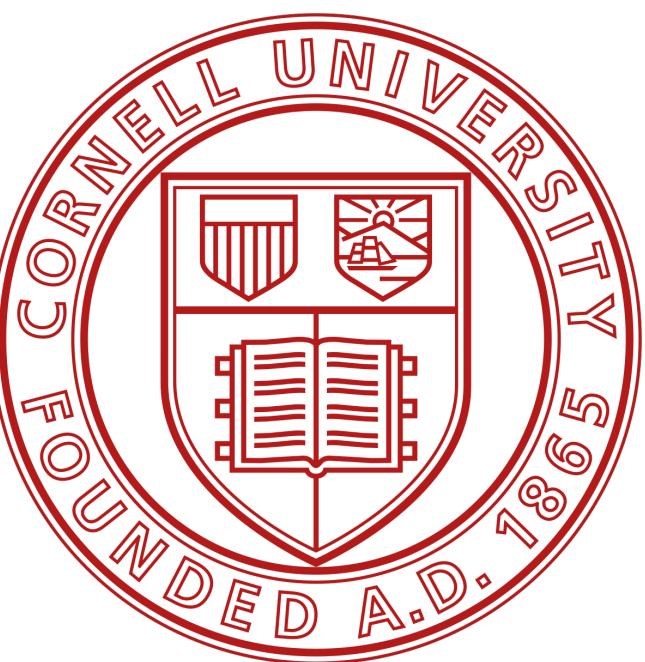


Introduction to R programming

STSCI 2120

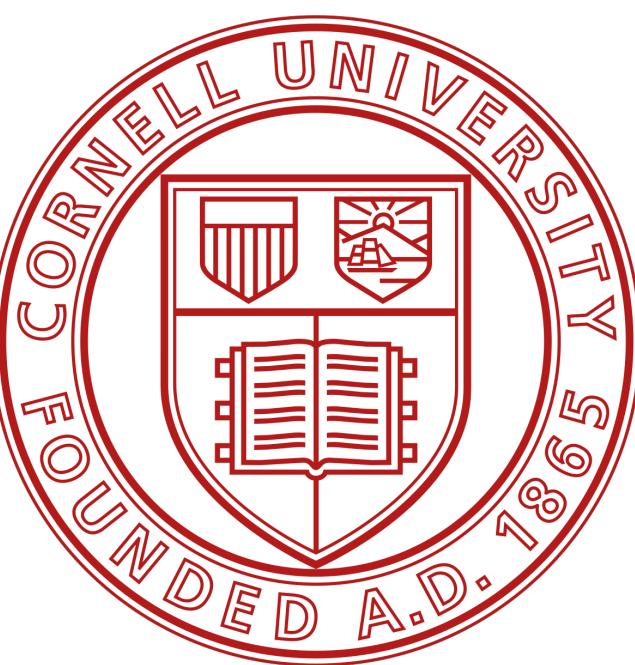
Nayel Bettache - Fall 2024

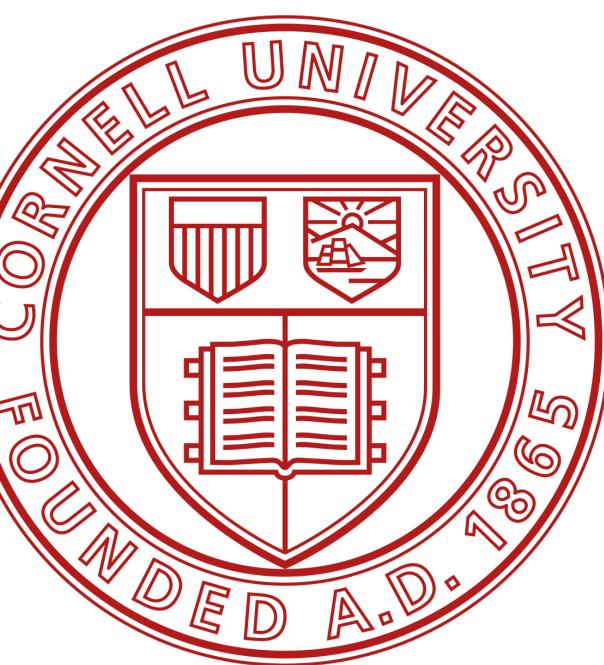


Personal Presentation

Nayel Bettache

Visiting Assistant Professor



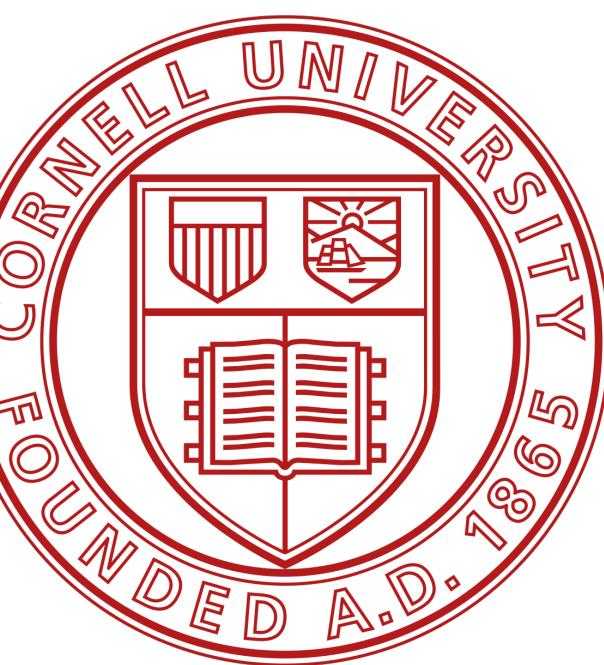


Nayel Bettache

Visiting Assistant Professor

- PhD in mathematical statistics from Institut Polytechnique de Paris



