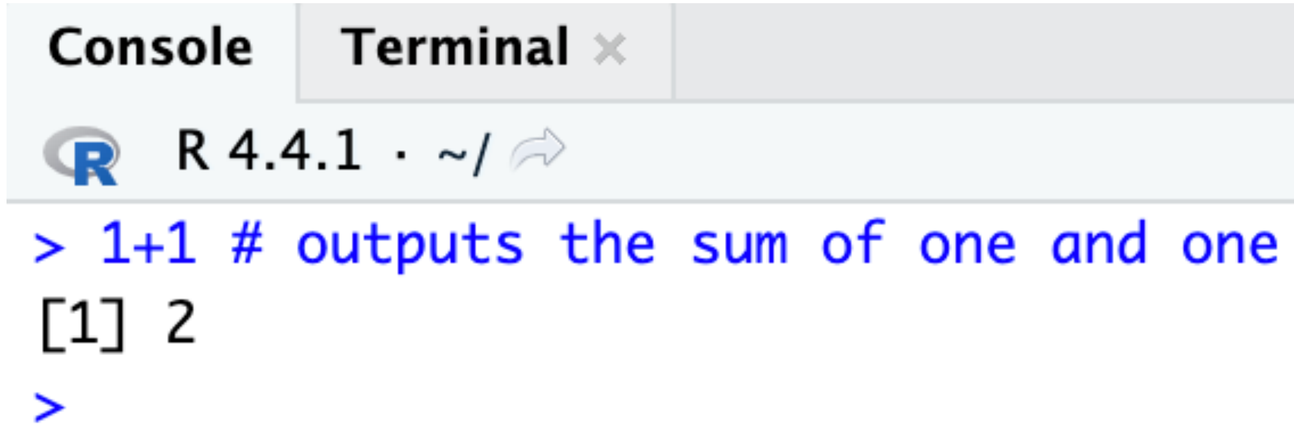
Calculator

- You can easily do anything in R that you would do with a calculator.



Basics

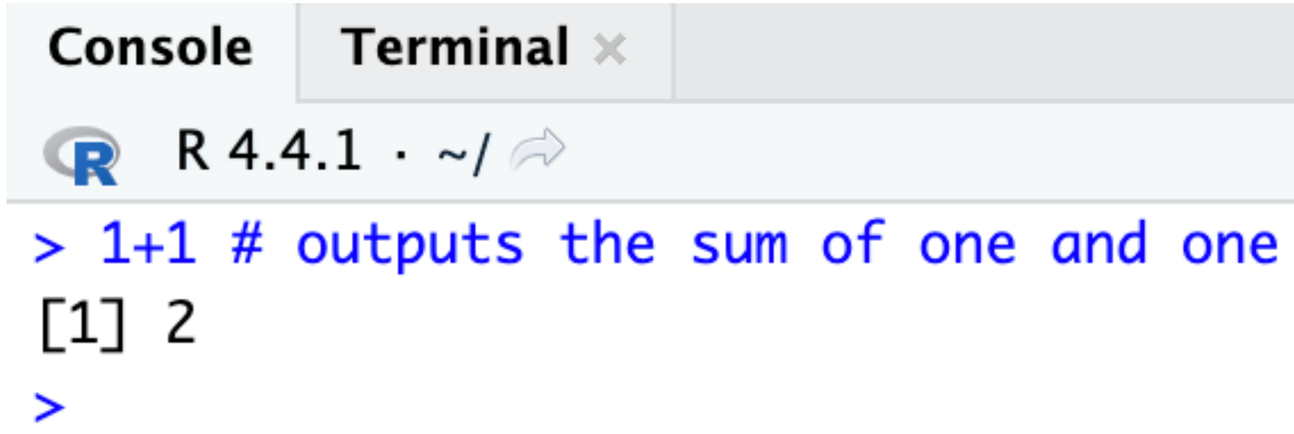
Comments



Basics

Comments

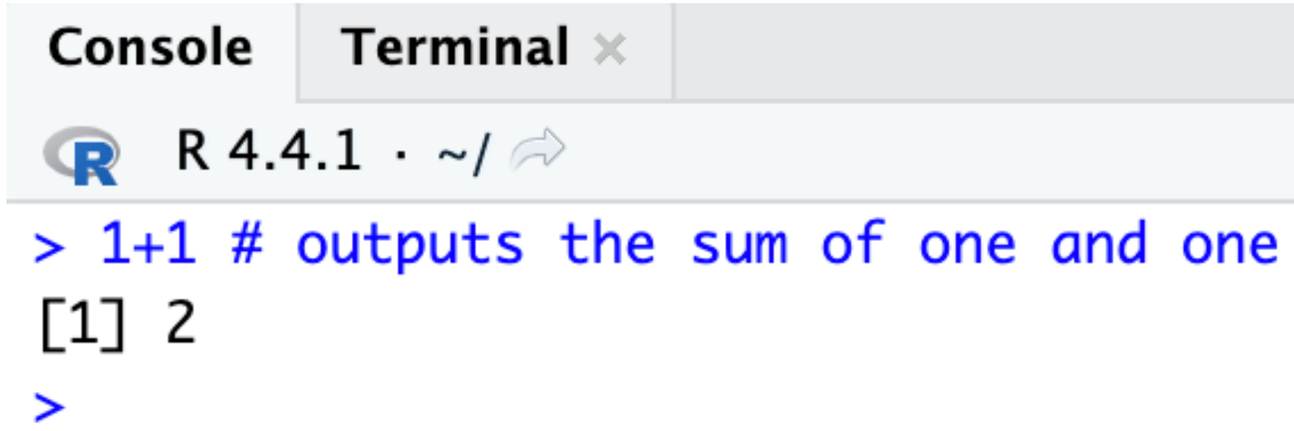
- R treats the hashtag character, #, in a special way.



Basics

Comments

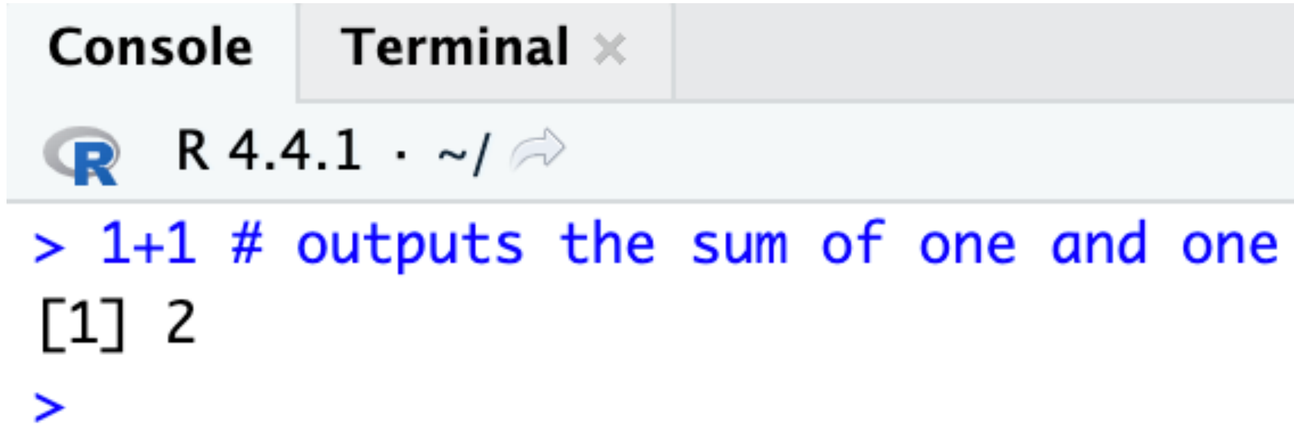
- R treats the hashtag character, #, in a special way.
- R will not run anything that follows a hashtag on a line.



Basics

Comments

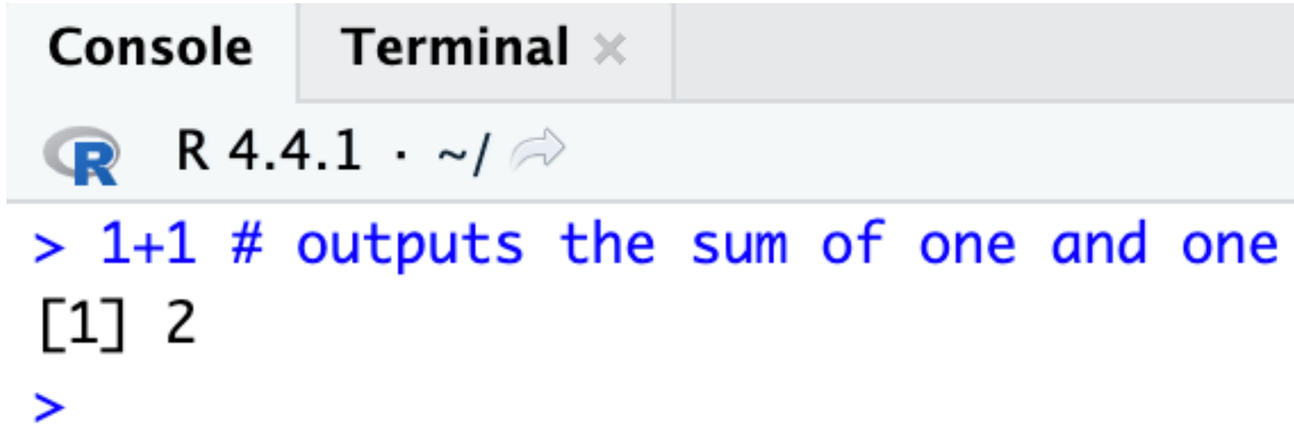
- R treats the hashtag character, #, in a special way.
- R will not run anything that follows a hashtag on a line.
- This makes hashtags very useful for adding comments and annotations to your code.



Basics

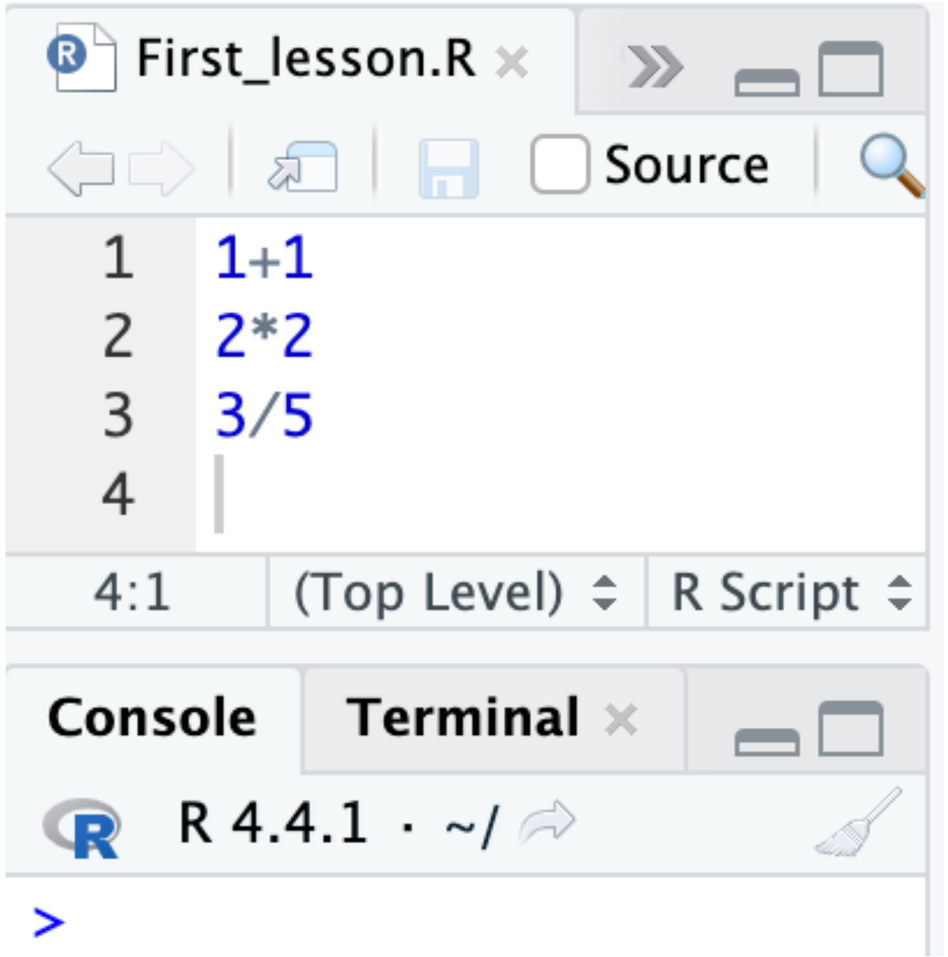
Comments

- R treats the hashtag character, #, in a special way.
- R will not run anything that follows a hashtag on a line.
- This makes hashtags very useful for adding comments and annotations to your code.
- Humans will be able to read the comments, but your computer will pass over them.



Basics

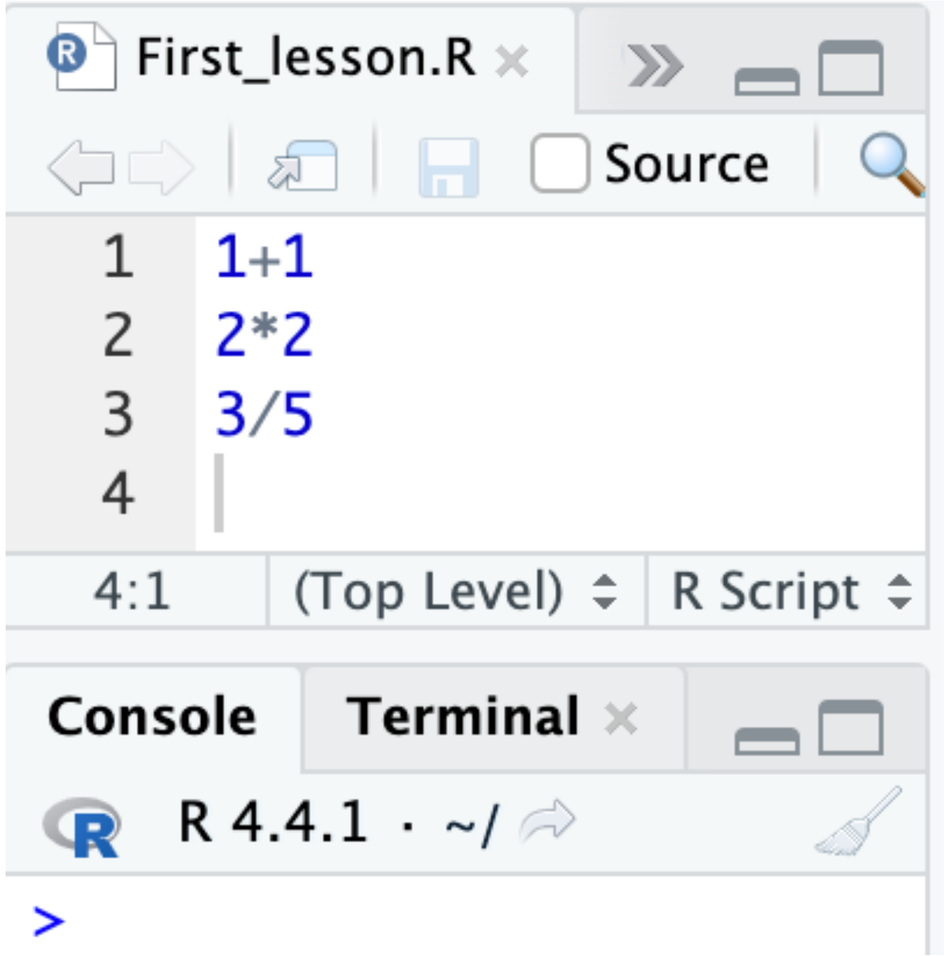
Console vs R. file



Basics

Console vs R. file

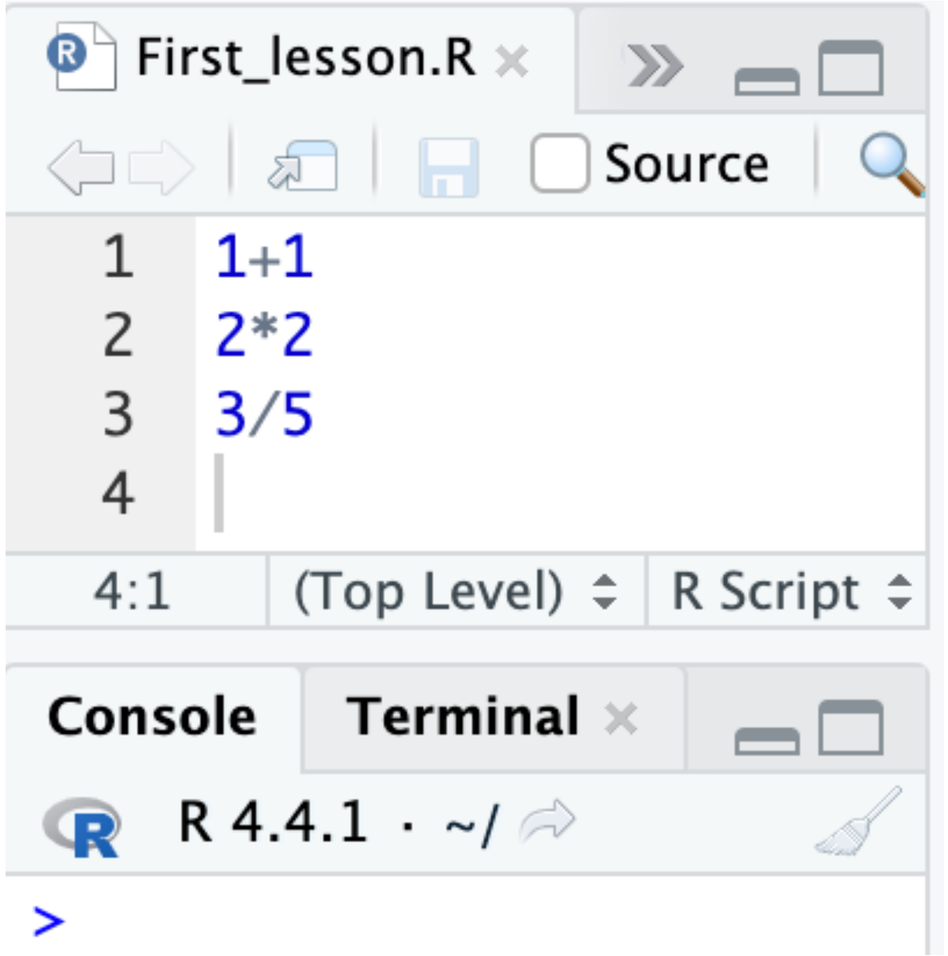
- In RStudio, code can be executed either by writing it directly in the console or by creating and running scripts from a .R file.



Basics

Console vs R. file

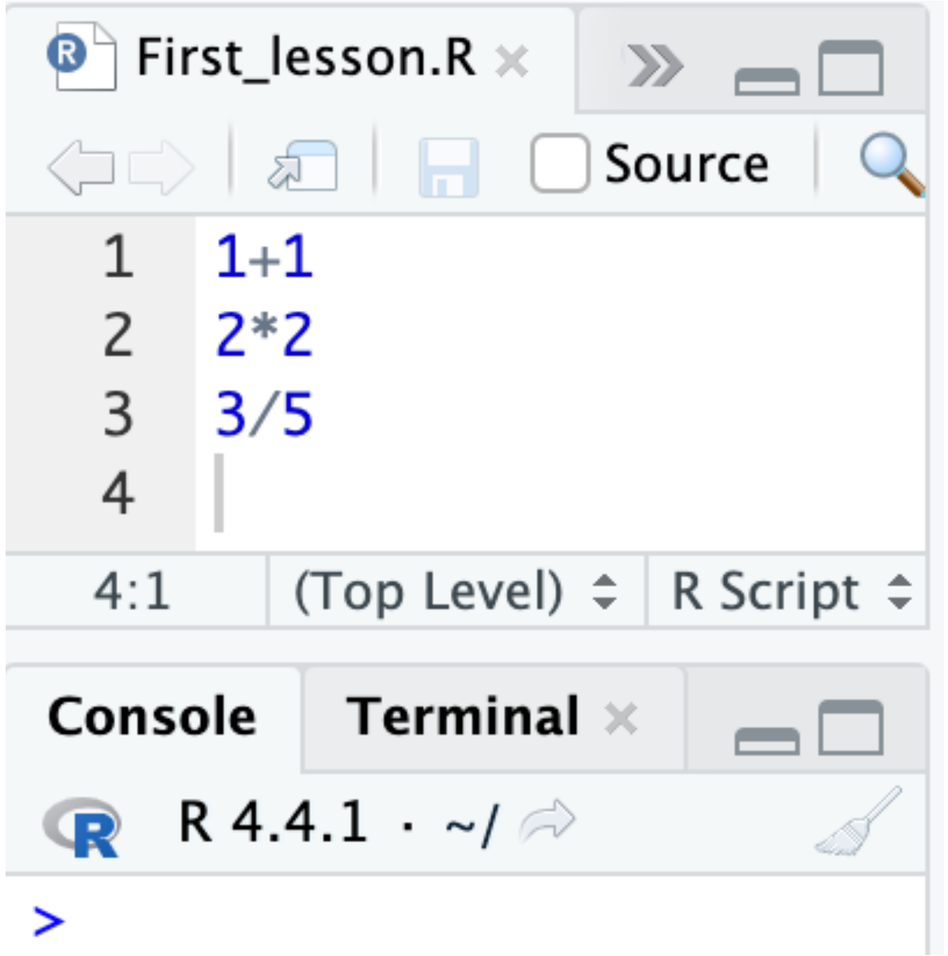
- In RStudio, code can be executed either by writing it directly in the console or by creating and running scripts from a .R file.
- The console in RStudio is an interactive environment where users can write and execute commands line-by-line.



Basics

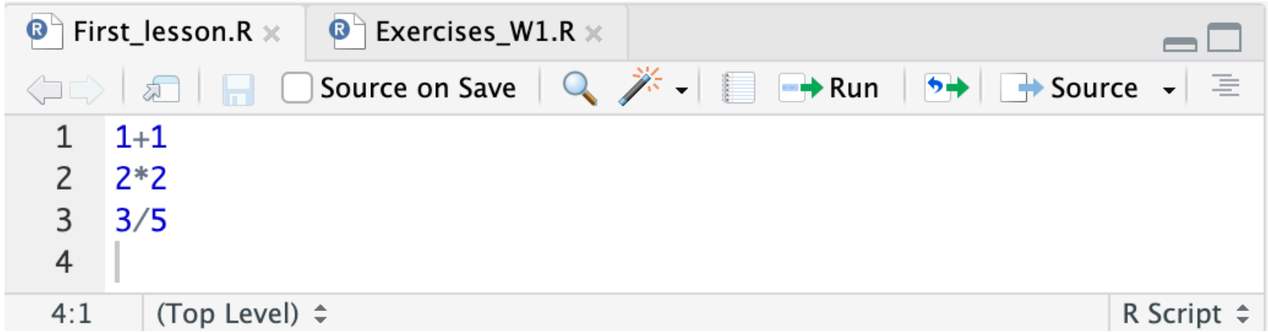
Console vs R. file

- In RStudio, code can be executed either by writing it directly in the console or by creating and running scripts from a .R file.
- The console in RStudio is an interactive environment where users can write and execute commands line-by-line.
- A .R file, on the other hand, is a script file where users can write, edit, and save multiple lines of code.