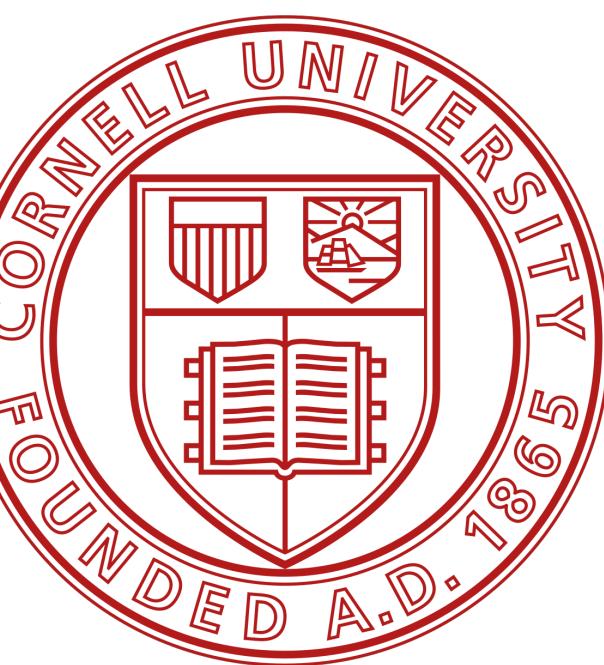


Nayel Bettache

Visiting Assistant Professor

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



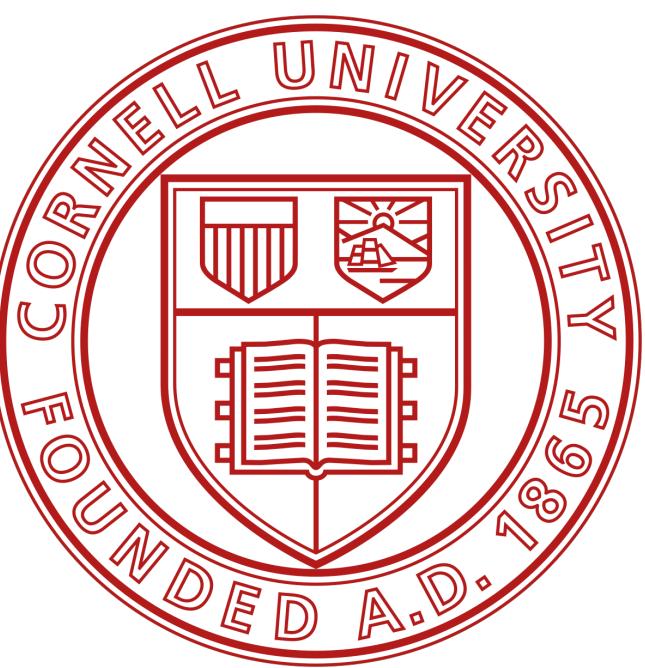


Nayel Bettache

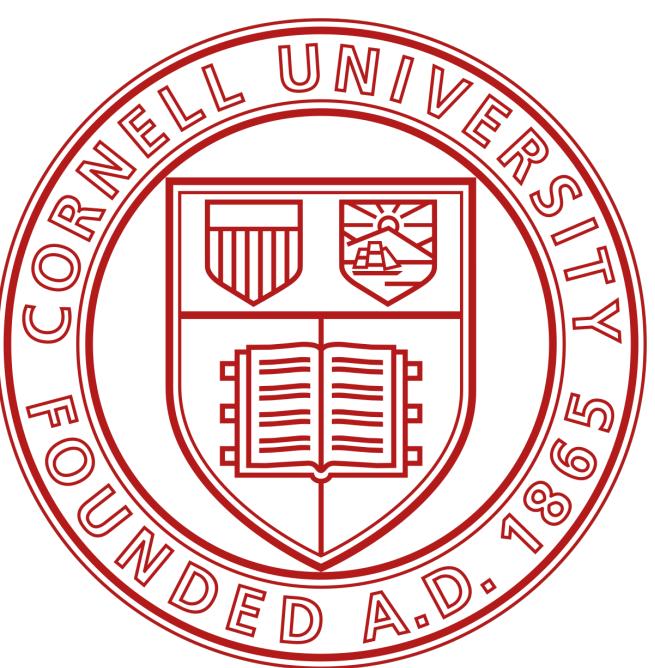
Visiting Assistant Professor

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





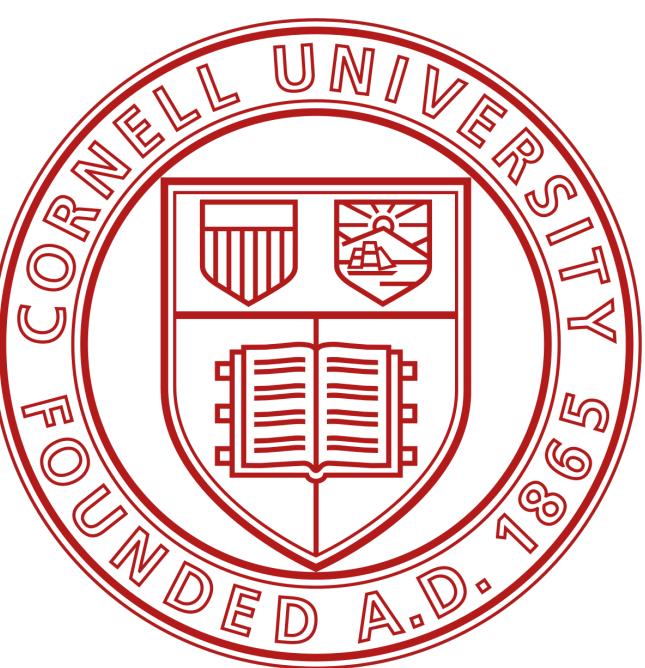
Overview of the course



What will you learn ?

Objectives



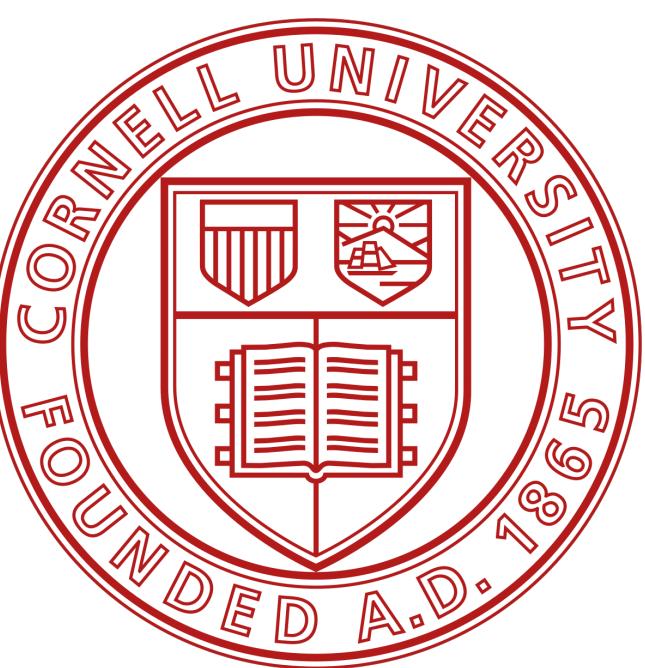


What will you learn ?

Objectives

- Help you learn the most important tools in R



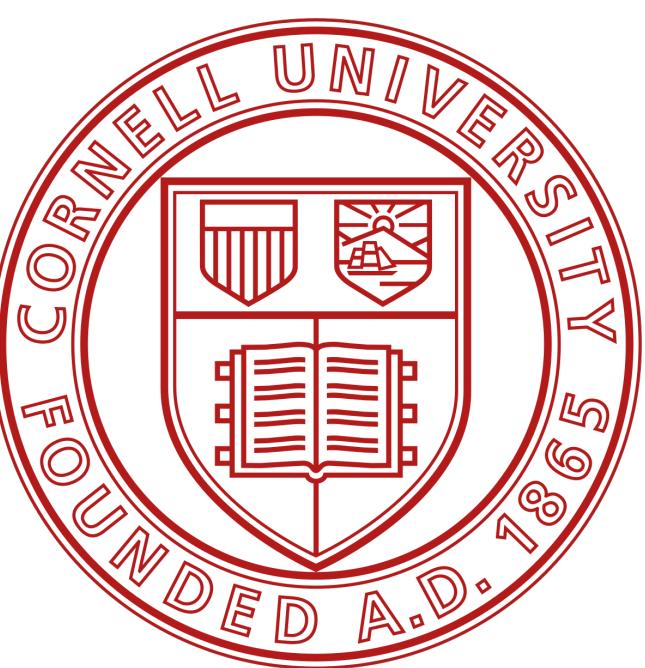


What will you learn ?

Objectives

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



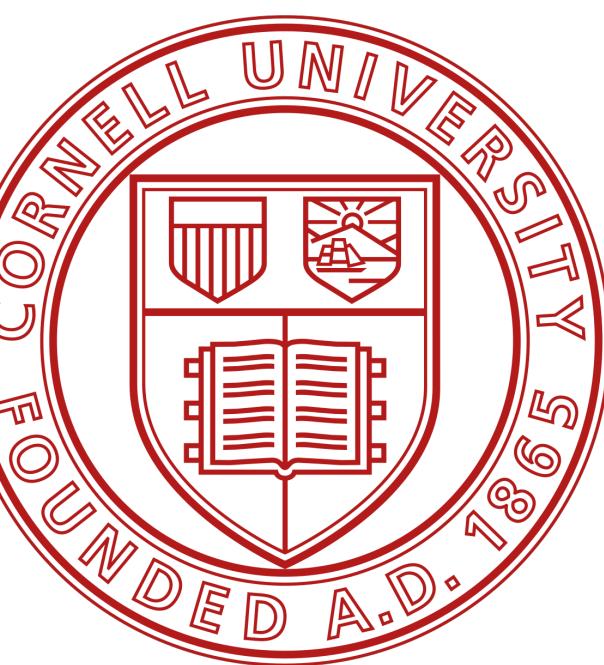


What will you learn ?