Basics

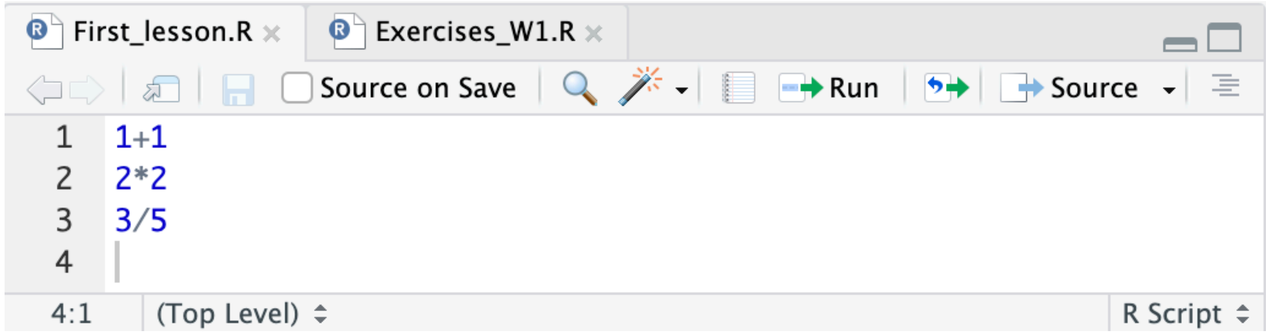
Run vs Source



Basics

Run vs Source

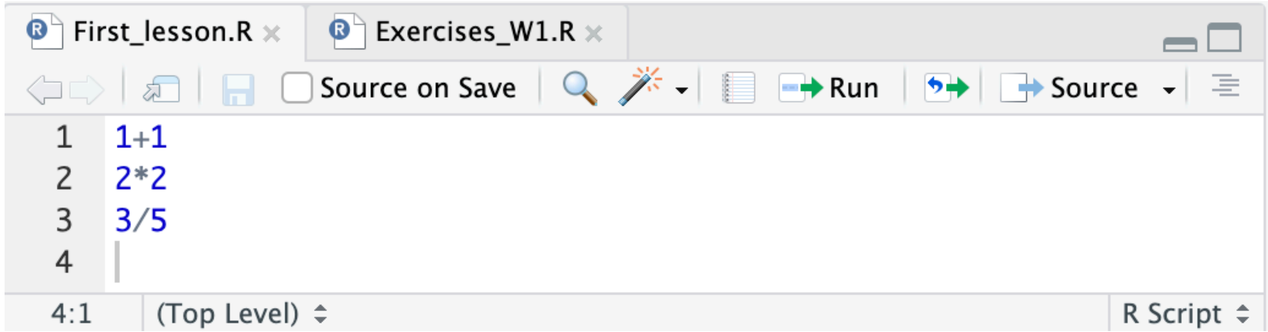
- The 'Source' function is used to execute all the code within an entire script file. When you source a file, R reads and executes every command in the script from start to finish.

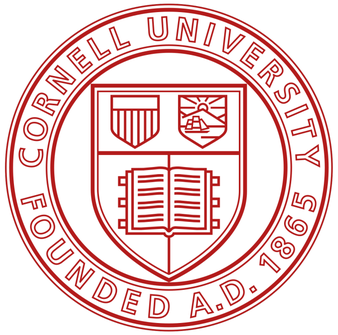


Basics

Run vs Source

- The 'Source' function is used to execute all the code within an entire script file. When you source a file, R reads and executes every command in the script from start to finish.
- The 'Run' command is used to execute selected lines of code from a script file. In RStudio, users can highlight specific portions of the script and use the 'Run' button to execute only those highlighted lines.





Playing with dices

Basics

Objects

Basics
Objects

- Let’s create a virtual die

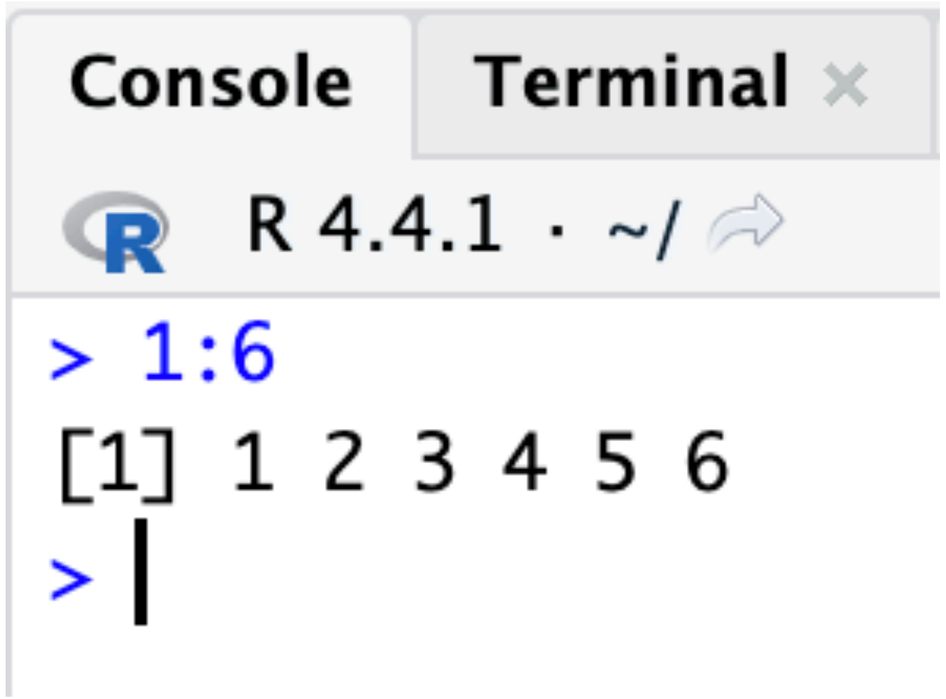
Basics

Objects

- Let’s create a virtual die
- The `:` operator returns its results as a vector, a one-dimensional set of numbers.

Basics

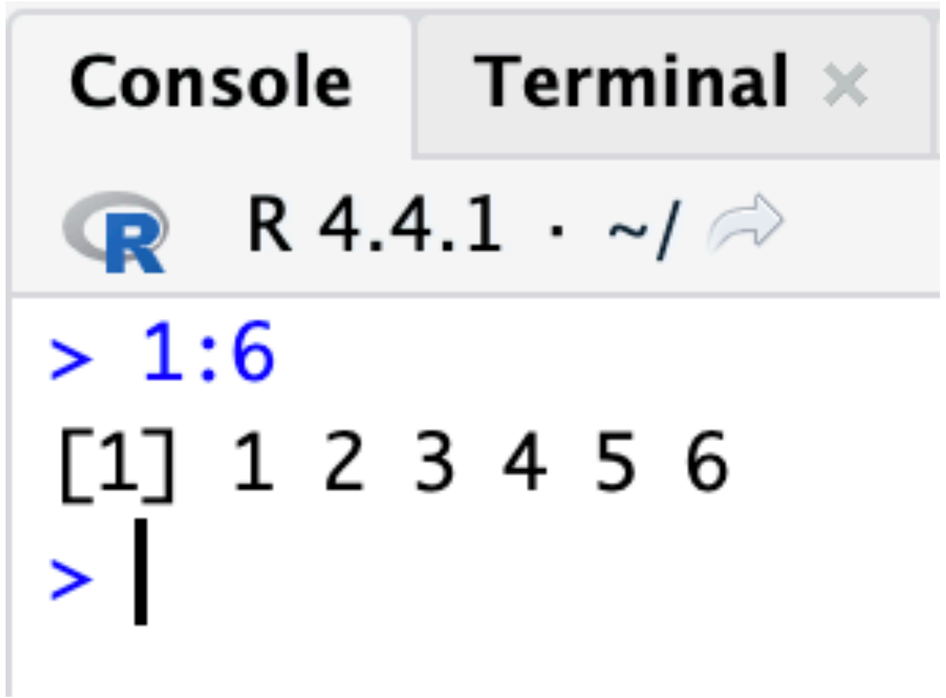
Vector



Basics

Vector

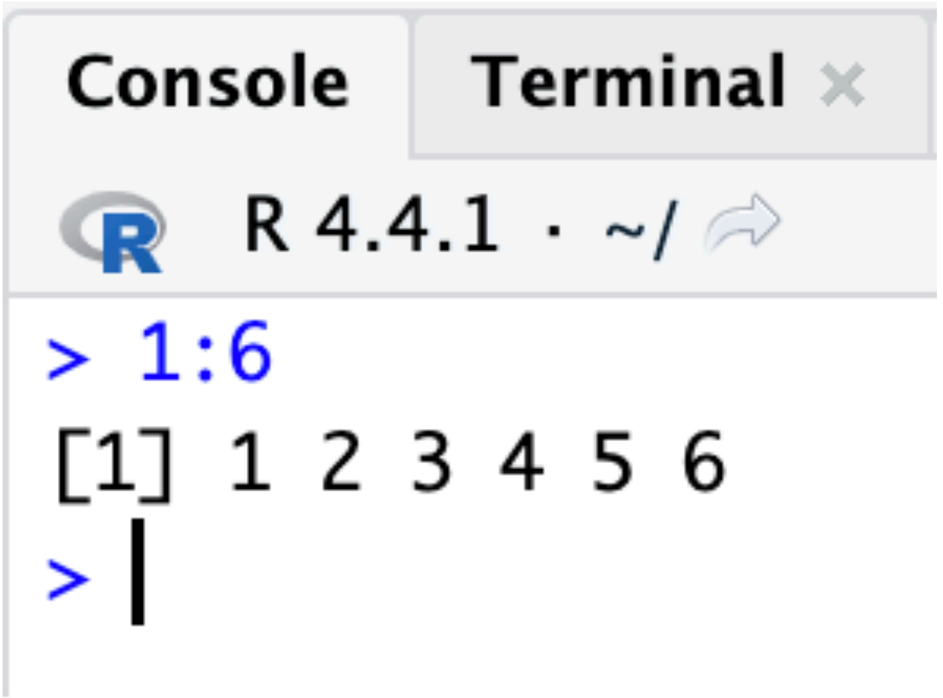
- Running `1:6` generated a vector of numbers for you to see, but it didn't save that vector anywhere in your computer's memory.



Basics

Vector

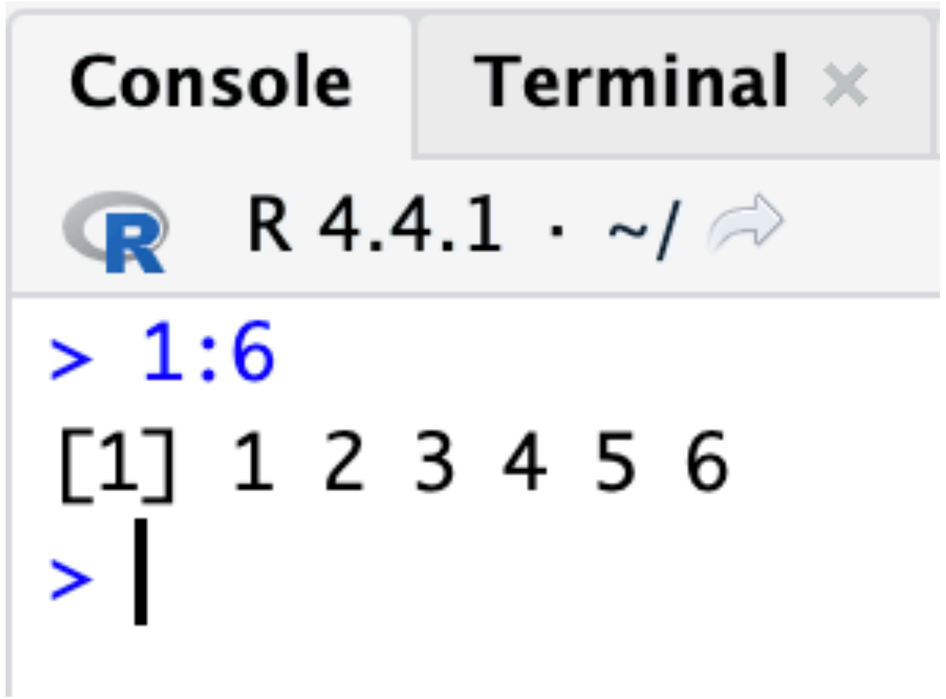
- Running `1:6` generated a vector of numbers for you to see, but it didn't save that vector anywhere in your computer's memory.
- If you want to use those numbers again, you'll have to ask your computer to save them somewhere.



Basics

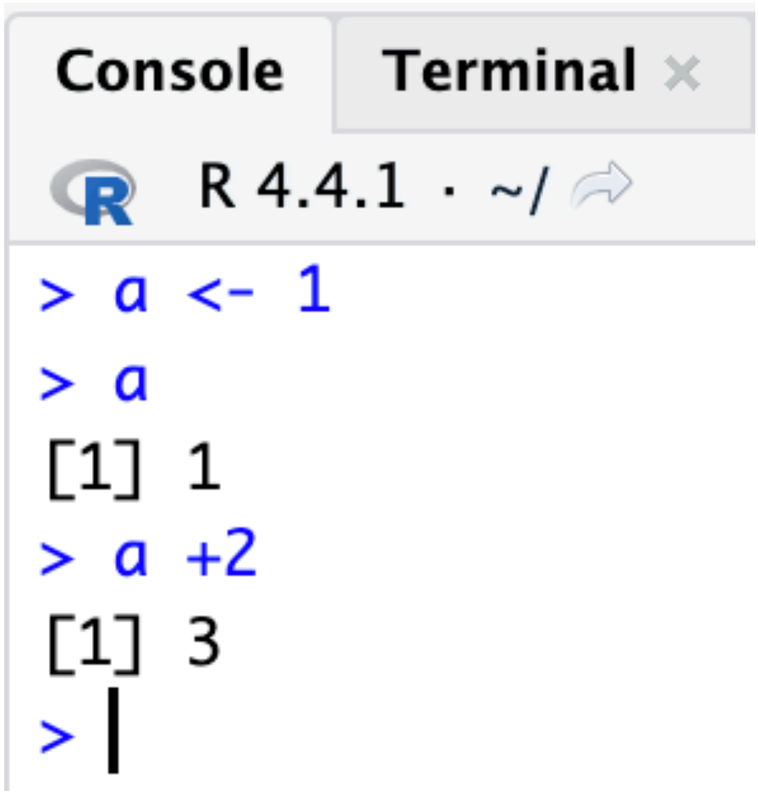
Vector

- Running `1:6` generated a vector of numbers for you to see, but it didn't save that vector anywhere in your computer's memory.
- If you want to use those numbers again, you'll have to ask your computer to save them somewhere.
- You can do that by creating an R *object*.



Basics

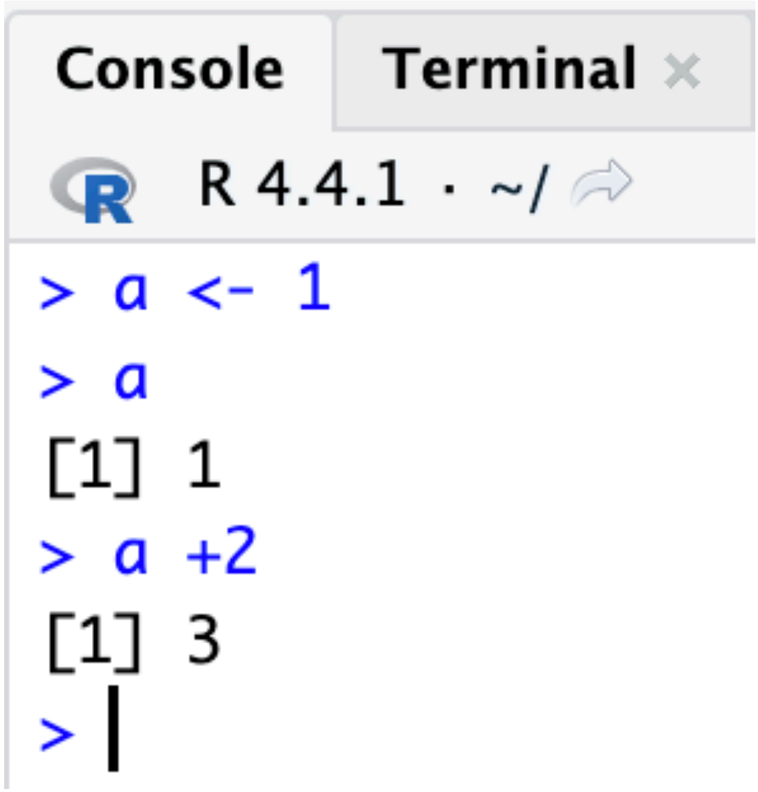
Storing data



Basics

Storing data

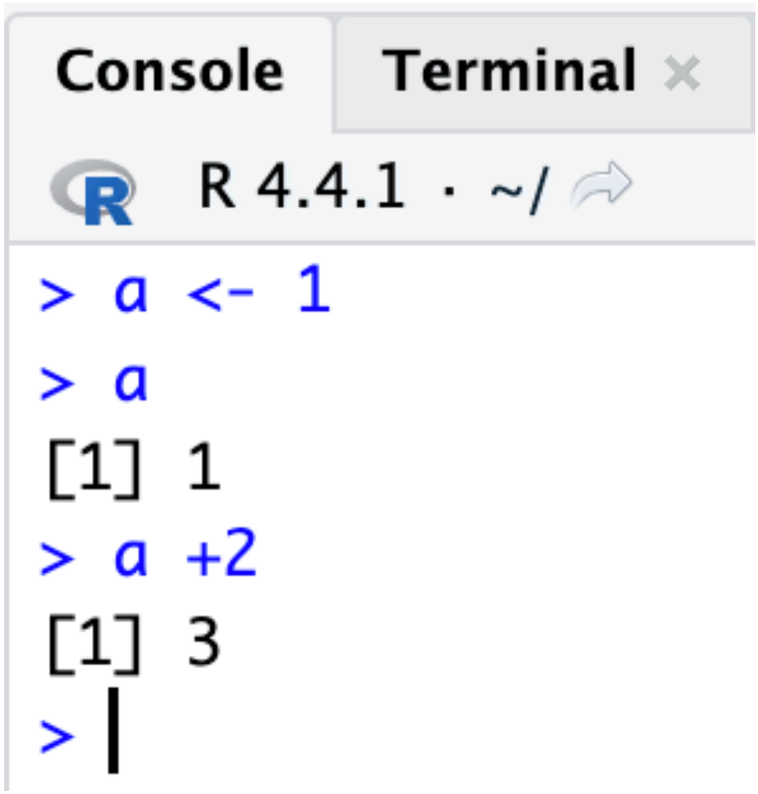
- R lets you save data by storing it inside an R object.



Basics

Storing data

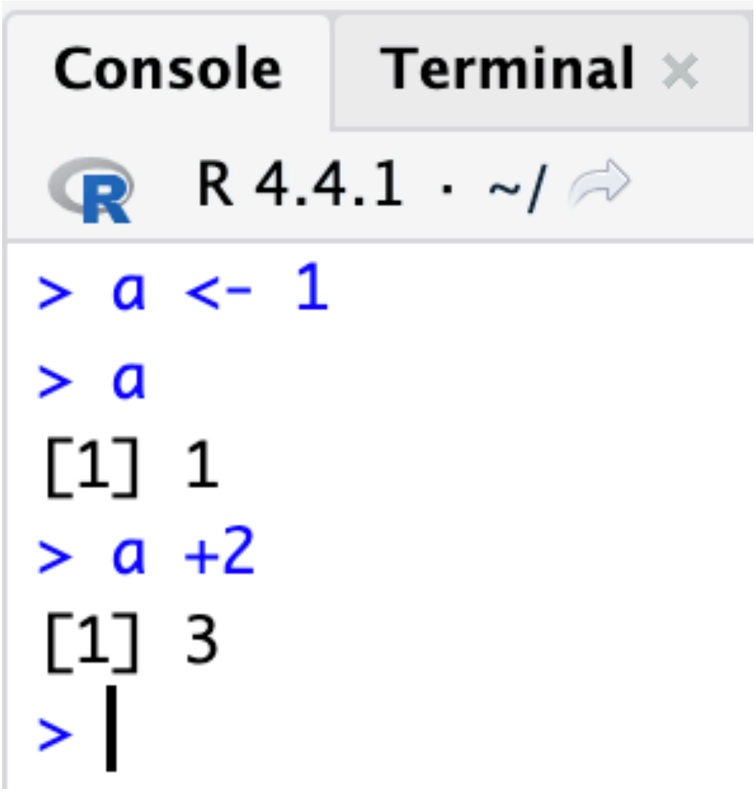
- R lets you save data by storing it inside an R object.
- What is an object? Just a name that you can use to call up stored data.



Basics

Storing data

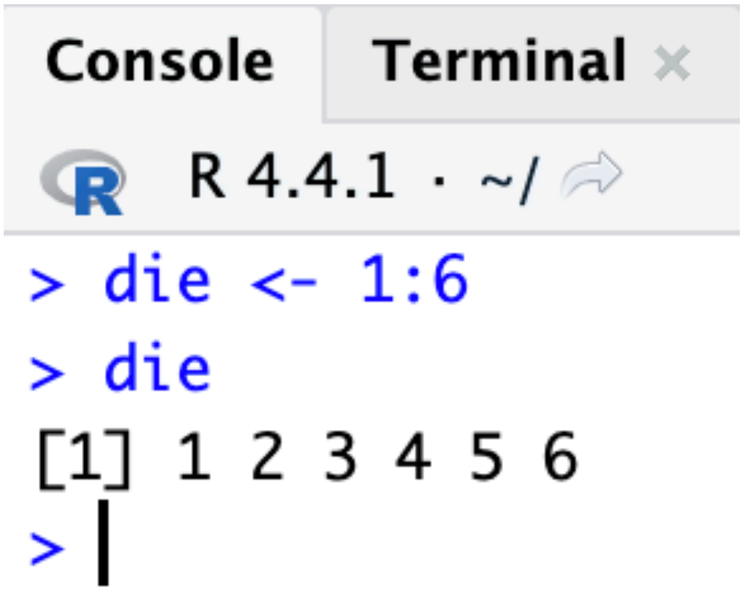
- R lets you save data by storing it inside an R object.
- What is an object? Just a name that you can use to call up stored data.
- For example, you can save data into an object like *a* or *b*. Wherever R encounters the object, it will replace it with the data saved inside.