Objectives

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

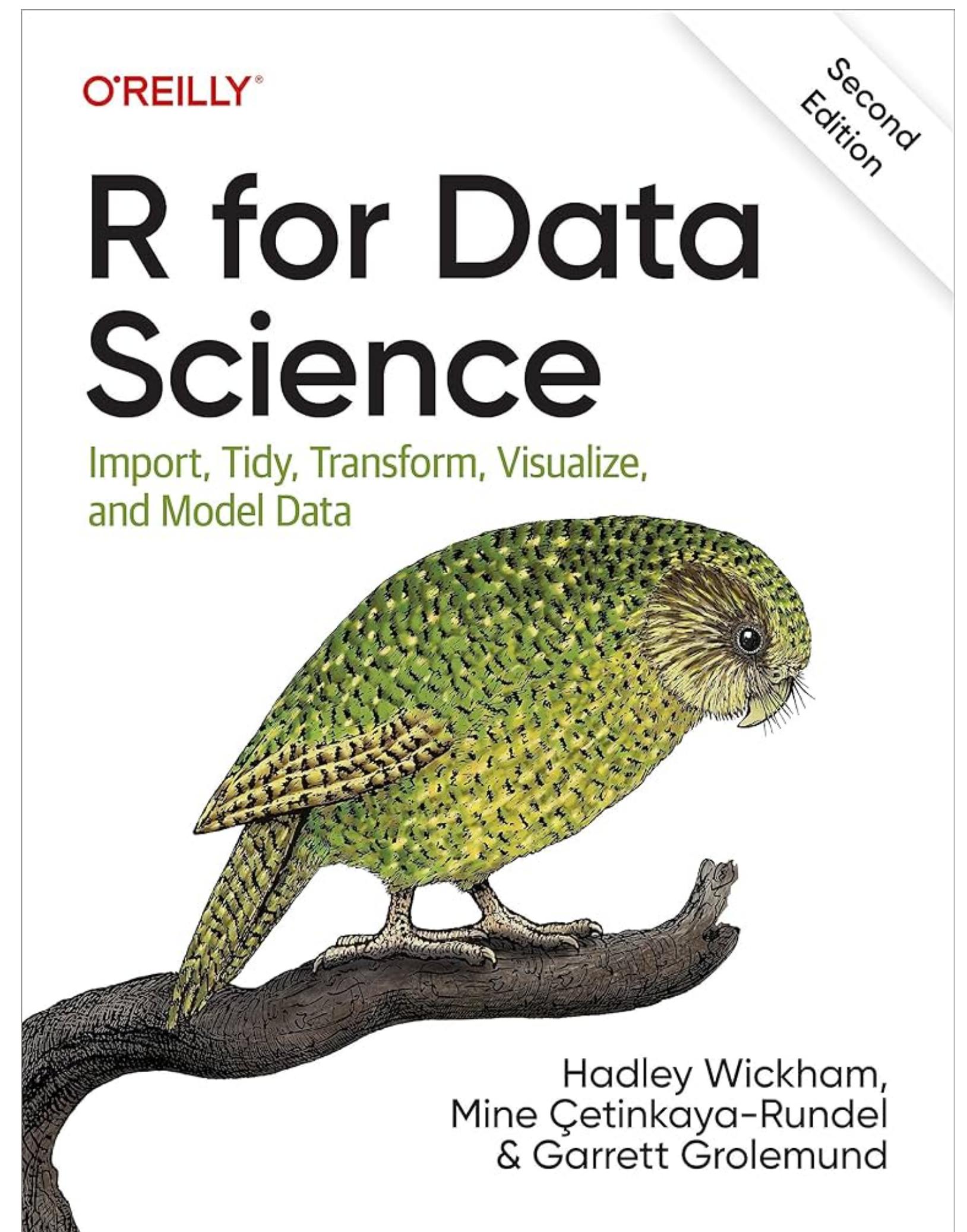


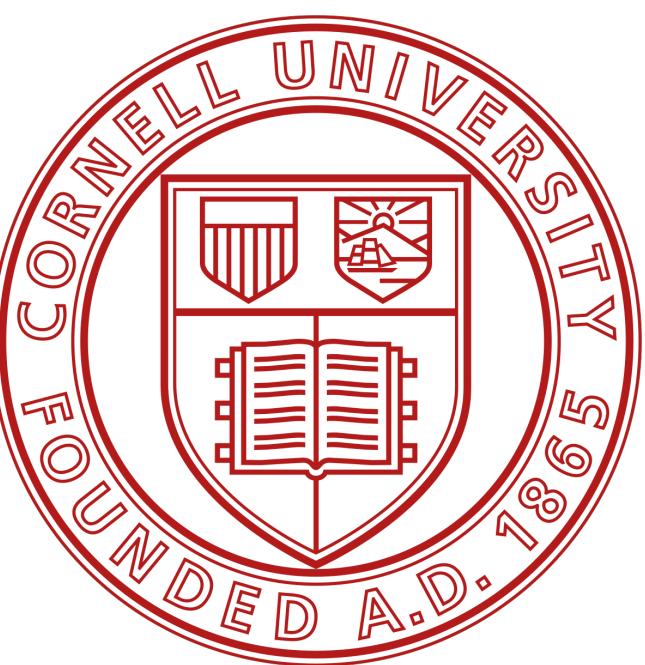


What will you learn ?