Running R code

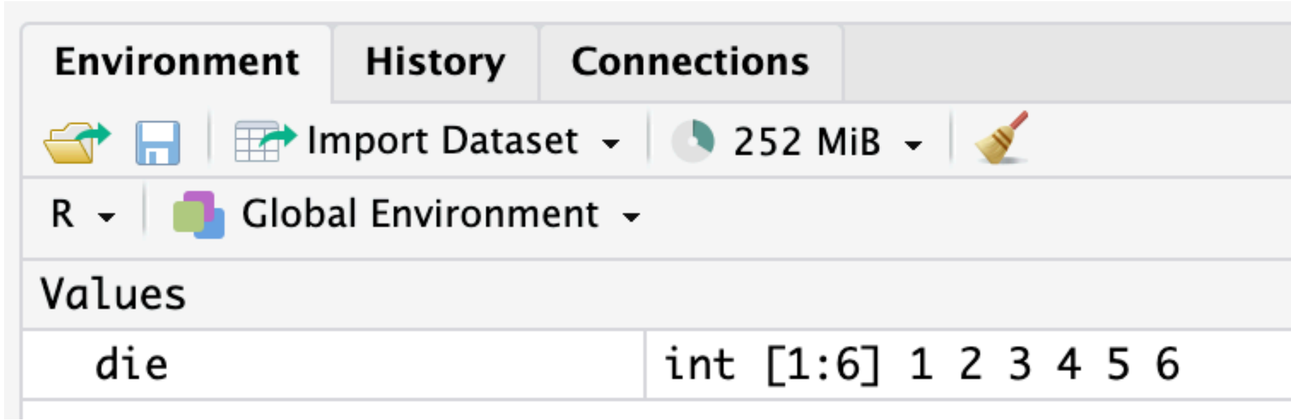
Create an object

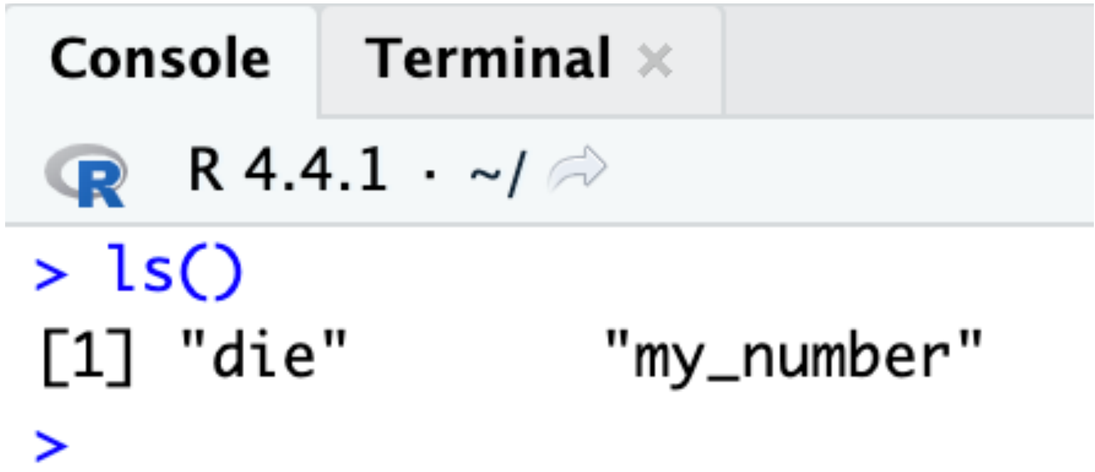
- Create an object named `die` that contains the numbers one through six.



Running R code

Environment pane

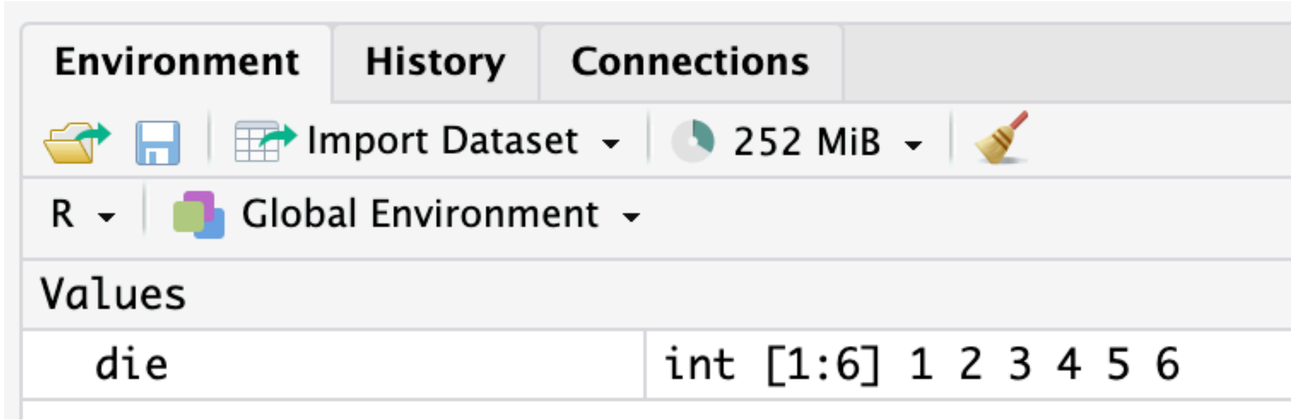


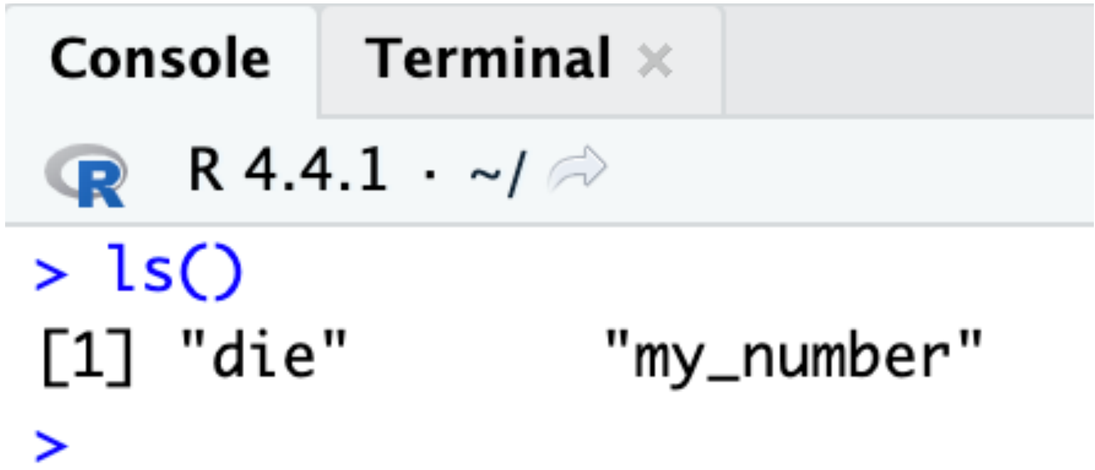


Running R code

Environment pane

- When you create an object, the object will appear in the environment pane of RStudio.

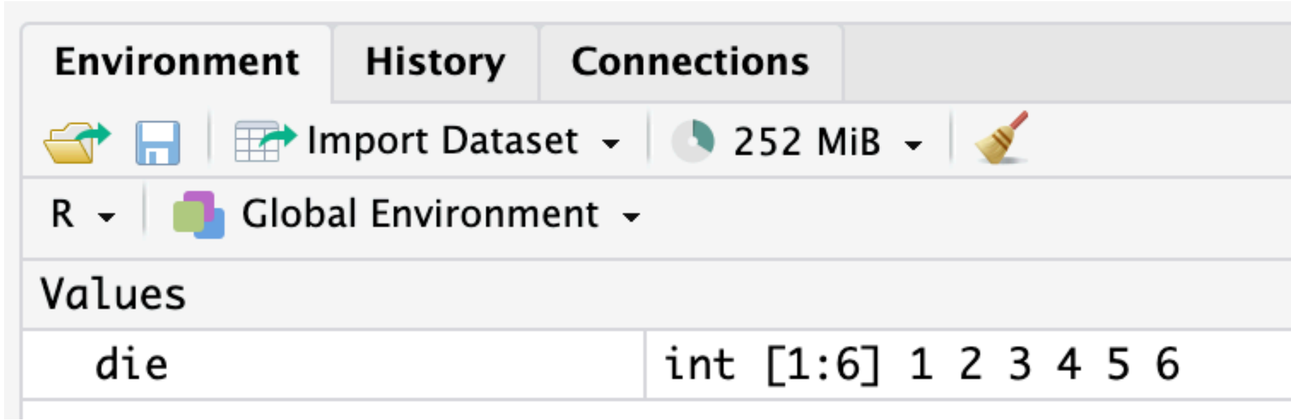


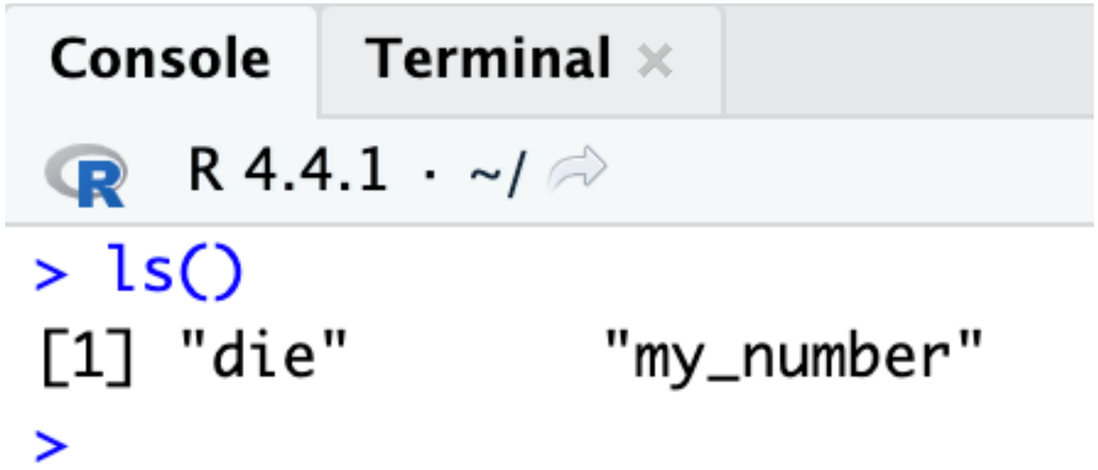


Running R code

Environment pane

- When you create an object, the object will appear in the environment pane of RStudio.
- This pane will show you all of the objects you've created since opening RStudio.

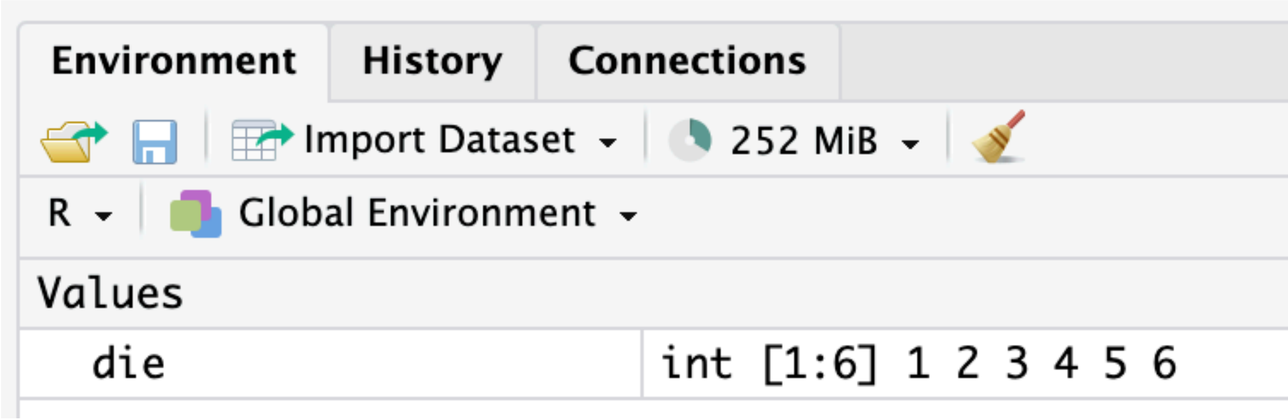


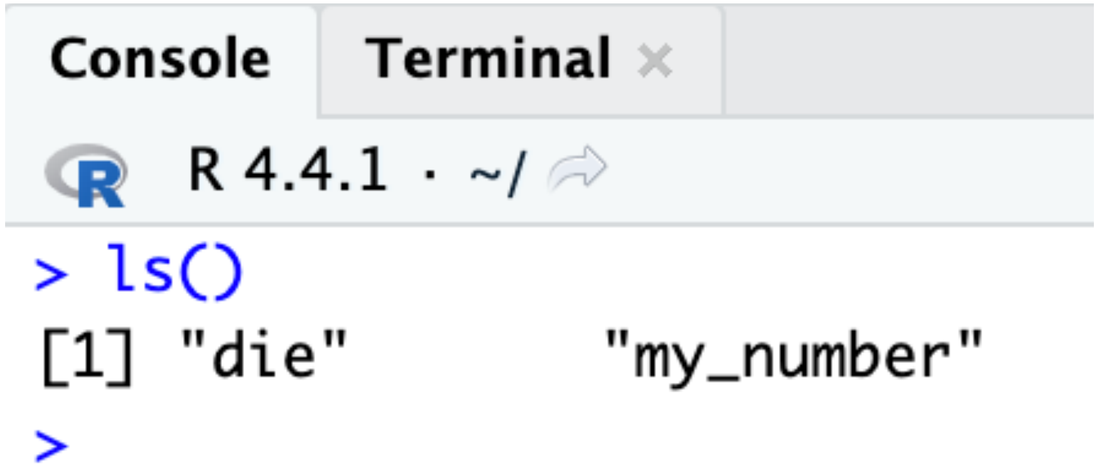


Running R code

Environment pane

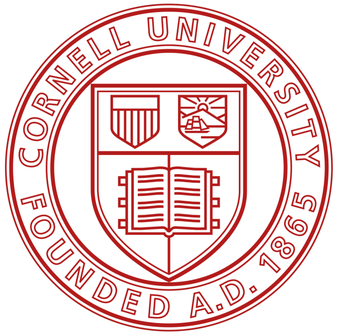
- When you create an object, the object will appear in the environment pane of RStudio.
- This pane will show you all of the objects you've created since opening RStudio.
- You can see which object names you have already used with the function `ls`





Running R code

Name an object



Good names	Names that cause errors
a	1trial
b	$
FOO	^mean
my_var	2nd
.day	!bad

Running R code

Name an object

- You can name an object in R almost anything you want, but there are a few rules.

Good names	Names that cause errors
a	1trial
b	$
FOO	^mean
my_var	2nd
.day	!bad

Running R code

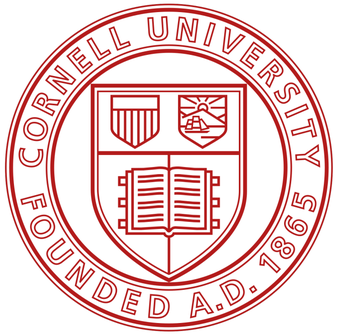
Name an object

- You can name an object in R almost anything you want, but there are a few rules.
- First, a name cannot start with a number.

Good names	Names that cause errors
a	1trial
b	$
FOO	^mean
my_var	2nd
.day	!bad

Running R code

Name an object

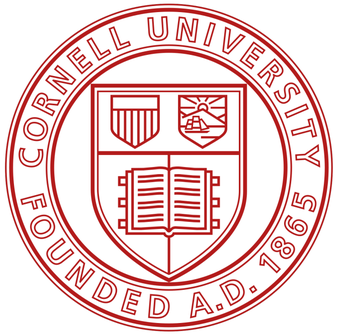


- You can name an object in R almost anything you want, but there are a few rules.
- First, a name cannot start with a number.
- Second, a name cannot use some special symbols, like `^`, `!`, `$`, `@`, `+`, `–`, `/`, or `*`

Good names	Names that cause errors
a	1trial
b	$
FOO	^mean
my_var	2nd
.day	!bad

Running R code

Name an object

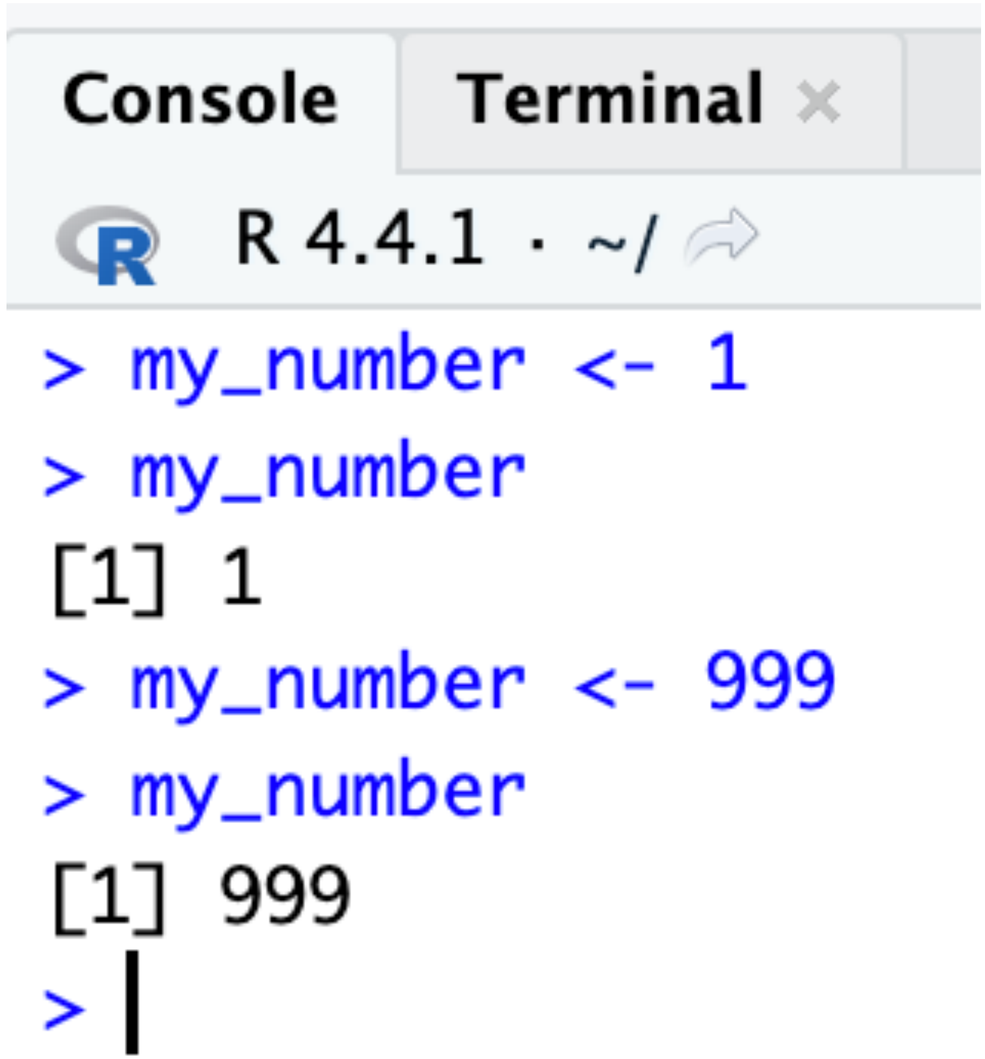


- You can name an object in R almost anything you want, but there are a few rules.
- First, a name cannot start with a number.
- Second, a name cannot use some special symbols, like `^`, `!`, `$`, `@`, `+`, `–`, `/`, or `*`
- R is case-sensitive, so `name` and `Name` will refer to different objects

Good names	Names that cause errors
a	1trial
b	$
FOO	^mean
my_var	2nd
.day	!bad

Running R code

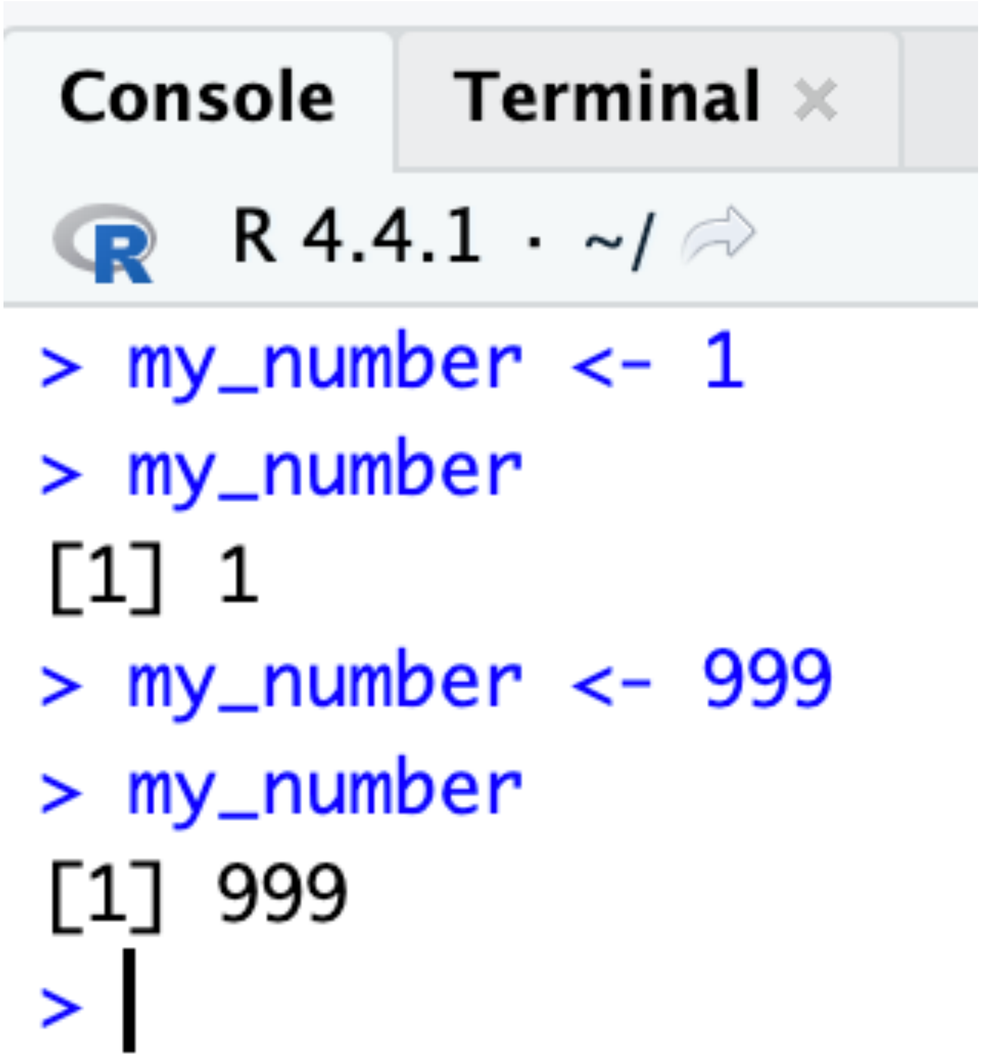
Name overwriting



Running R code

Name overwriting

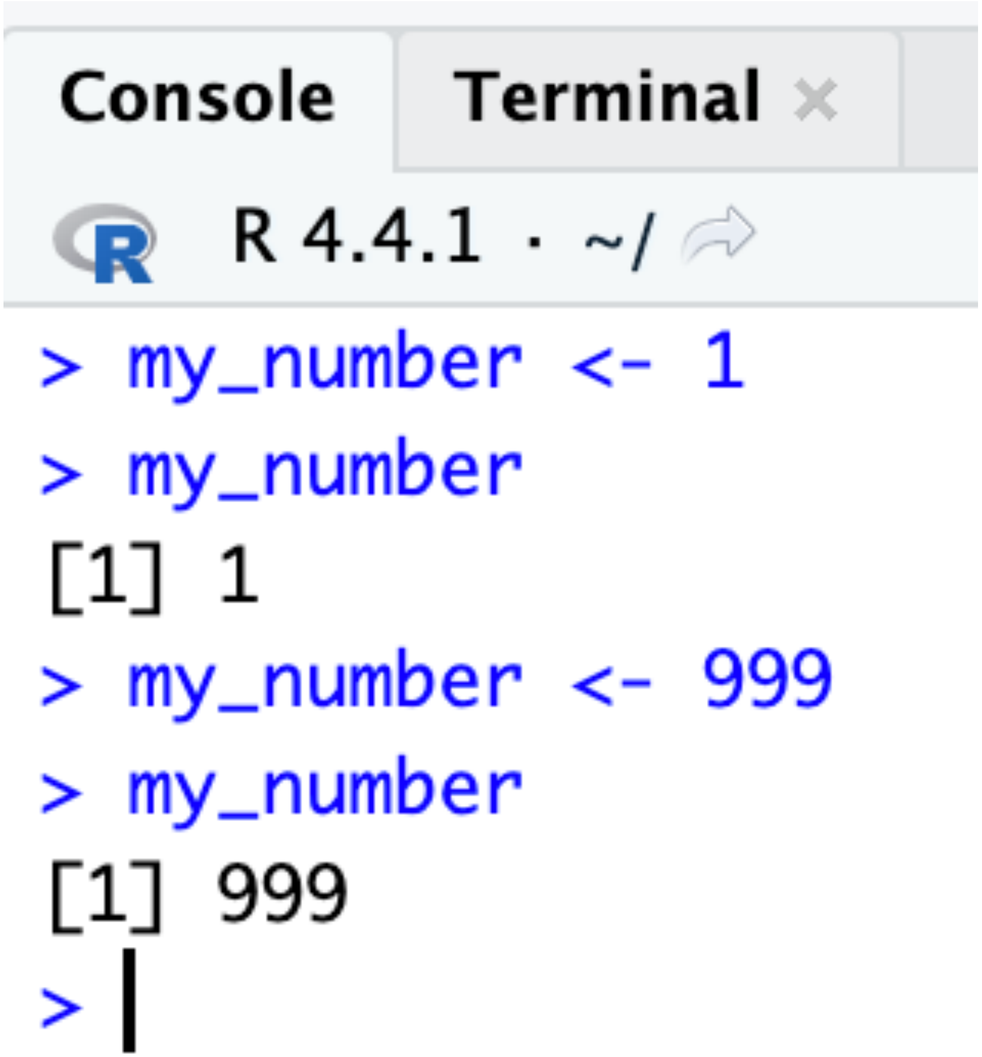
- R will overwrite any previous information stored in an object without asking you for permission.