Ressources

- R for Data Science





What will you learn ?

Ressources

- R for Data Science
- Tidy modeling with R

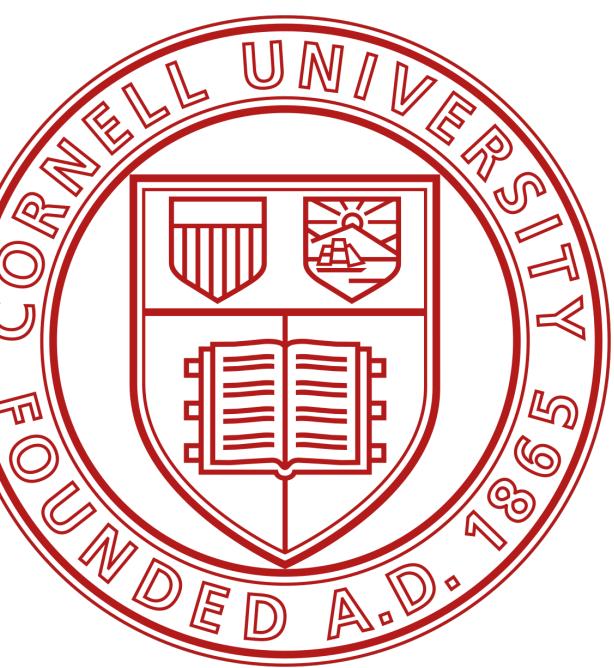
O'REILLY®

Tidy Modeling with R

A Framework for Modeling in the Tidyverse



Max Kuhn & Julia Silge

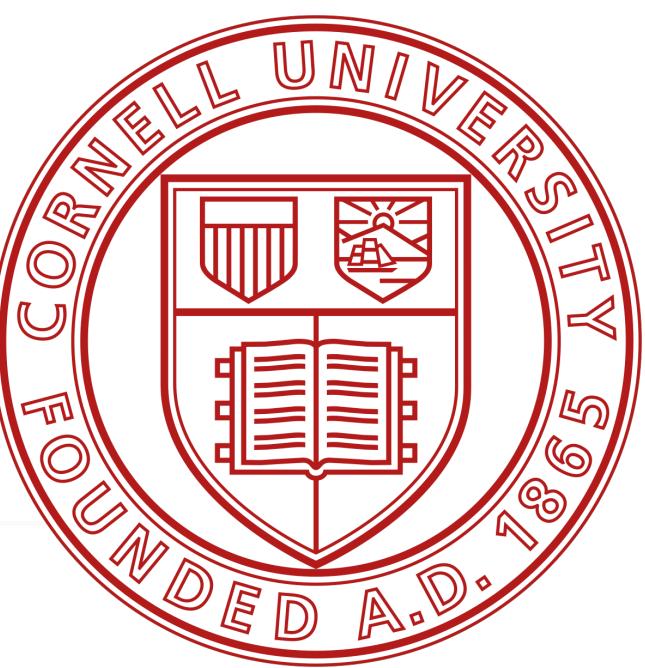


What will you learn ?

Ressources

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





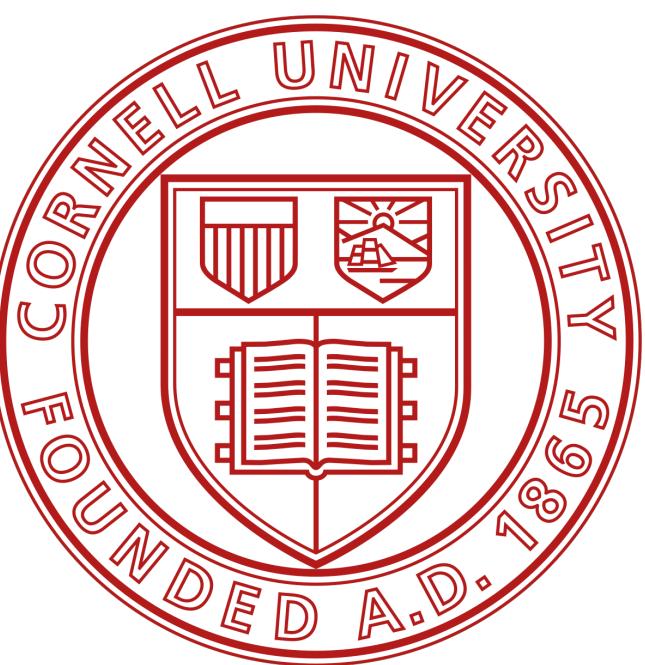
What will you learn ?