Running R code

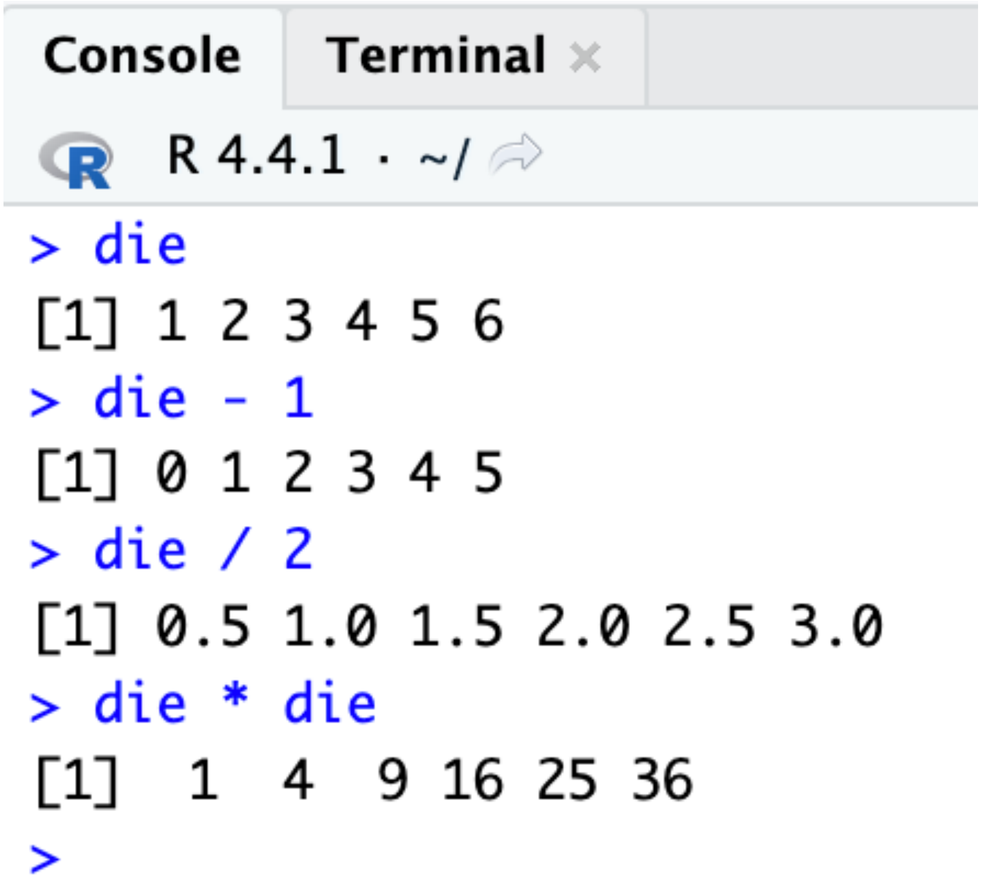
Name overwriting

- R will overwrite any previous information stored in an object without asking you for permission.
- It is a good idea to *not* use names that are already taken.



Running R code

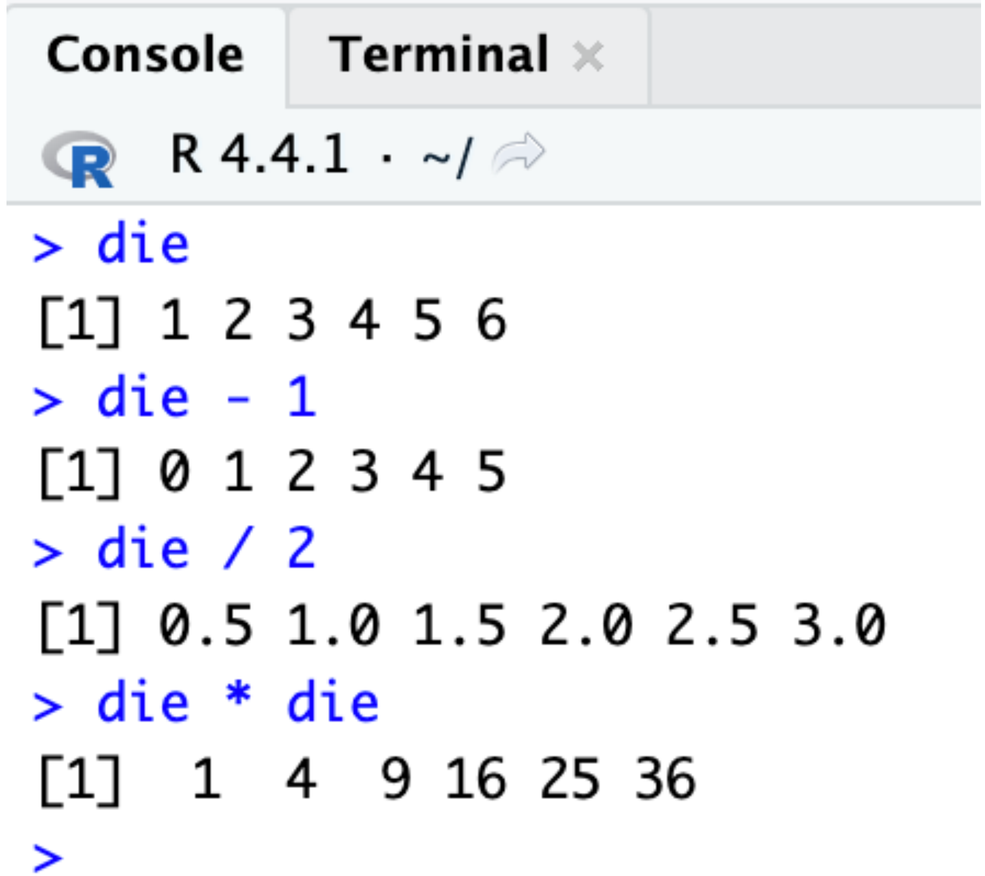
Element-wise execution



Running R code

Element-wise execution

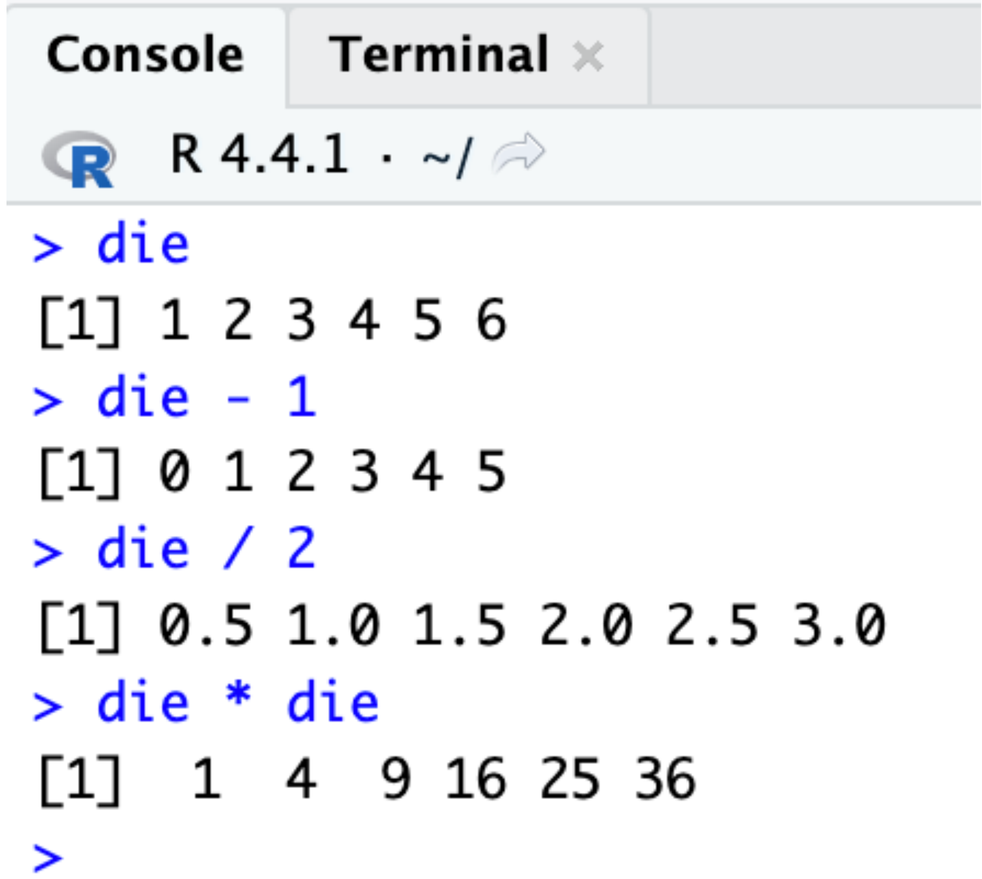
- You now have a virtual die that is stored in your computer's memory.



Running R code

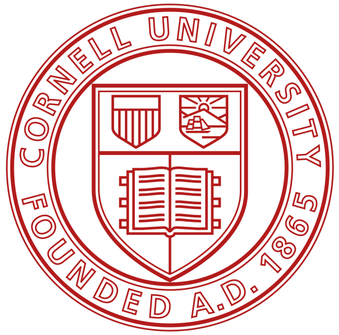
Element-wise execution

- You now have a virtual die that is stored in your computer's memory.
- You can do all sorts of math with the die.

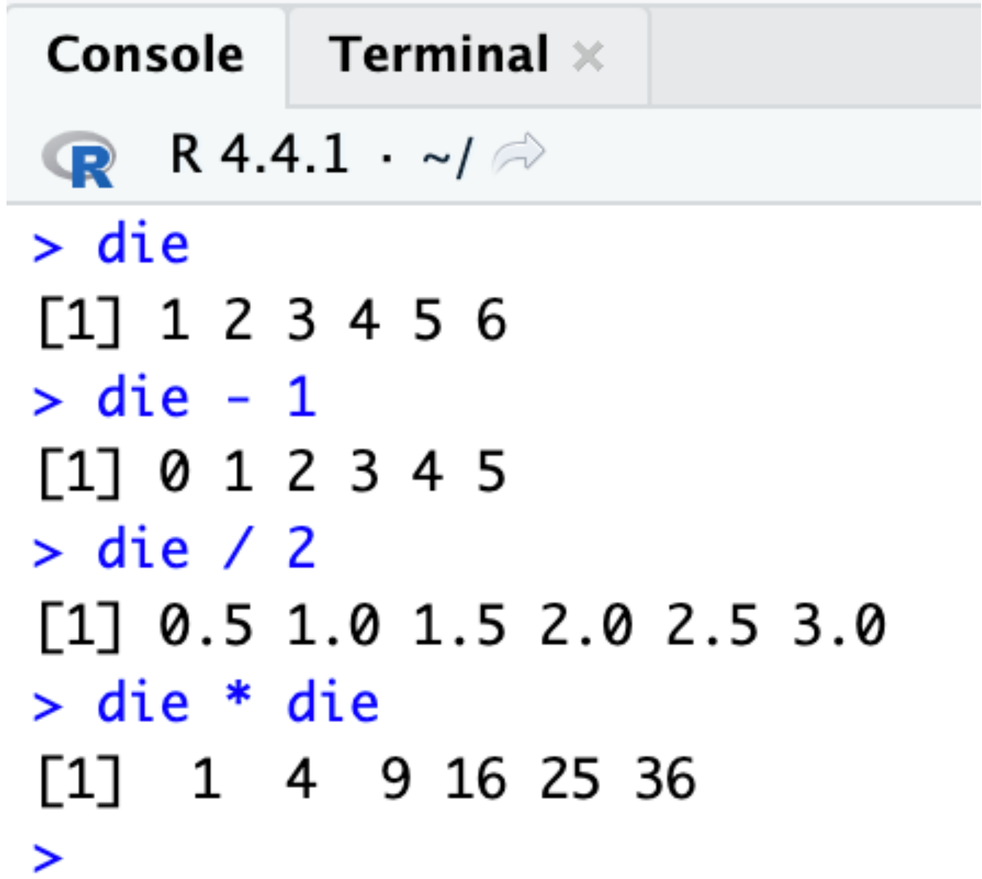


Running R code

Element-wise execution

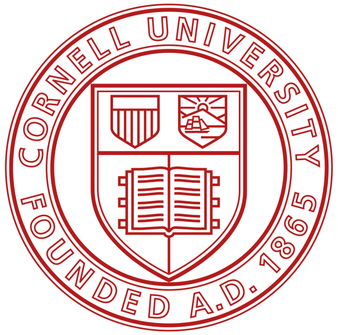


- You now have a virtual die that is stored in your computer's memory.
- You can do all sorts of math with the die.
- R does not always follow the rules of matrix multiplication.

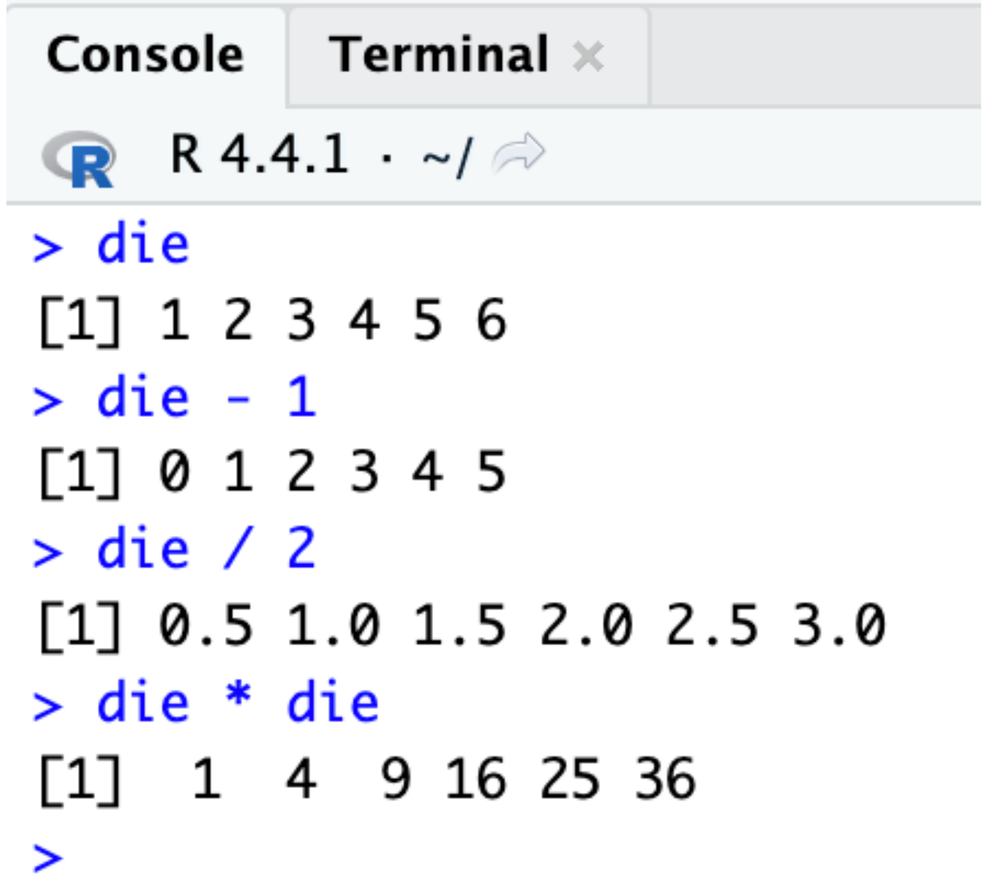


Running R code

Element-wise execution

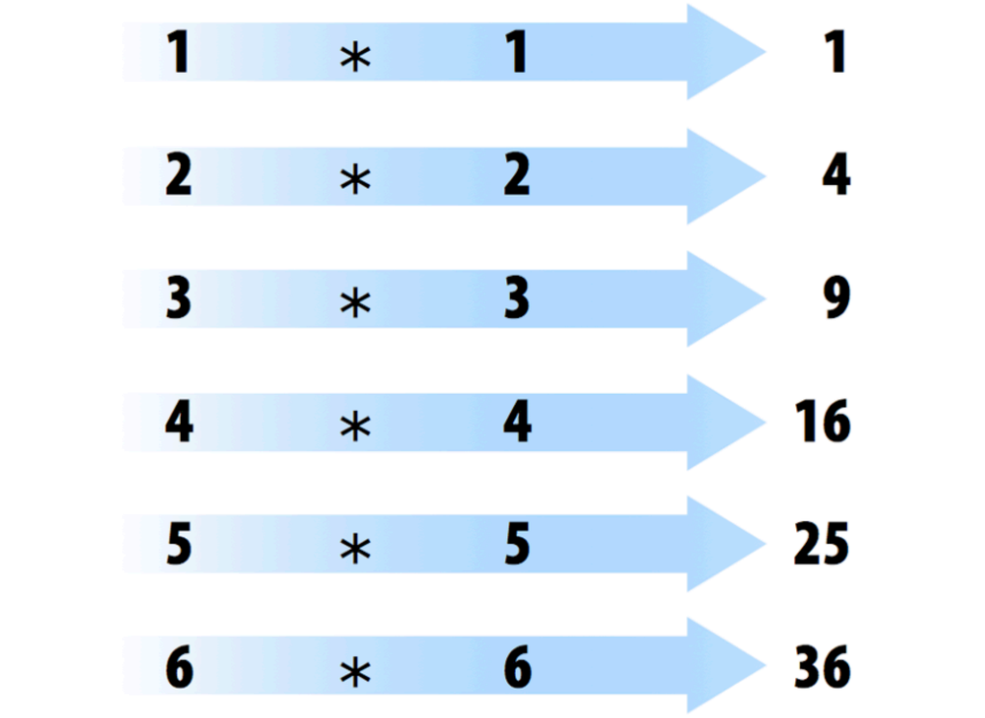


- You now have a virtual die that is stored in your computer's memory.
- You can do all sorts of math with the die.
- R does not always follow the rules of matrix multiplication.
- R uses element-wise execution.



Running R code

Element-wise execution

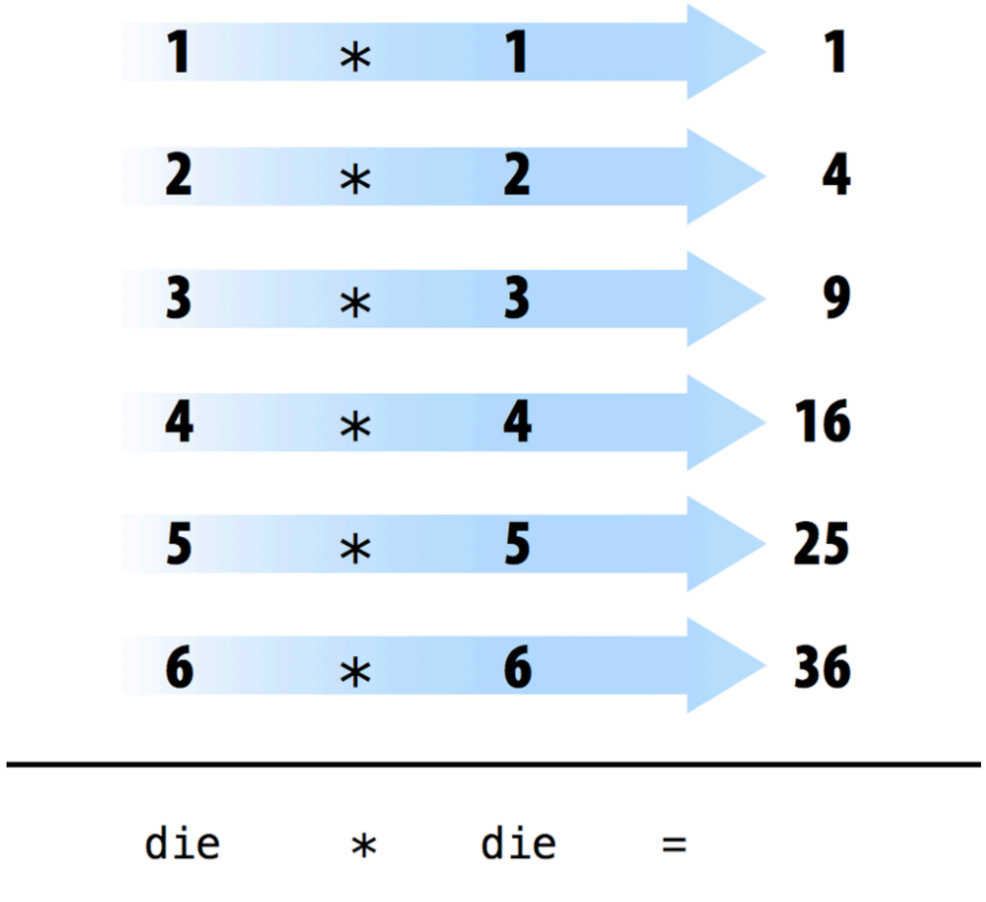



```
die     *     die     =
```

Running R code

Element-wise execution

- When you use two or more vectors in an operation, R will line up the vectors and perform a sequence of individual operations.



Running R code

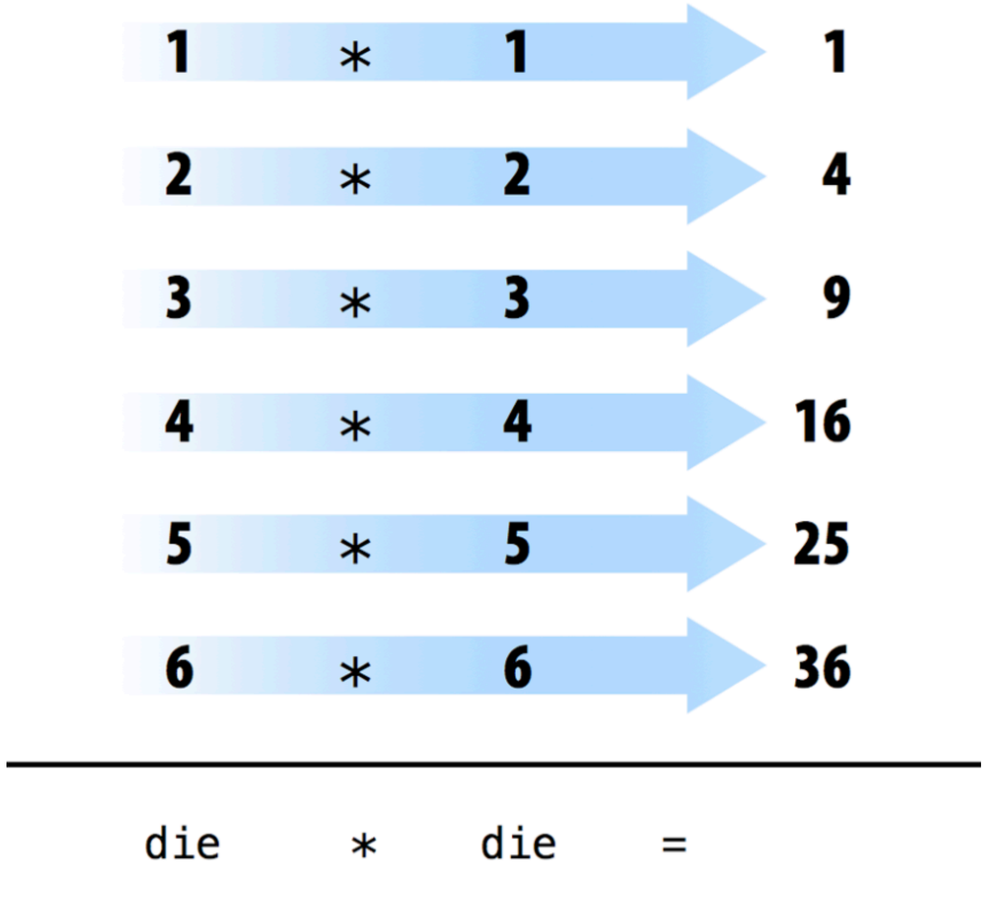
Element-wise execution

- When you use two or more vectors in an operation, R will line up the vectors and perform a sequence of individual operations.
- For example, when you run `die * die`, R lines up the two `die` vectors and then multiplies the first element of vector 1 by the first element of vector 2.

Running R code

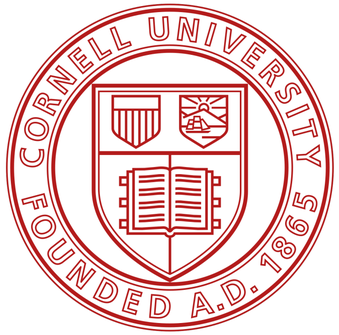
Element-wise execution

- When you use two or more vectors in an operation, R will line up the vectors and perform a sequence of individual operations.
- For example, when you run `die * die`, R lines up the two `die` vectors and then multiplies the first element of vector 1 by the first element of vector 2.
- R then multiplies the second element of vector 1 by the second element of vector 2, and so on, until every element has been multiplied.



Running R code

Vector recycling



Console | Terminal

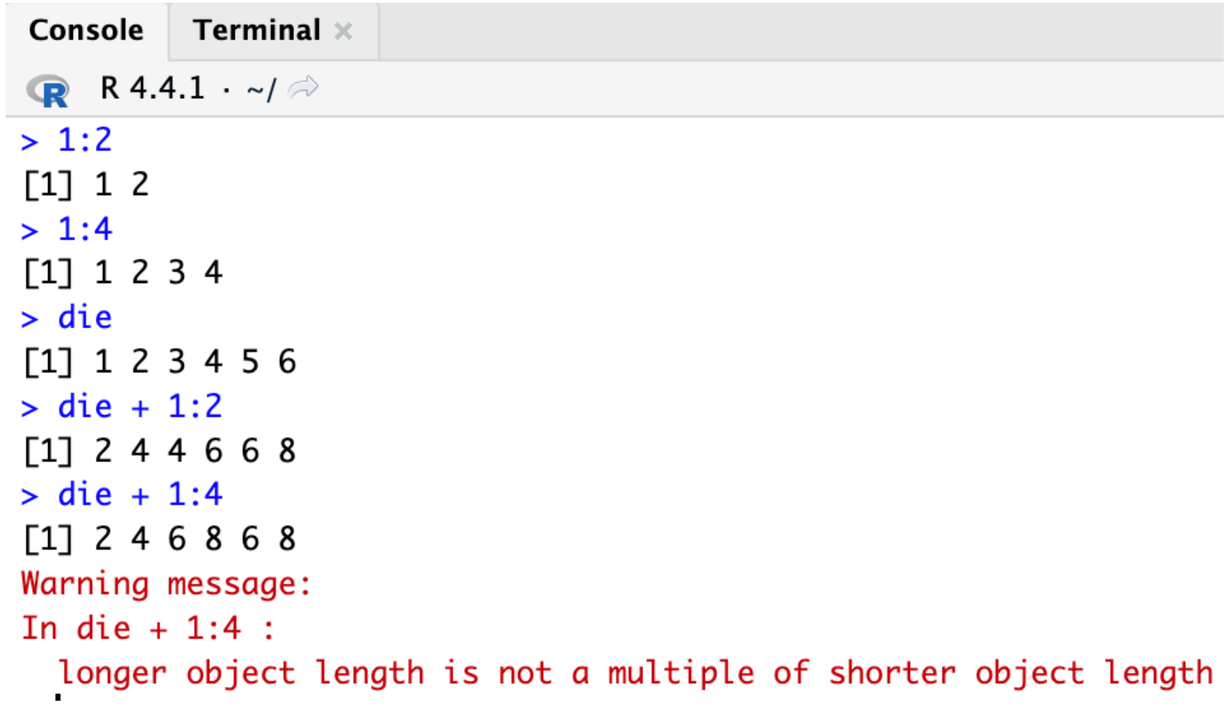
R 4.4.1 · ~/

```
> 1:2
[1] 1 2
> 1:4
[1] 1 2 3 4
> die
[1] 1 2 3 4 5 6
> die + 1:2
[1] 2 4 4 6 6 8
> die + 1:4
[1] 2 4 6 8 6 8
Warning message:
In die + 1:4 :
  longer object length is not a multiple of shorter object length
```

Running R code

Vector recycling

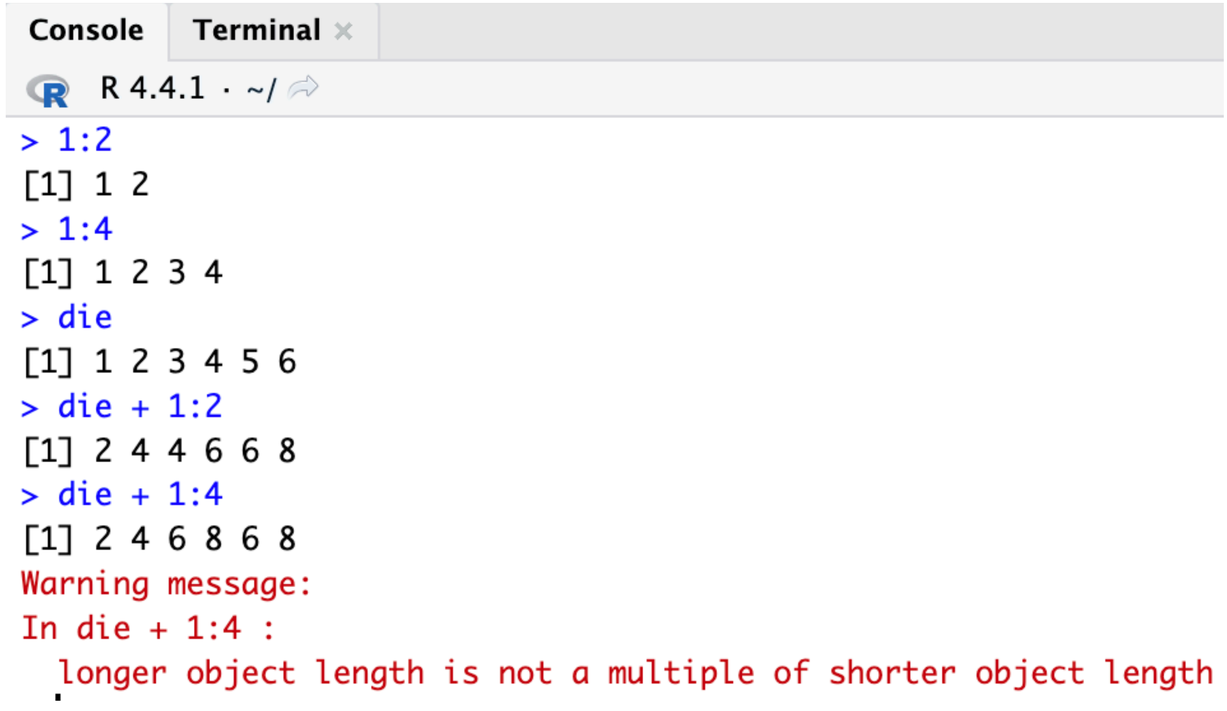
- If you give R two vectors of unequal lengths, R will repeat the shorter vector until it is as long as the longer vector, and then do the math (*vector recycling*)



Running R code

Vector recycling

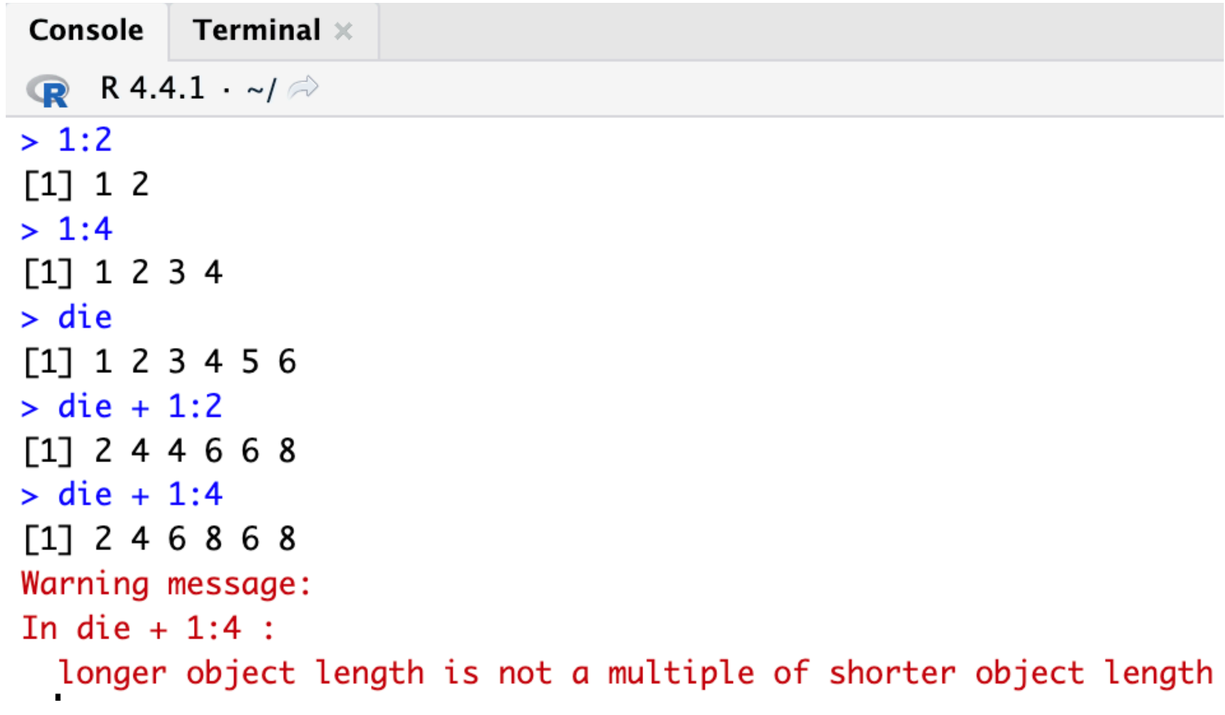
- If you give R two vectors of unequal lengths, R will repeat the shorter vector until it is as long as the longer vector, and then do the math (*vector recycling*)
- This isn't a permanent change, the shorter vector will be its original size after R does the math.



Running R code

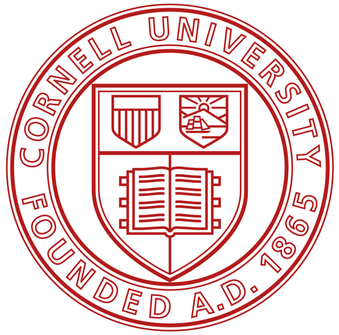
Vector recycling

- If you give R two vectors of unequal lengths, R will repeat the shorter vector until it is as long as the longer vector, and then do the math (*vector recycling*)
- This isn't a permanent change, the shorter vector will be its original size after R does the math.
- If the length of the short vector does not divide evenly into the length of the long vector, R will return a warning message.



Running R code

Linear algebra operations



Console | Terminal

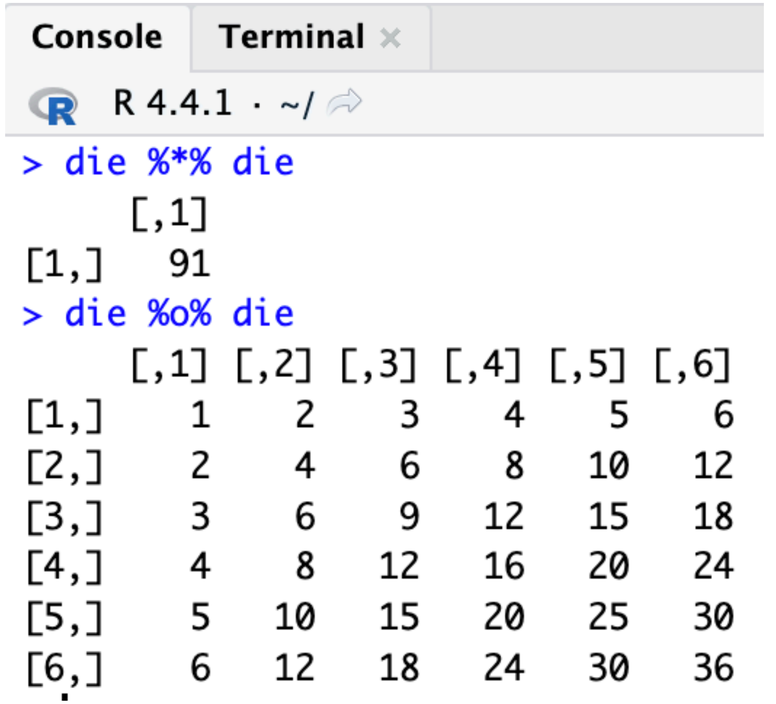
R 4.4.1 · ~/

```
> die %*% die
     [,1]
[1,]   91
> die %o% die
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    2    3    4    5    6
[2,]    2    4    6    8   10   12
[3,]    3    6    9   12   15   18
[4,]    4    8   12   16   20   24
[5,]    5   10   15   20   25   30
[6,]    6   12   18   24   30   36
```

Running R code

Linear algebra operations

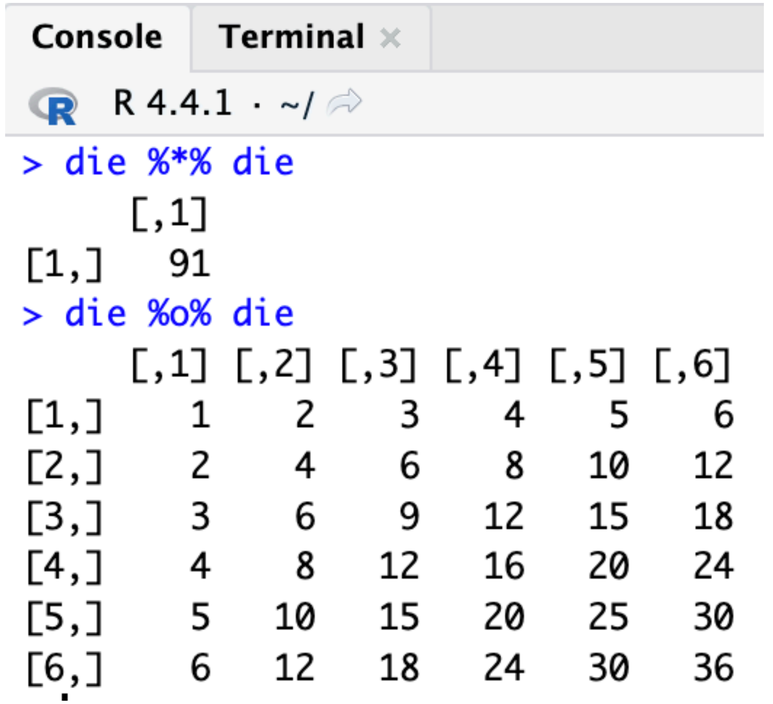
- Working with data sets, element-wise operations ensure that values from one observation are only paired with values from the same observation.



Running R code

Linear algebra operations

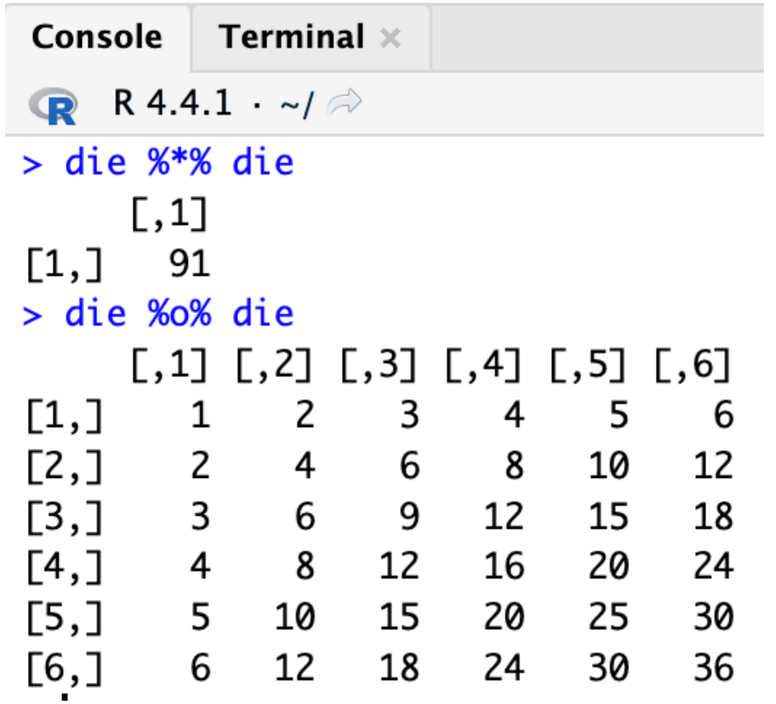
- Working with data sets, element-wise operations ensure that values from one observation are only paired with values from the same observation.
- Don’t think that R has given up on traditional matrix multiplication.



Running R code

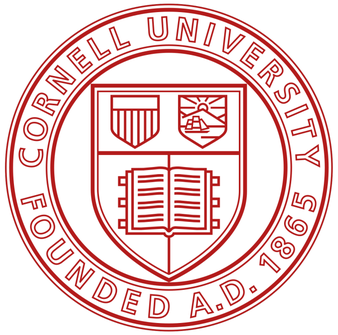
Linear algebra operations

- Working with data sets, element-wise operations ensure that values from one observation are only paired with values from the same observation.
- Don’t think that R has given up on traditional matrix multiplication.
- You just have to ask for it when you want it. You can do inner multiplication with the `%*%` operator and outer multiplication with the `%o%` operator.



Running R code

Vector inner product

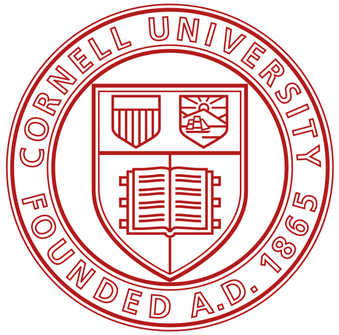


$$\left\langle \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\rangle = x^\mathsf{T} y = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + \cdots + x_n y_n$$

$\langle \cdot, \cdot \rangle$

Running R code

Vector inner product



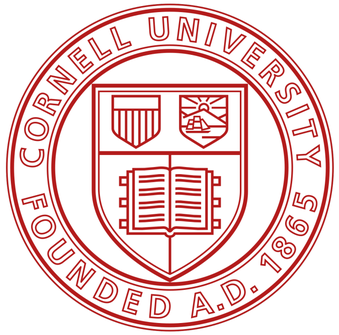
- A vector inner product is a binary operation that takes two vectors and returns a scalar.

$$\left\langle \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\rangle = x^\mathsf{T} y = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + \cdots + x_n y_n$$

$$\langle \cdot, \cdot \rangle$$

Running R code

Vector inner product

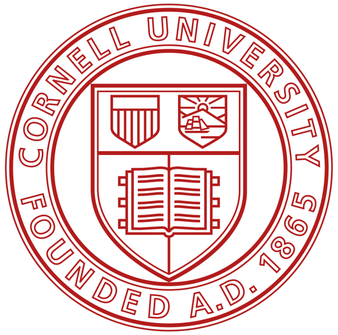


- A vector inner product is a binary operation that takes two vectors and returns a scalar.
- It is often denoted $\langle \cdot, \cdot \rangle$

$$\left\langle \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\rangle = x^\mathsf{T} y = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + \cdots + x_n y_n$$

Running R code

Vector outer product



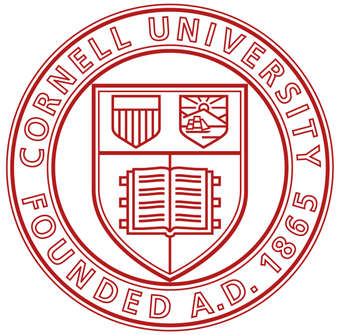
Given two vectors of size $m \times 1$ and $n \times 1$ respectively

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

$$\mathbf{u} \otimes \mathbf{v} = \mathbf{A} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & \dots & u_1 v_n \\ u_2 v_1 & u_2 v_2 & \dots & u_2 v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_m v_1 & u_m v_2 & \dots & u_m v_n \end{bmatrix}$$

Running R code

Vector outer product



- The outer product of two coordinate vectors is the matrix whose entries are all products of an element in the first vector with an element in the second vector.

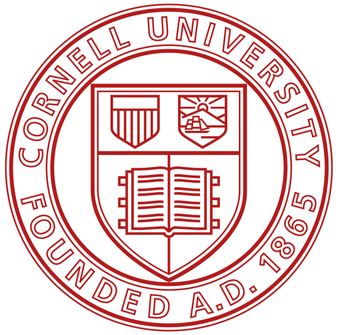
Given two vectors of size $m \times 1$ and $n \times 1$ respectively

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

$$\mathbf{u} \otimes \mathbf{v} = \mathbf{A} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & \dots & u_1 v_n \\ u_2 v_1 & u_2 v_2 & \dots & u_2 v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_m v_1 & u_m v_2 & \dots & u_m v_n \end{bmatrix}$$

Running R code

Vector outer product



- The outer product of two coordinate vectors is the matrix whose entries are all products of an element in the first vector with an element in the second vector.
- If the two coordinate vectors have dimensions n and m, then their outer product is an n × m matrix.