Ressources

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

An Introduction to R

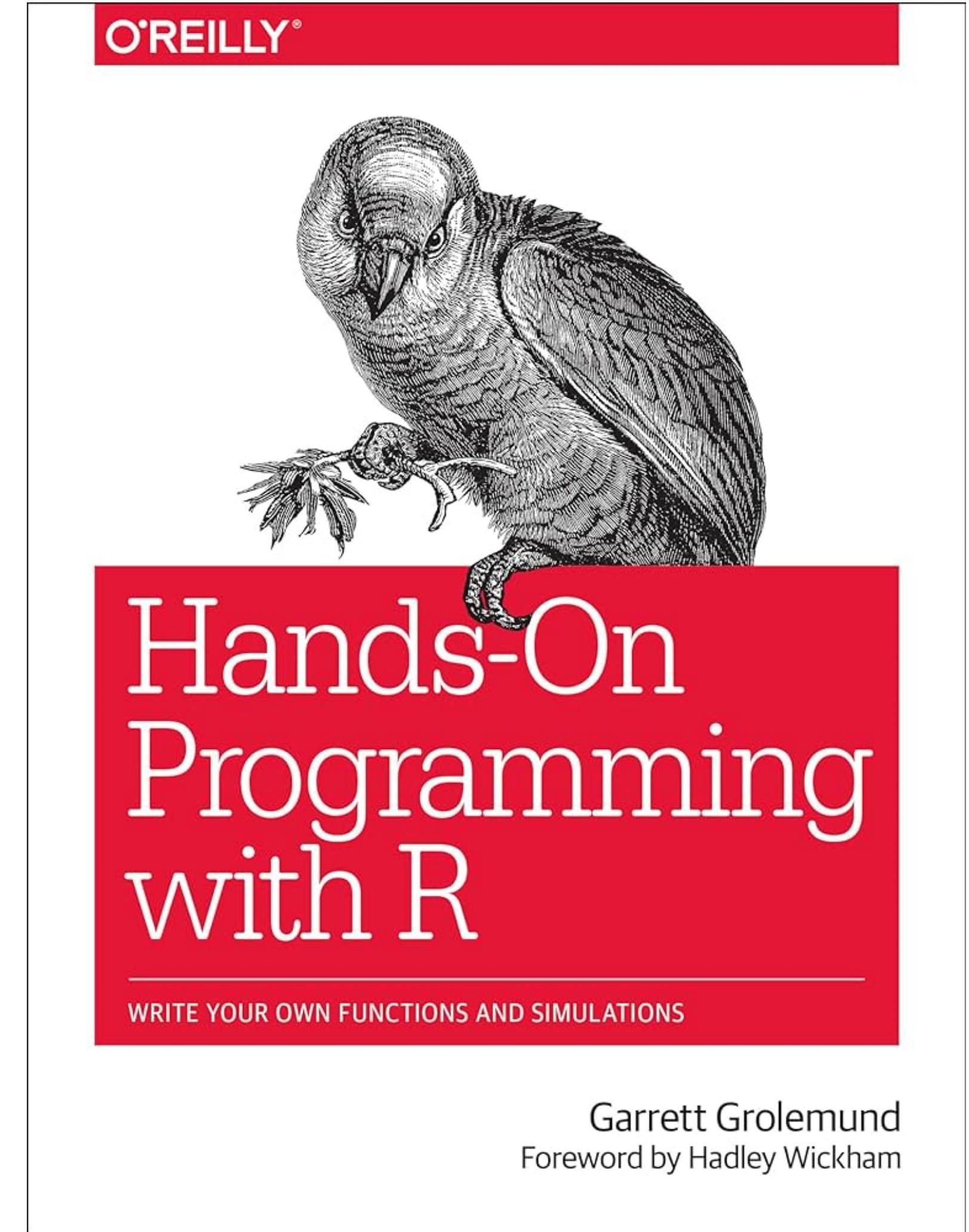
Notes on R: A Programming Environment for Data Analysis and Graphics
Version 4.4.1 (2024-06-14)

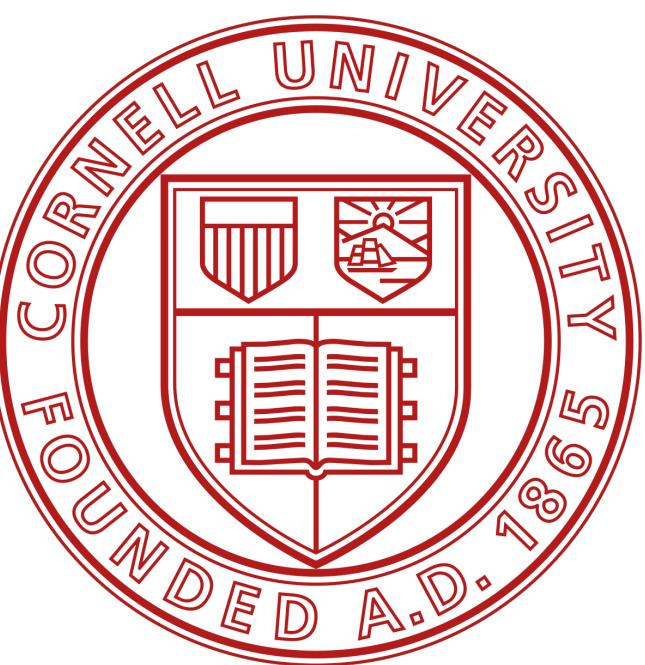


What will you learn ?