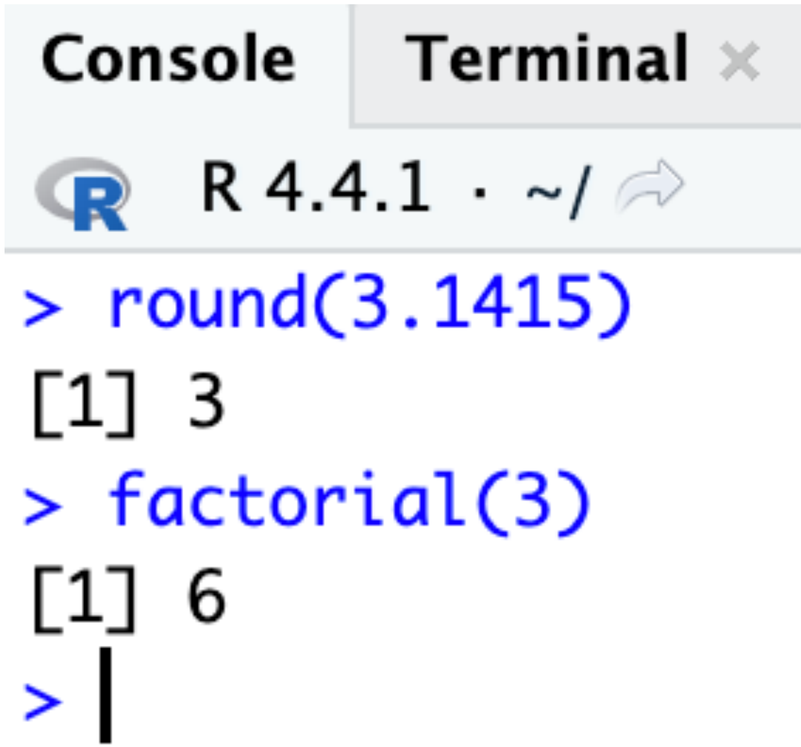
Given two vectors of size $m \times 1$ and $n \times 1$ respectively

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

$$\mathbf{u} \otimes \mathbf{v} = \mathbf{A} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & \dots & u_1 v_n \\ u_2 v_1 & u_2 v_2 & \dots & u_2 v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_m v_1 & u_m v_2 & \dots & u_m v_n \end{bmatrix}$$

Running R code

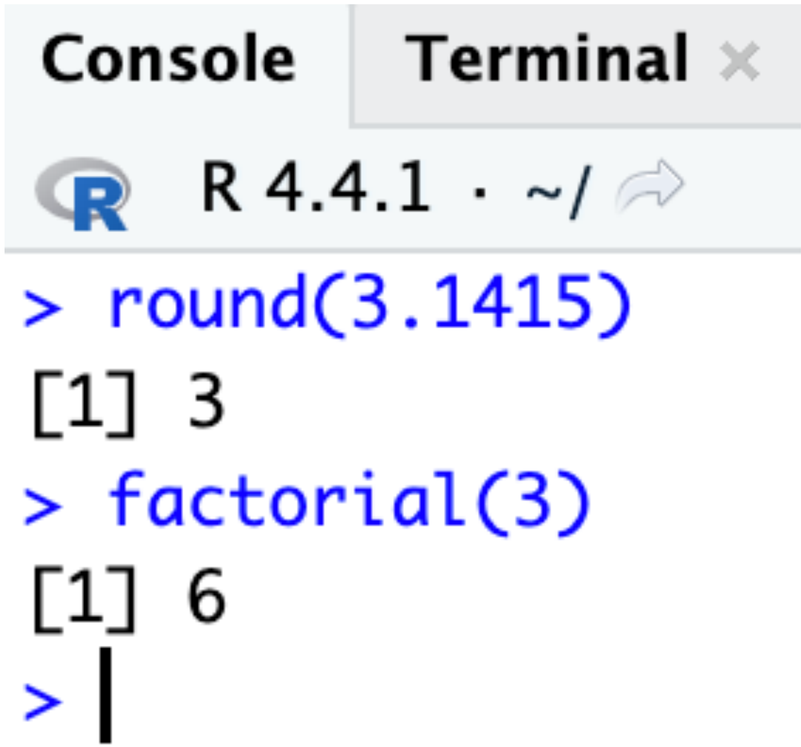
Functions



Running R code

Functions

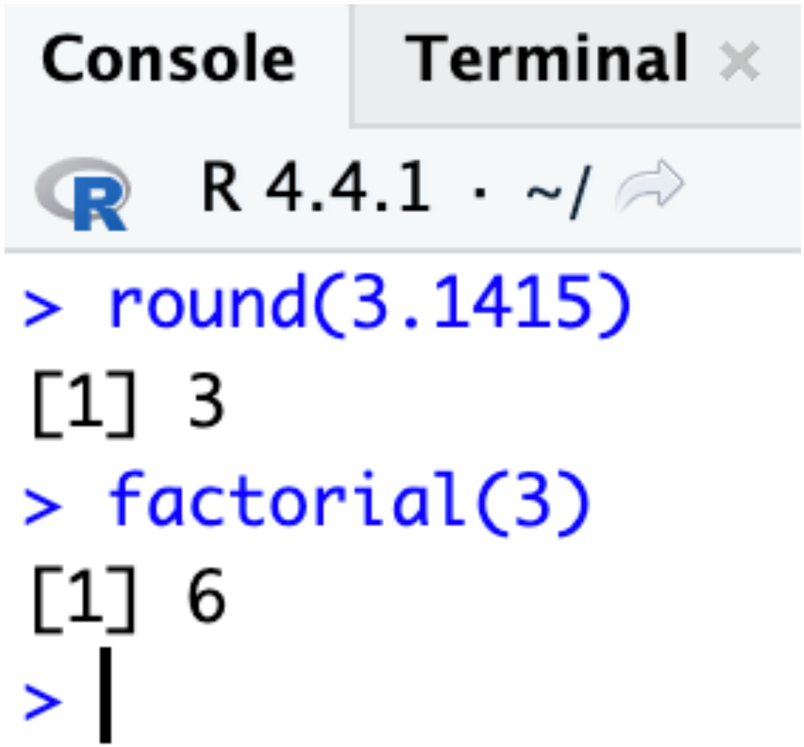
- R comes with many functions that you can use to do sophisticated tasks like random sampling.



Running R code

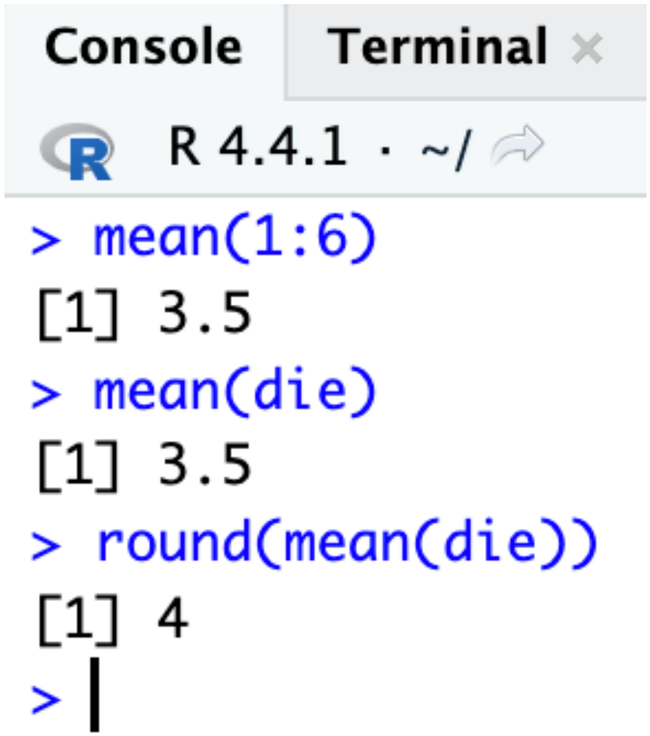
Functions

- R comes with many functions that you can use to do sophisticated tasks like random sampling.
- For example, you can round a number with the `round` function, or calculate its factorial with the `factorial` function.



Running R code

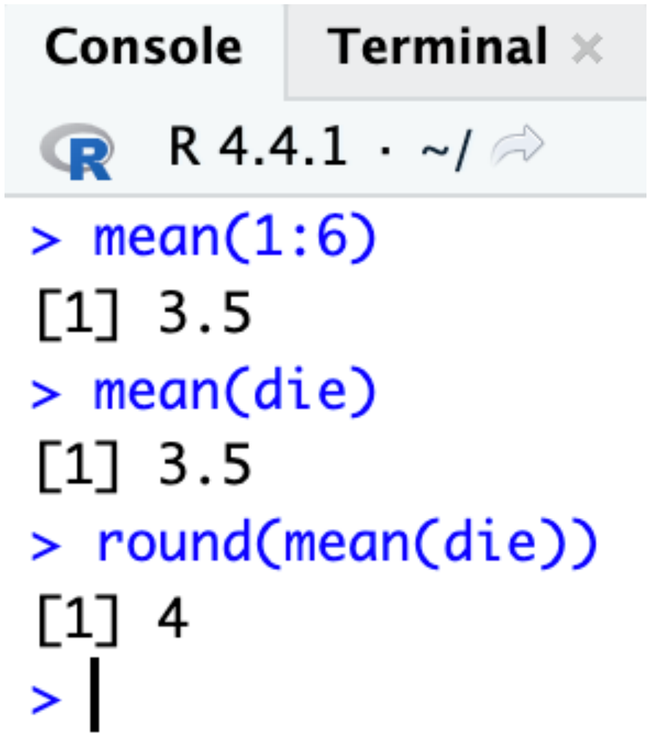
Functions



Running R code

Functions

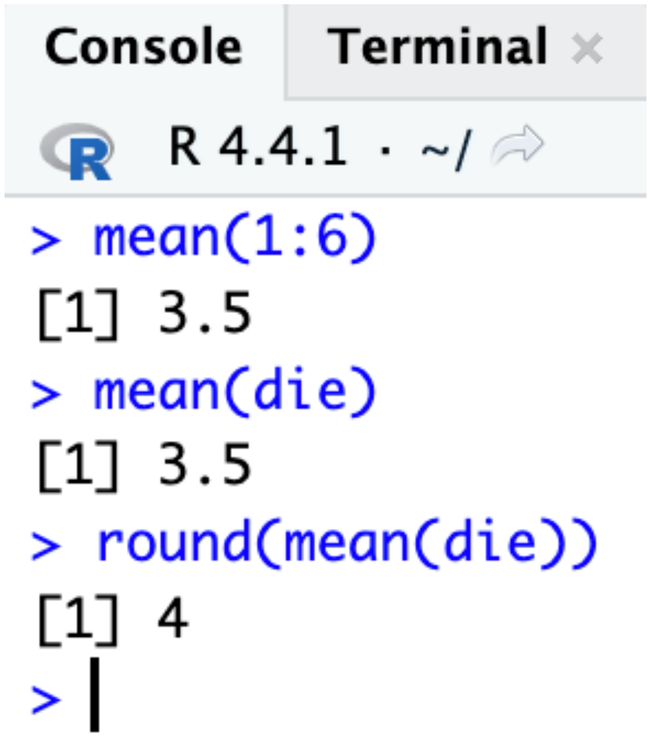
- The data that you pass into the function is called the function's *argument*.



Running R code

Functions

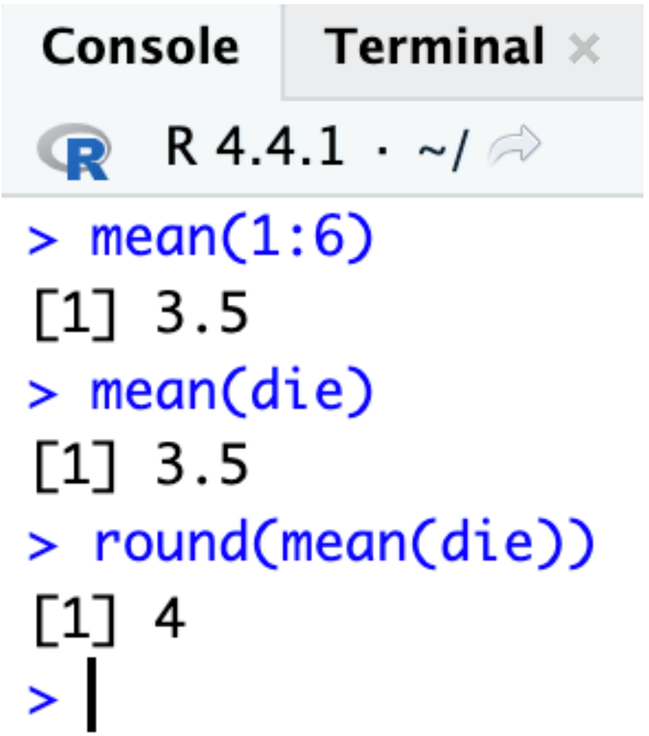
- The data that you pass into the function is called the function's *argument*.
- The argument can be raw data, an R object, or even the results of another R function.



Running R code

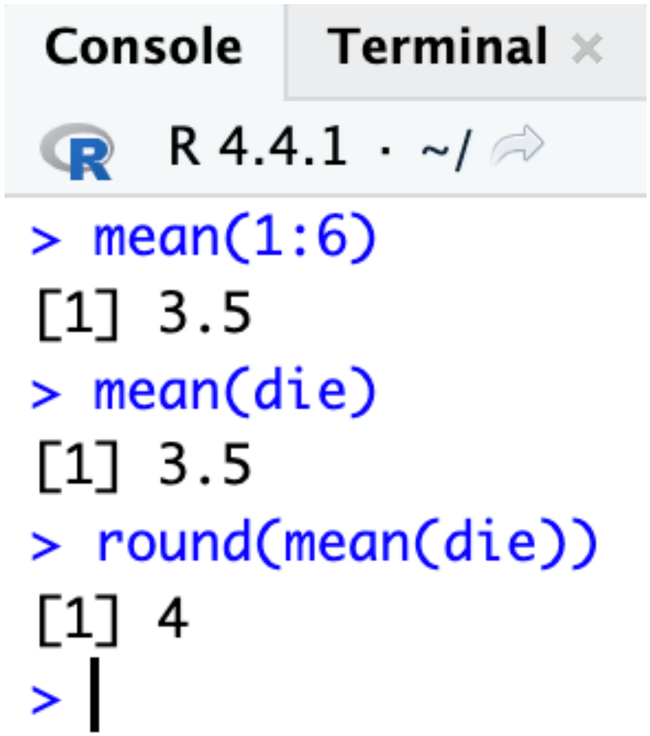
Functions

- The data that you pass into the function is called the function's *argument*.
- The argument can be raw data, an R object, or even the results of another R function.
- In this last case, R will work from the innermost function to the outermost



Running R code

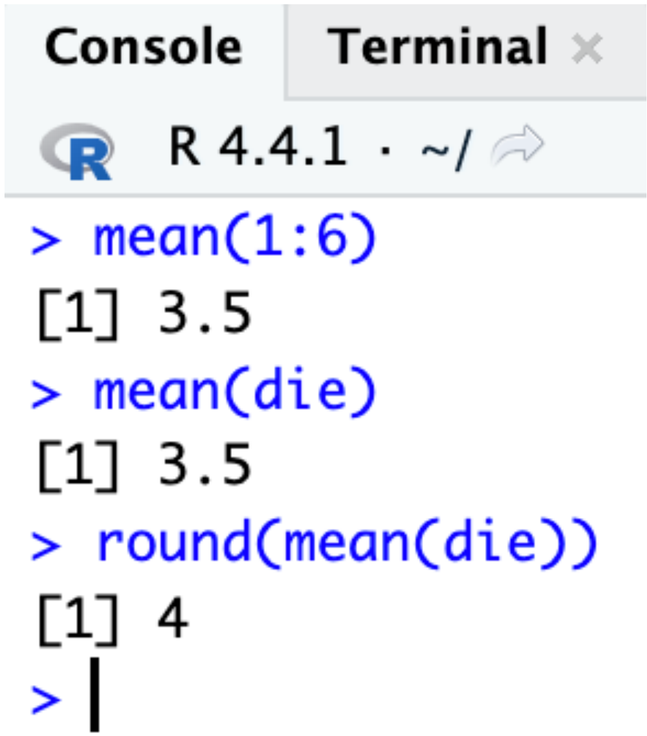
Functions



Running R code

Functions

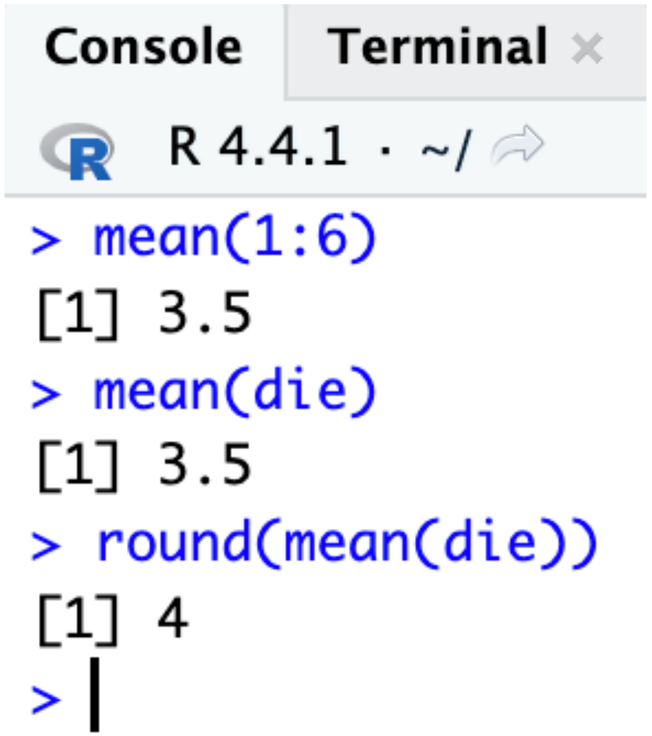
- The data that you pass into the function is called the function's *argument*.



Running R code

Functions

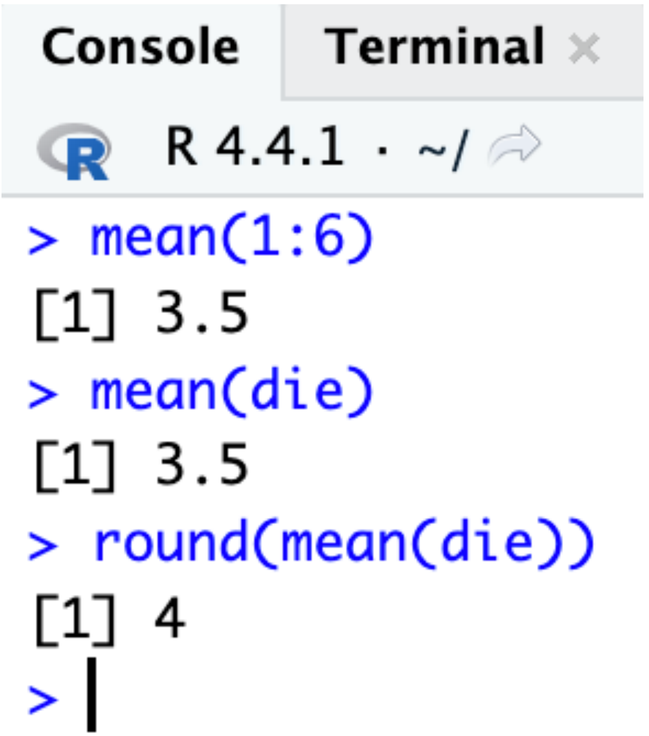
- The data that you pass into the function is called the function's *argument*.
- The argument can be raw data, an R object, or even the results of another R function.



Running R code

Functions

- The data that you pass into the function is called the function's *argument*.
- The argument can be raw data, an R object, or even the results of another R function.
- In this last case, R will work from the innermost function to the outermost



Basics

Roll a dice

Basics

Roll a dice

- Let’s roll the virtual die

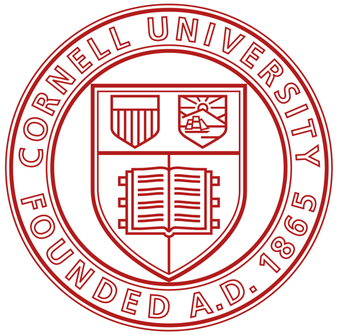
Basics

Roll a dice

- Let's roll the virtual die
- You can simulate a roll of the die with R's `sample` function.

Basics

`sample` function



Console | Terminal

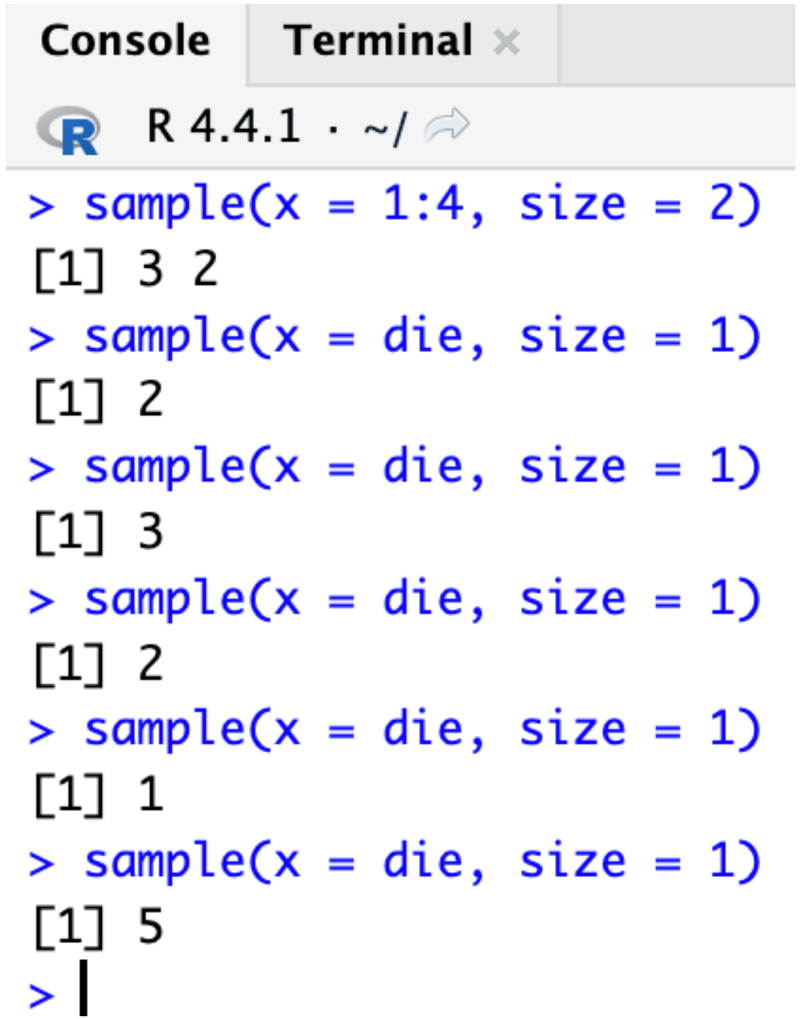
R 4.4.1 · ~/

```
> sample(x = 1:4, size = 2)
[1] 3 2
> sample(x = die, size = 1)
[1] 2
> sample(x = die, size = 1)
[1] 3
> sample(x = die, size = 1)
[1] 2
> sample(x = die, size = 1)
[1] 1
> sample(x = die, size = 1)
[1] 5
>
```

Basics

`sample` function

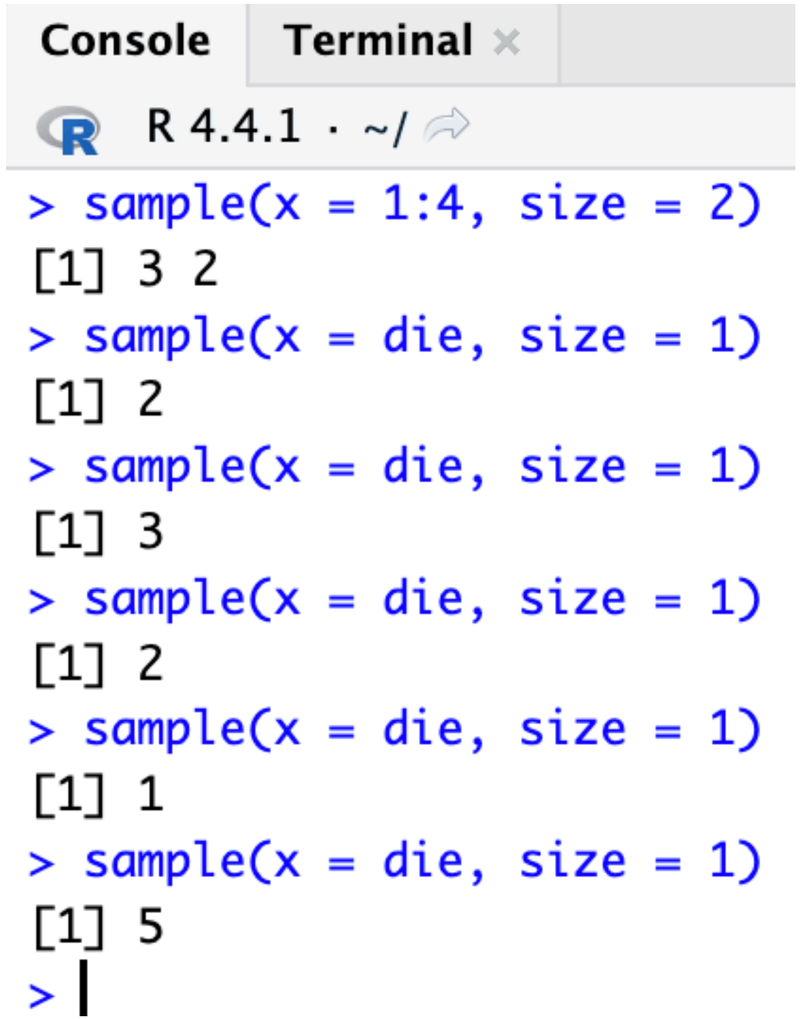
- `sample` takes *two* arguments: a vector named `x` and a number named `size`.