Ressources

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

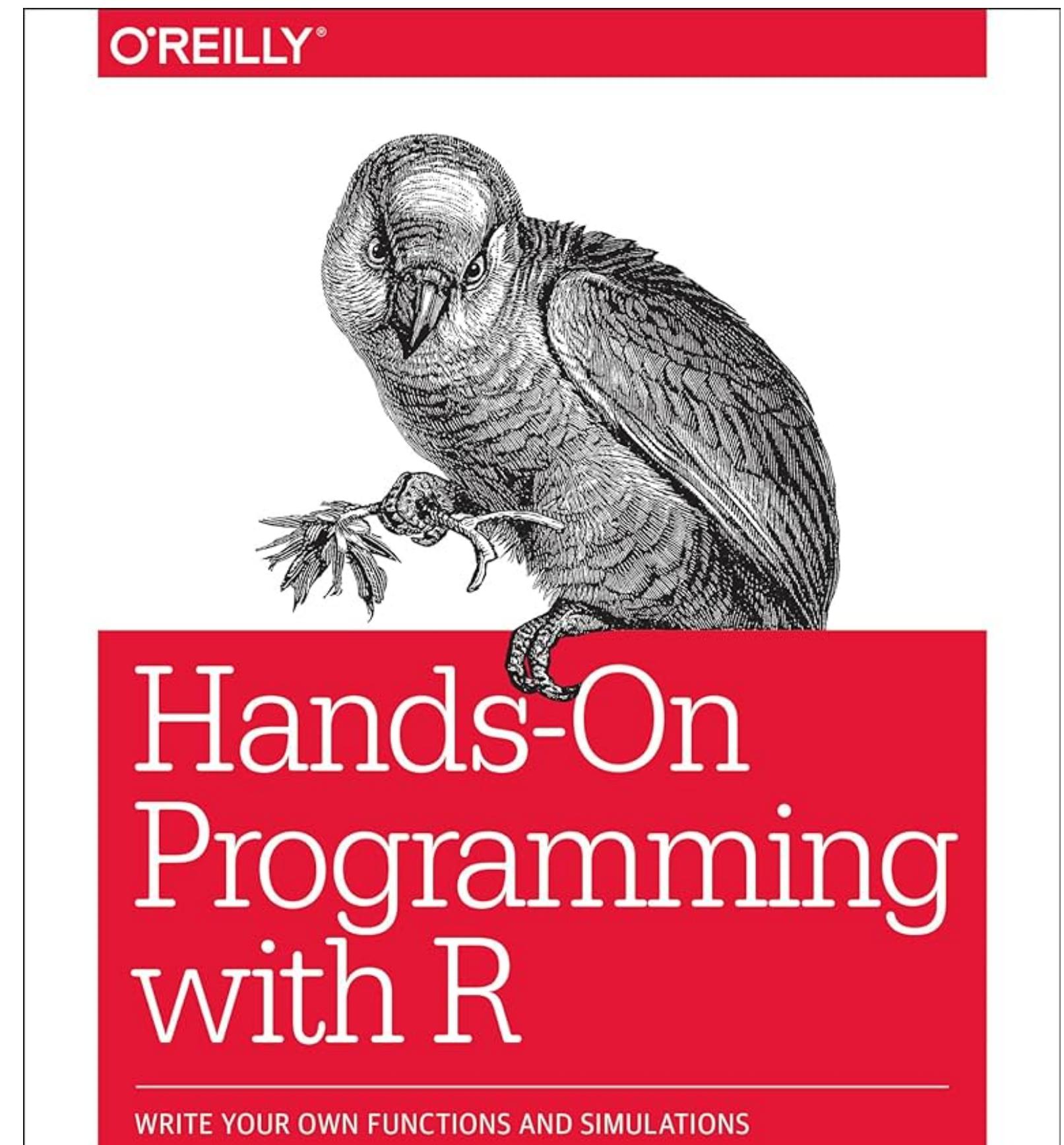




What will you learn ?