Basics

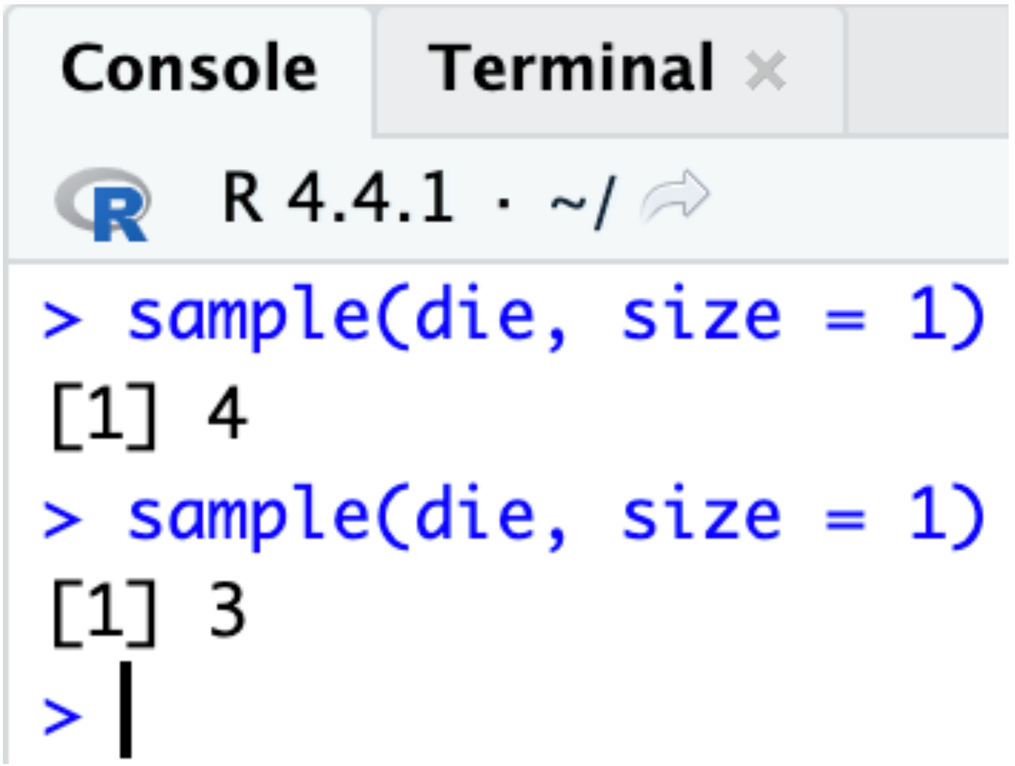
`sample` function

- `sample` takes *two* arguments: a vector named `x` and a number named `size`.
- `sample` will return `size` elements from the vector



Basics

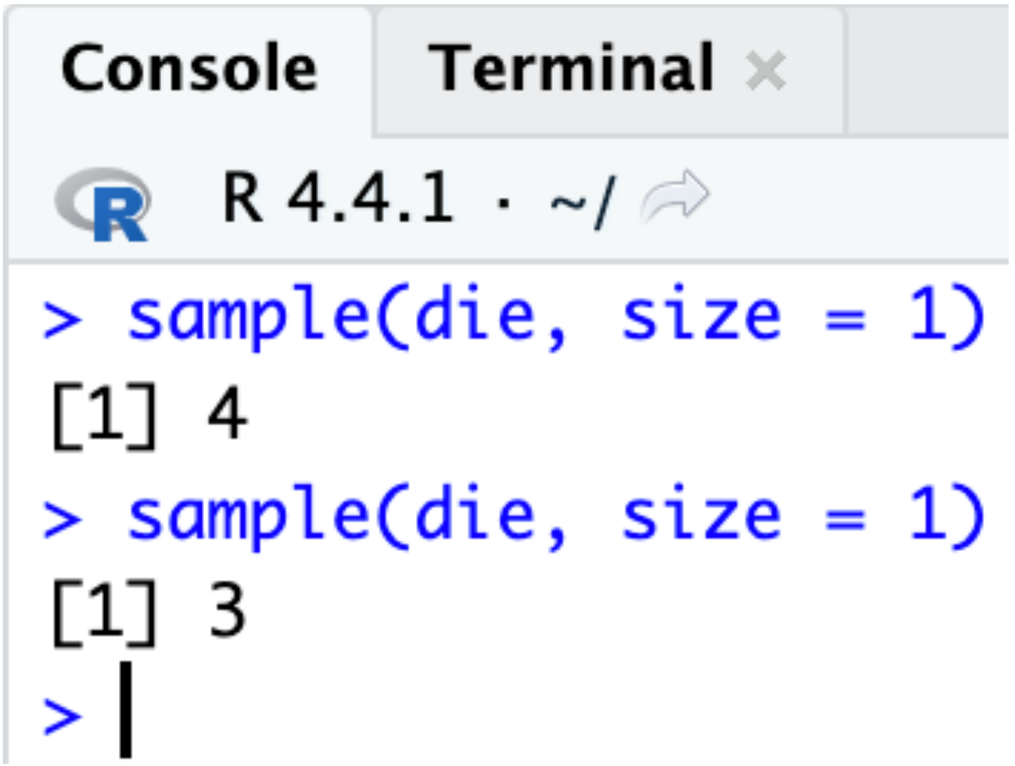
Arguments



Basics

Arguments

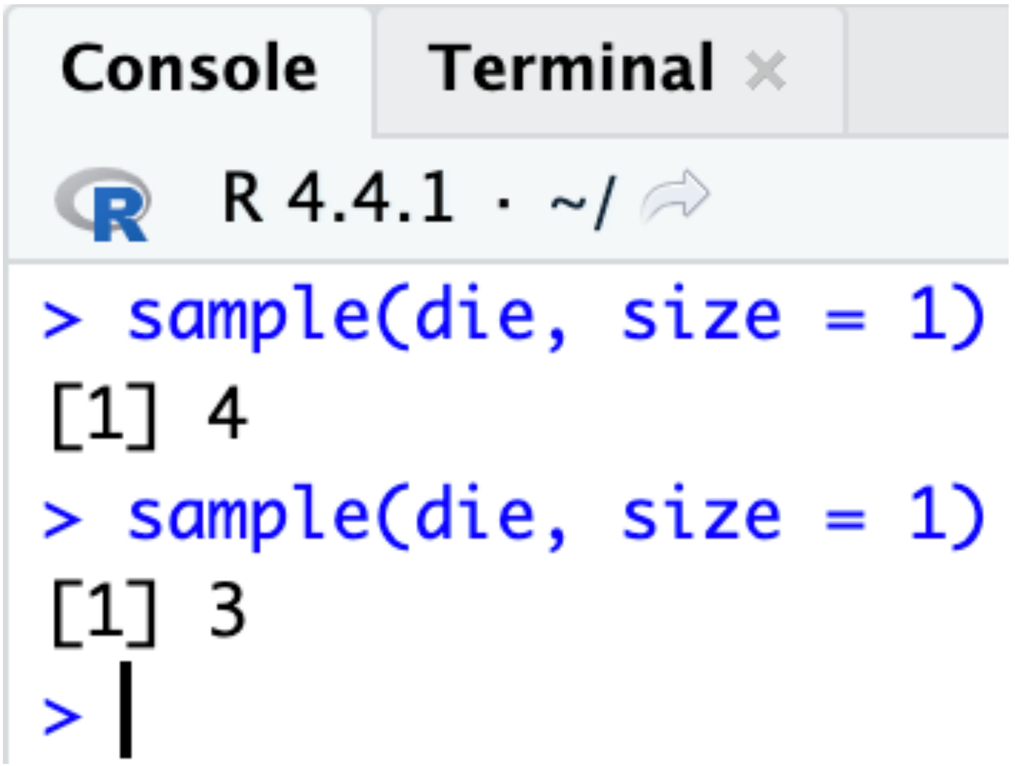
- Every argument in every R function has a name.



Basics

Arguments

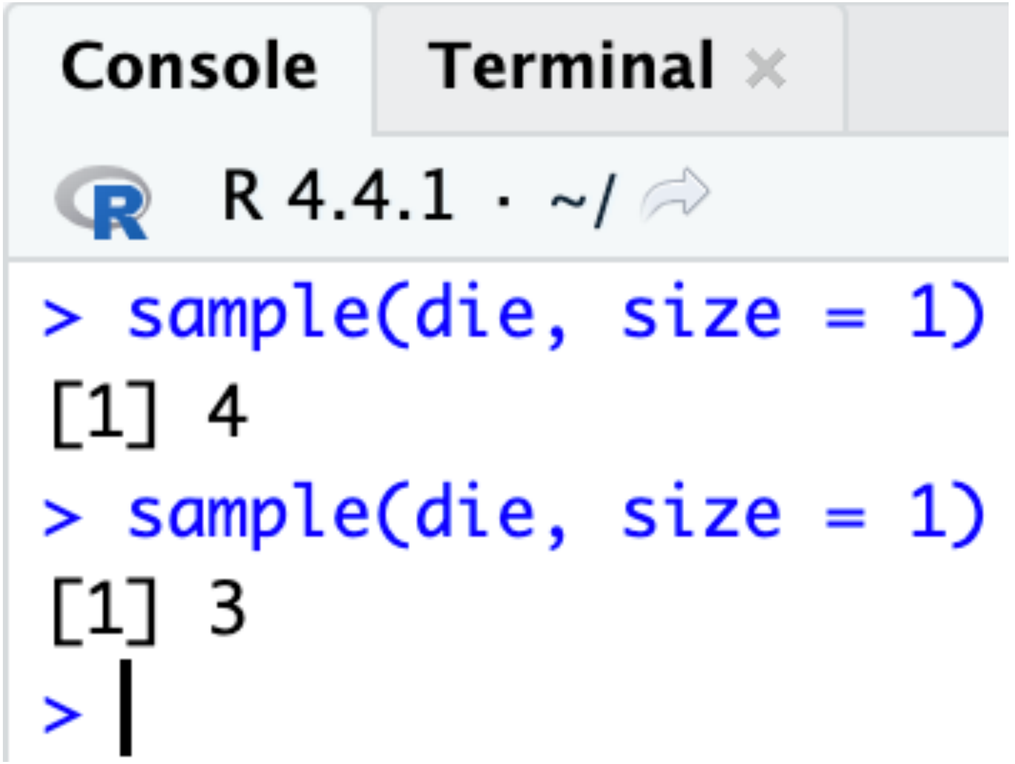
- Every argument in every R function has a name.
- You can specify which data should be assigned to which argument by setting a name equal to data.



Basics

Arguments

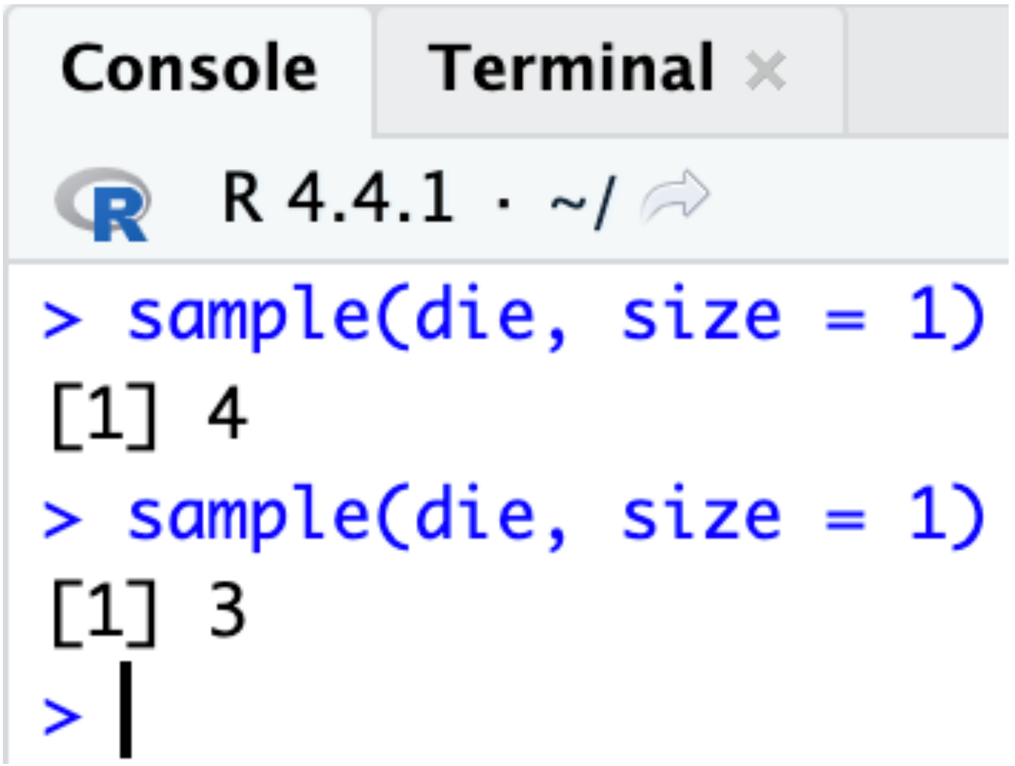
- Every argument in every R function has a name.
- You can specify which data should be assigned to which argument by setting a name equal to data.
- Names help you avoid passing the wrong data to the wrong argument.



Basics

Arguments

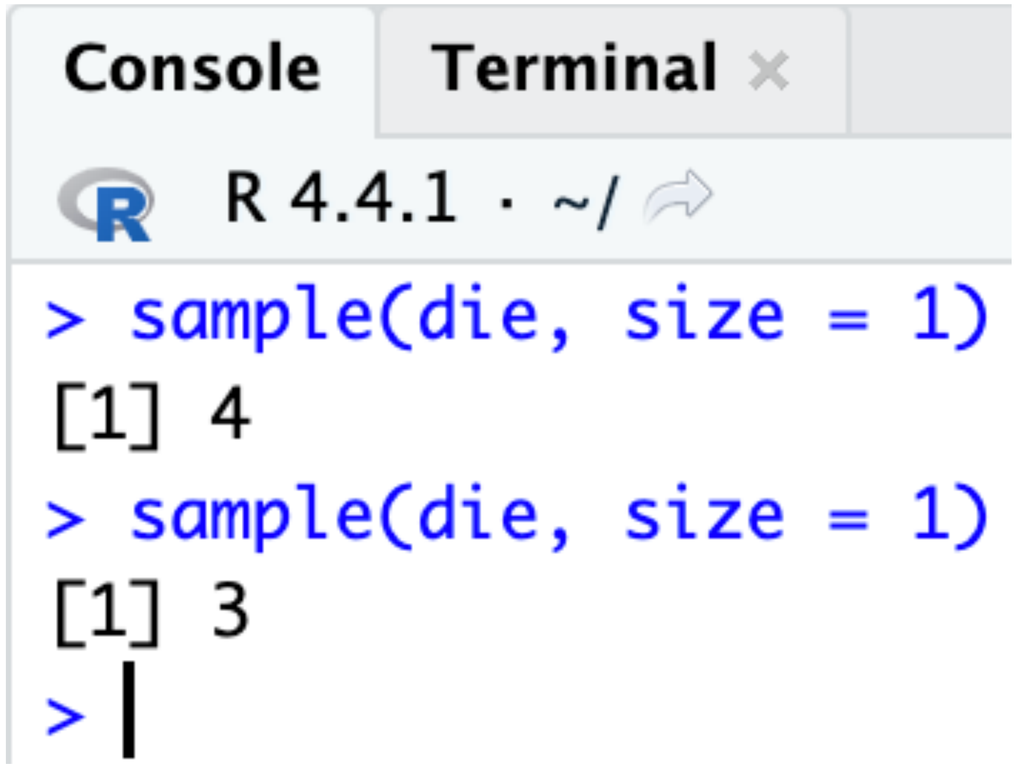
- Every argument in every R function has a name.
- You can specify which data should be assigned to which argument by setting a name equal to data.
- Names help you avoid passing the wrong data to the wrong argument.
- Using names is optional in R.



Basics

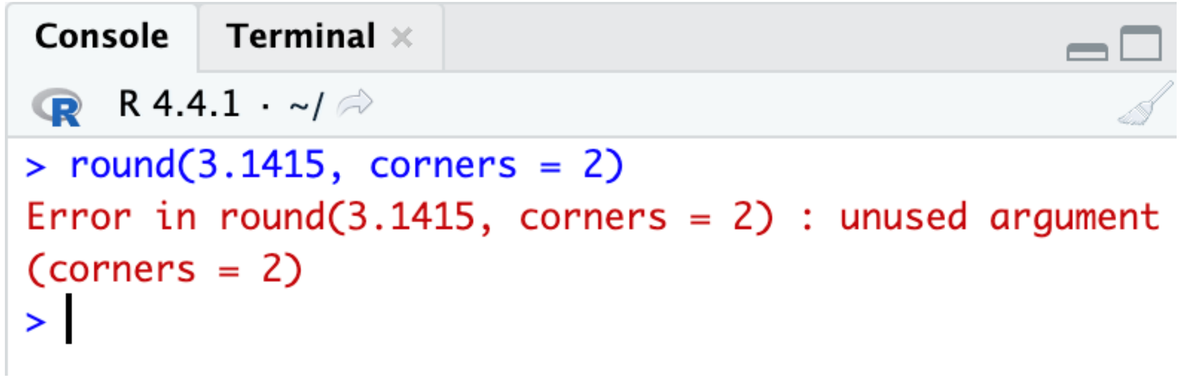
Arguments

- Every argument in every R function has a name.
- You can specify which data should be assigned to which argument by setting a name equal to data.
- Names help you avoid passing the wrong data to the wrong argument.
- Using names is optional in R.
- R users do not often use the name of the first argument in a function



Basics

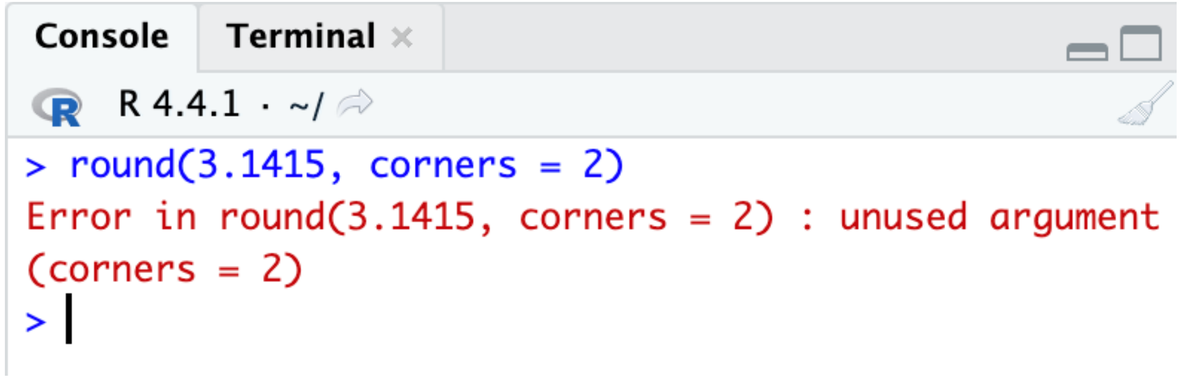
Arguments



Basics

Arguments

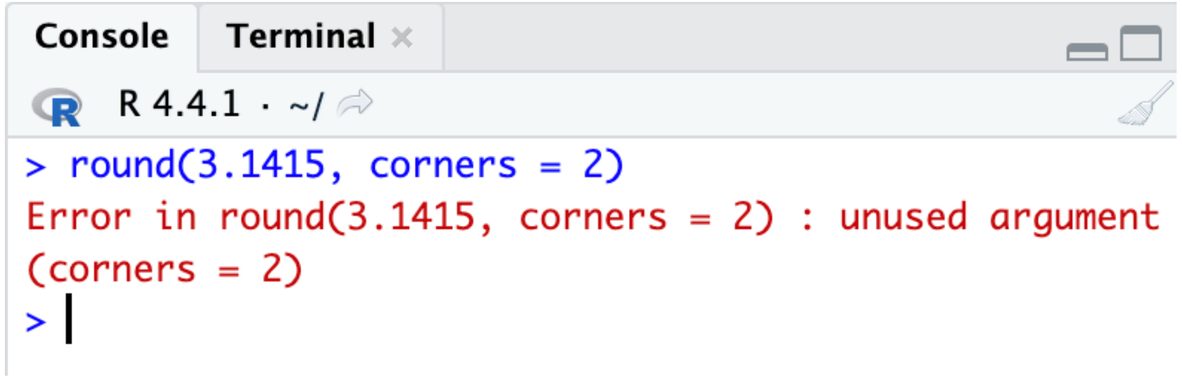
- How do you know which argument names to use?



Basics

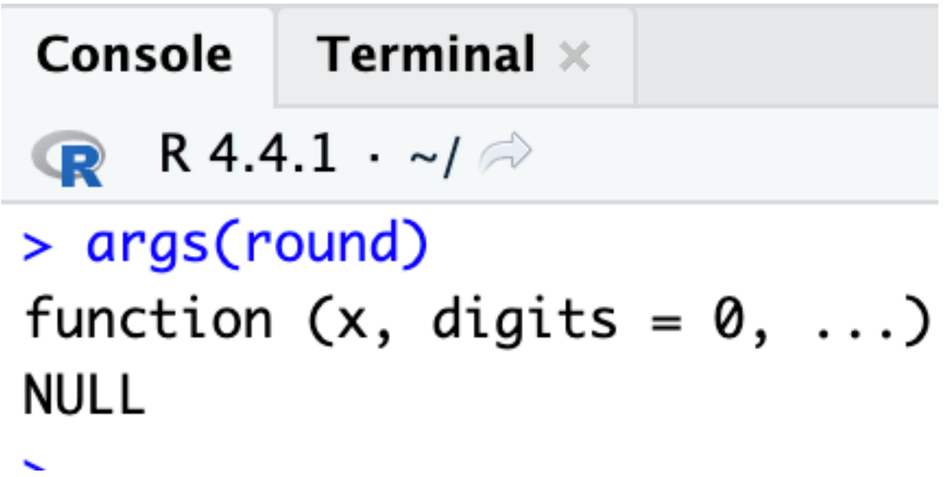
Arguments

- How do you know which argument names to use?
- If you try to use a name that a function does not expect, you will likely get an error.



Basics

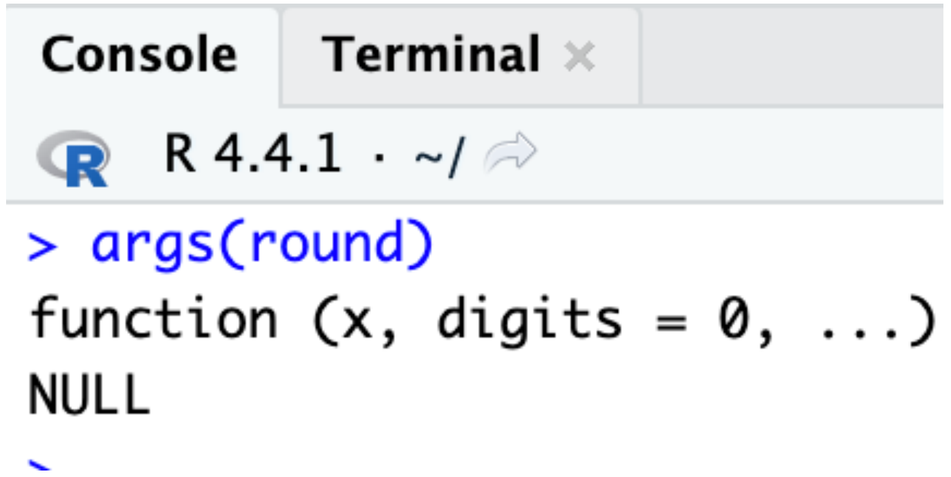
Arguments



Basics

Arguments

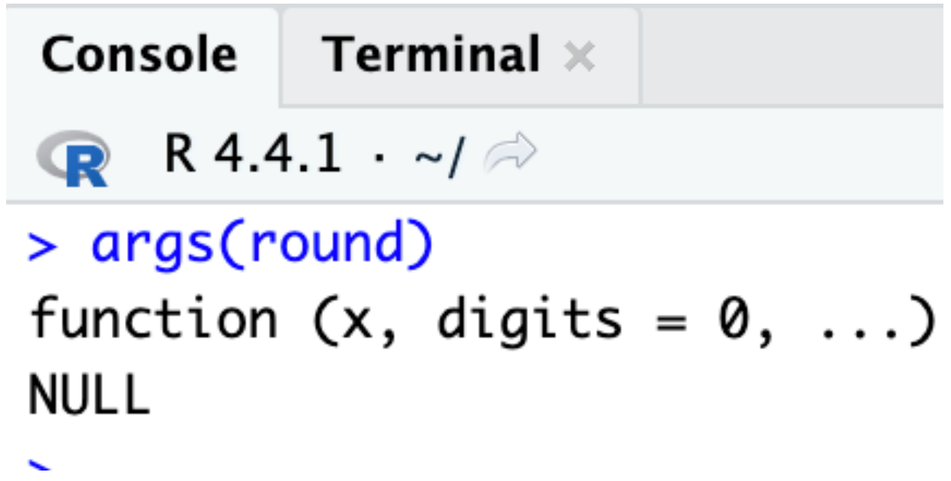
- If you're not sure which names to use with a function, you can look up the function's arguments with `args`.



Basics

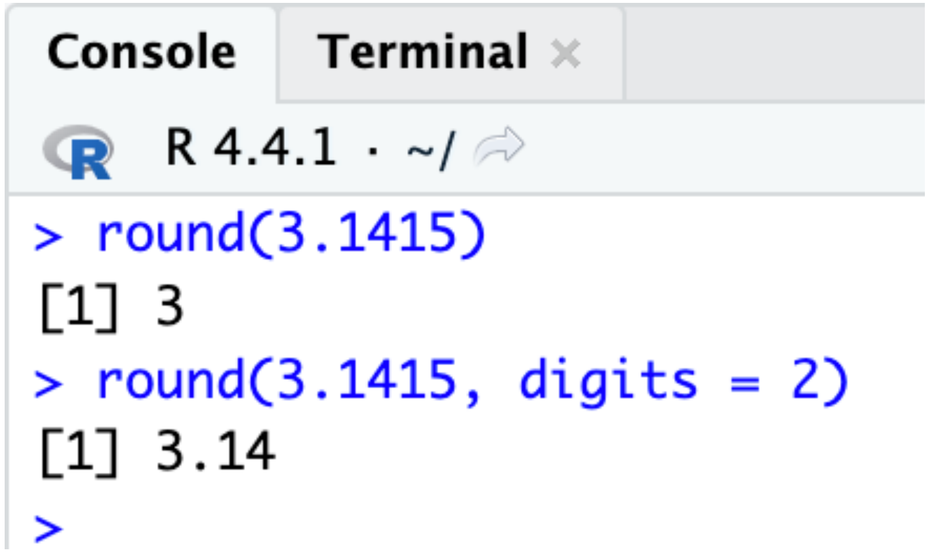
Arguments

- If you're not sure which names to use with a function, you can look up the function's arguments with `args`.
- To do this, place the name of the function in the parentheses behind `args`



Basics

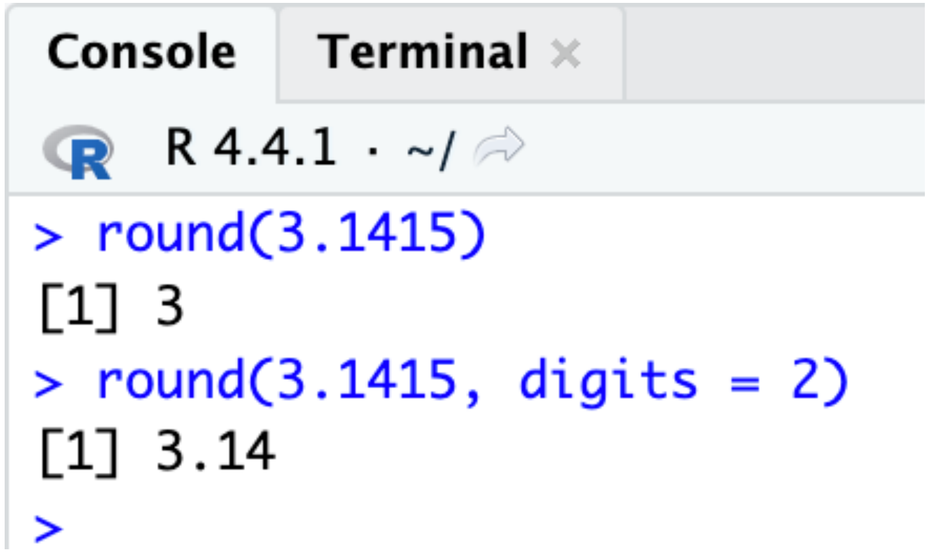
Arguments



Basics

Arguments

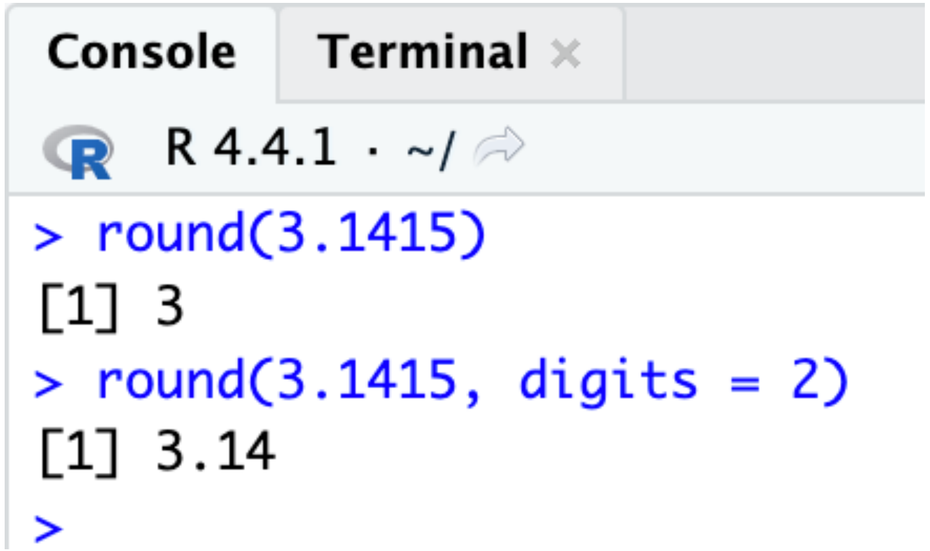
- Did you notice that `args` shows that the `digits` argument of `round` is already set to 0?



Basics

Arguments

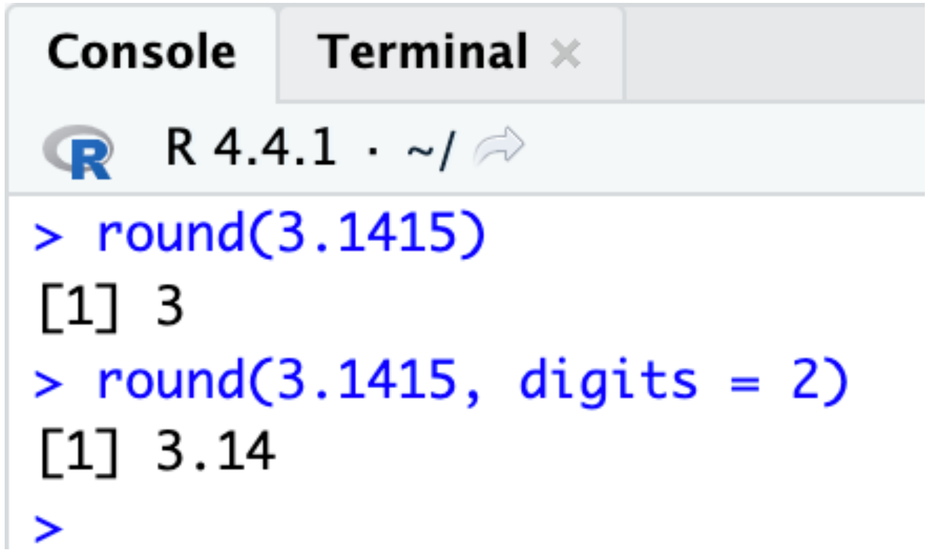
- Did you notice that `args` shows that the `digits` argument of `round` is already set to 0?
- Frequently, an R function will take optional arguments like `digits`.



Basics

Arguments

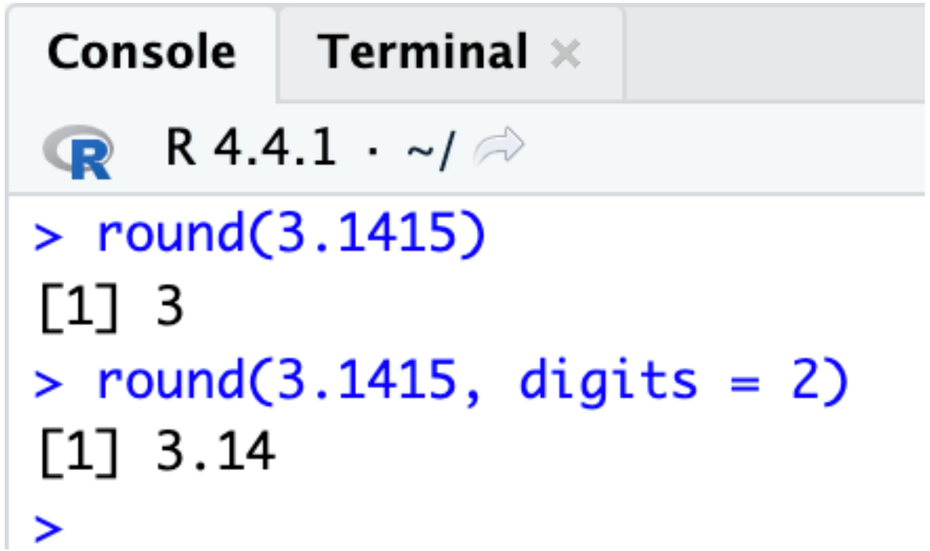
- Did you notice that `args` shows that the `digits` argument of `round` is already set to 0?
- Frequently, an R function will take optional arguments like `digits`.
- These arguments are considered optional because they come with a default value.



Basics

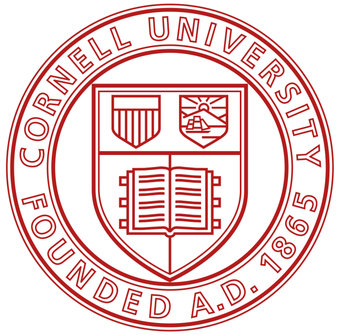
Arguments

- Did you notice that `args` shows that the `digits` argument of `round` is already set to 0?
- Frequently, an R function will take optional arguments like `digits`.
- These arguments are considered optional because they come with a default value.
- You can pass a new value to an optional argument if you want, and R will use the default value if you do not.



Basics

Sample with replacement



Console | Terminal

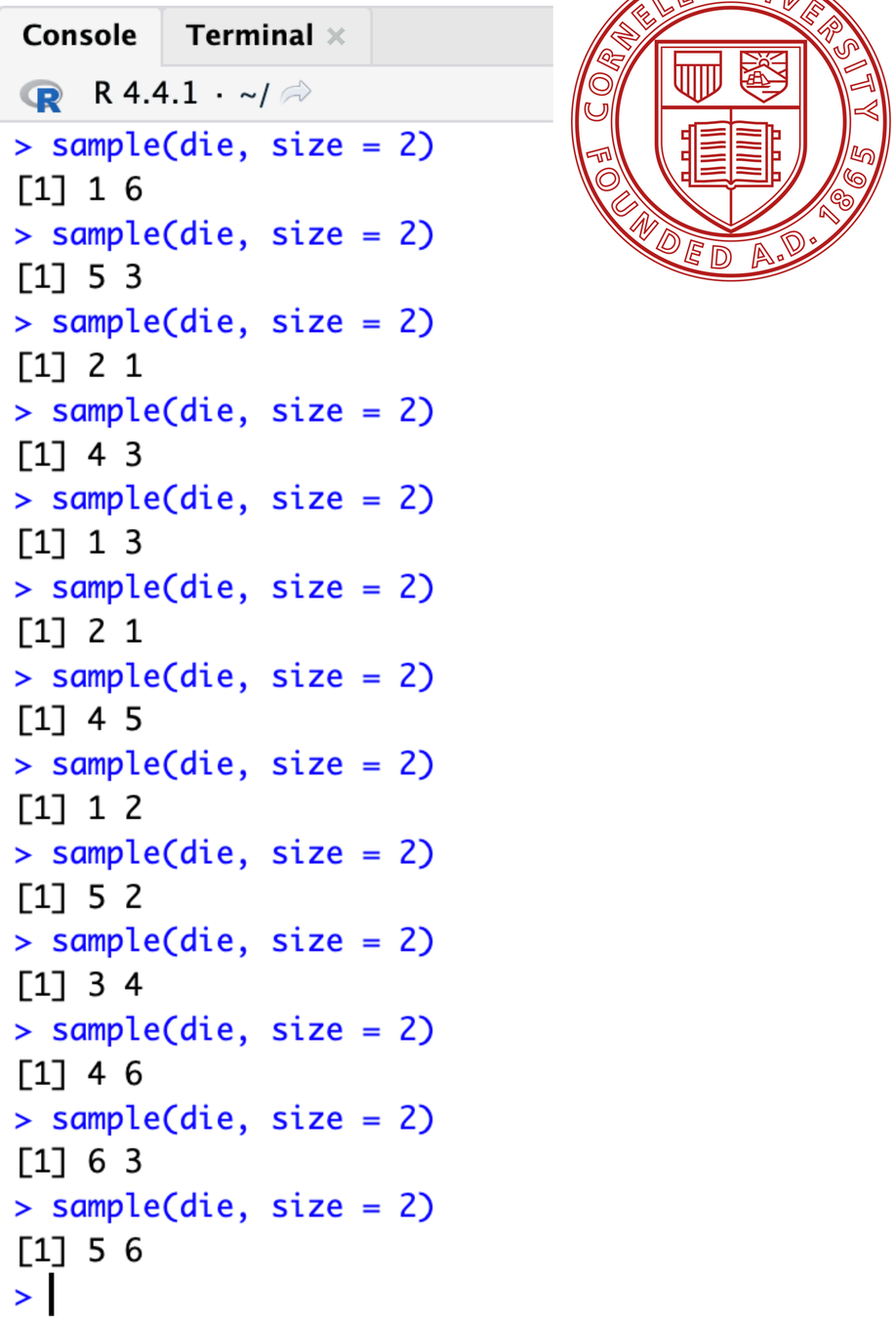
R 4.4.1 · ~/

```
> sample(die, size = 2)
[1] 1 6
> sample(die, size = 2)
[1] 5 3
> sample(die, size = 2)
[1] 2 1
> sample(die, size = 2)
[1] 4 3
> sample(die, size = 2)
[1] 1 3
> sample(die, size = 2)
[1] 2 1
> sample(die, size = 2)
[1] 4 5
> sample(die, size = 2)
[1] 1 2
> sample(die, size = 2)
[1] 5 2
> sample(die, size = 2)
[1] 3 4
> sample(die, size = 2)
[1] 4 6
> sample(die, size = 2)
[1] 6 3
> sample(die, size = 2)
[1] 5 6
>
```

Basics

Sample with replacement

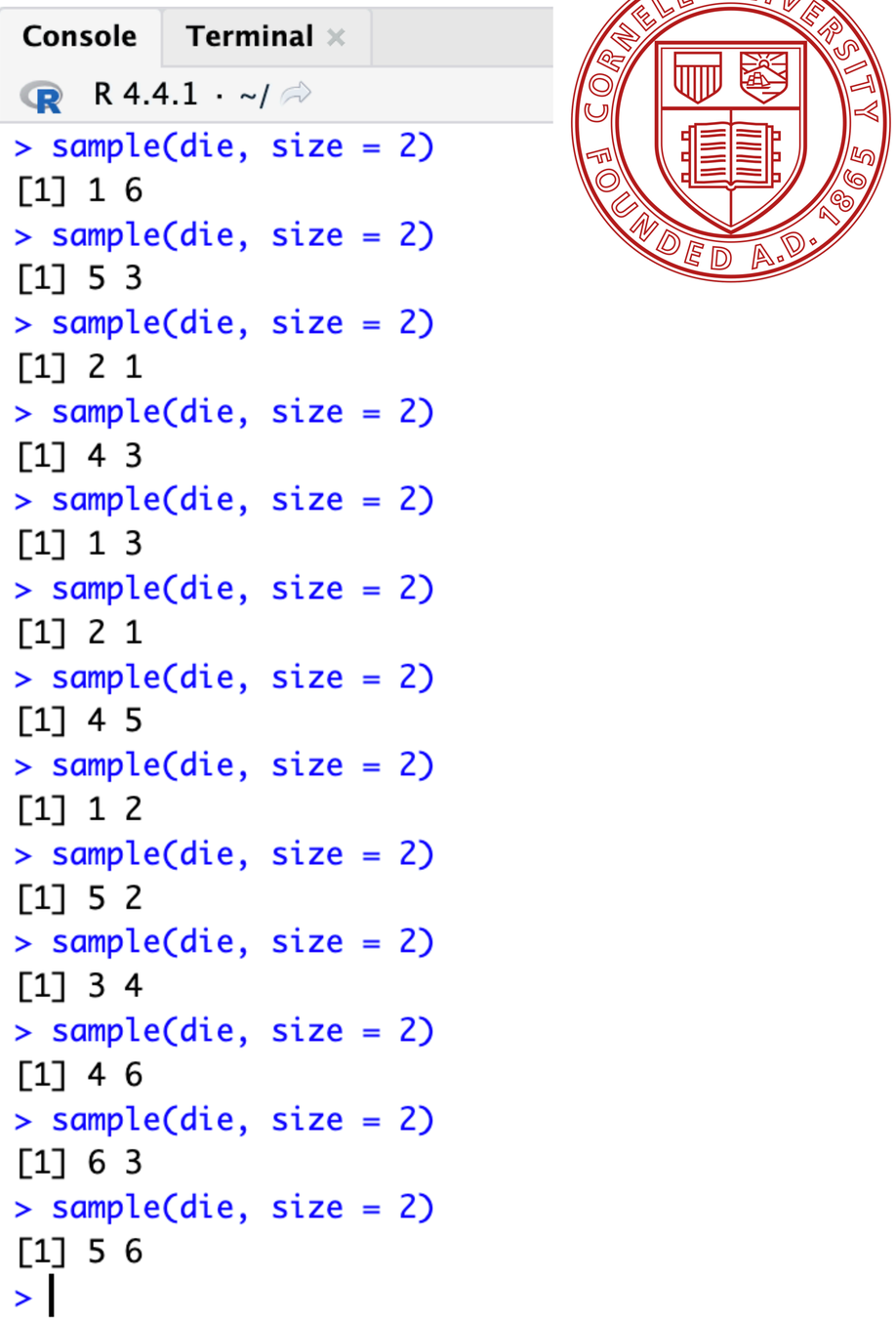
- With `sample` if you set `size = 2`, you can *almost* simulate a pair of dice.



Basics

Sample with replacement

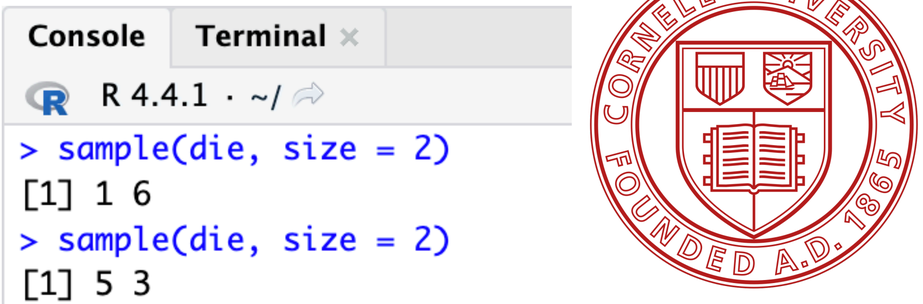
- With `sample` if you set `size = 2`, you can *almost* simulate a pair of dice.
- "Almost" because if you use it many times, you'll notice that the second die never has the same value as the first die.



Basics

Sample with replacement

- With `sample` if you set `size = 2`, you can *almost* simulate a pair of dice.
- "Almost" because if you use it many times, you'll notice that the second die never has the same value as the first die.
- By default, `sample` builds a sample without replacement.



Console | Terminal

R 4.4.1 · ~/

```
> sample(die, size = 2)
[1] 1 6
> sample(die, size = 2)
[1] 5 3
> sample(die, size = 2)
[1] 2 1
> sample(die, size = 2)
[1] 4 3
> sample(die, size = 2)
[1] 1 3
> sample(die, size = 2)
[1] 2 1
> sample(die, size = 2)
[1] 4 5
> sample(die, size = 2)
[1] 1 2
> sample(die, size = 2)
[1] 5 2
> sample(die, size = 2)
[1] 3 4
> sample(die, size = 2)
[1] 4 6
> sample(die, size = 2)
[1] 6 3
> sample(die, size = 2)
[1] 5 6
>
```