Main ressource

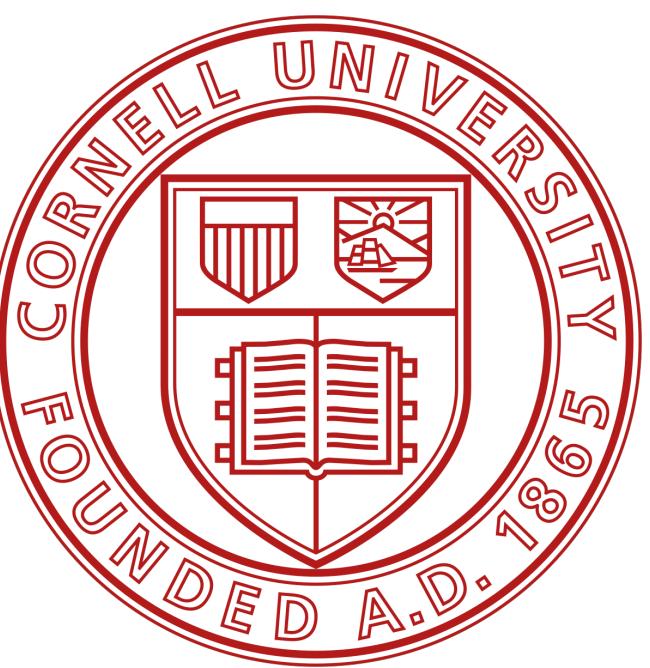
- Hands on Programming with R

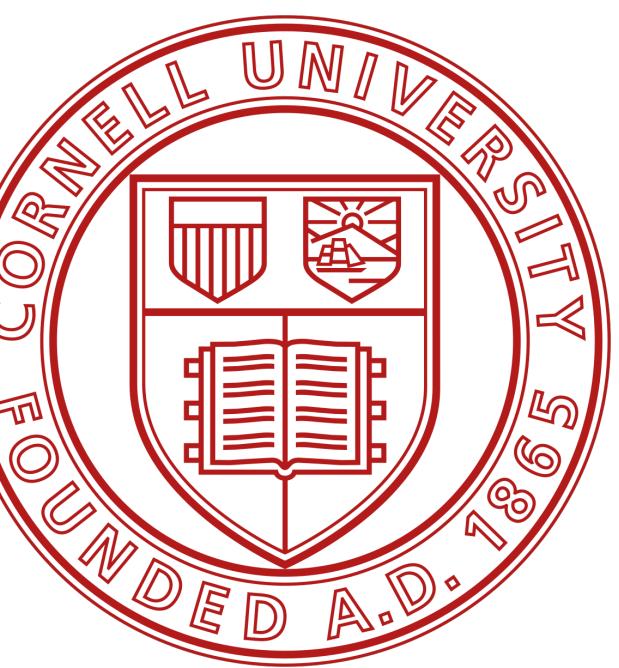


Garrett Grolemund
Foreword by Hadley Wickham

What will you learn ?

Details



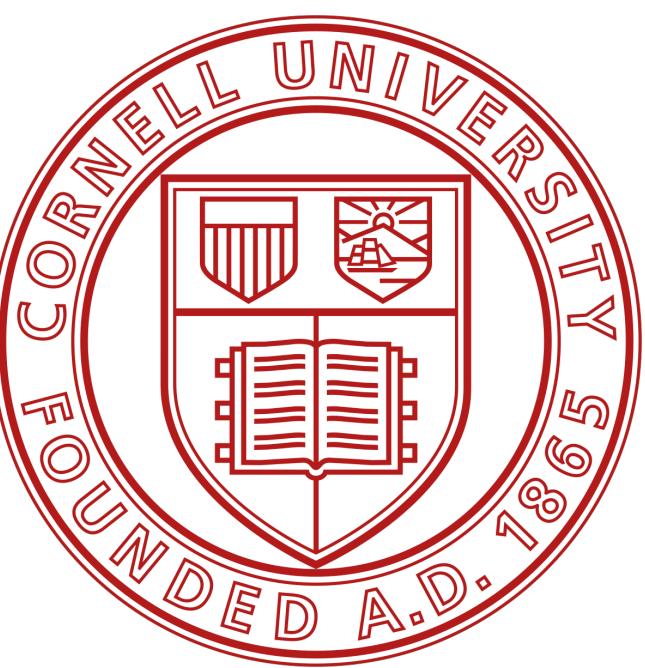


What will you learn ?

Details

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



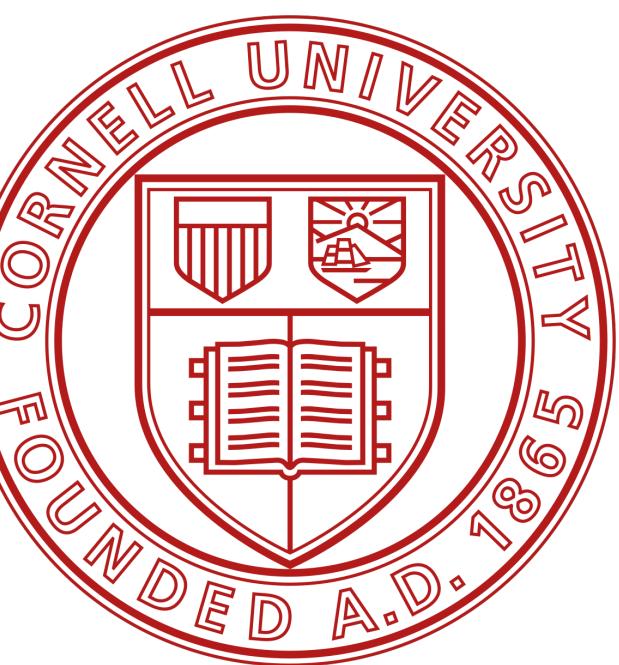


What will you learn ?

Details

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



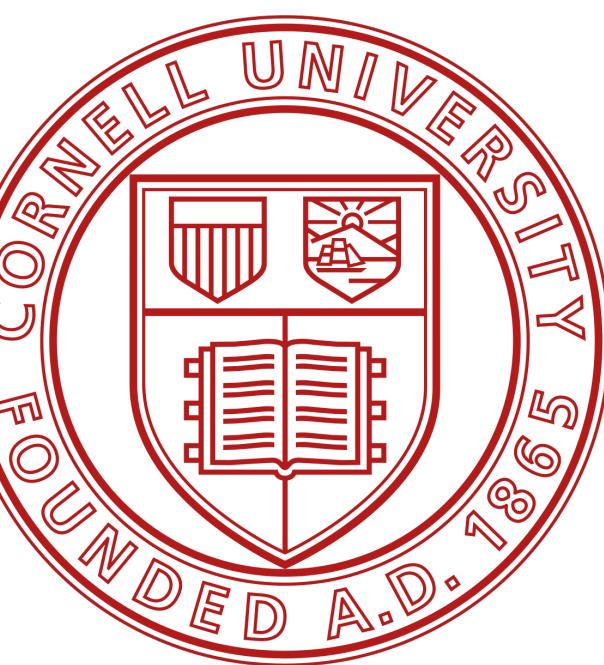


What will you learn ?

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.



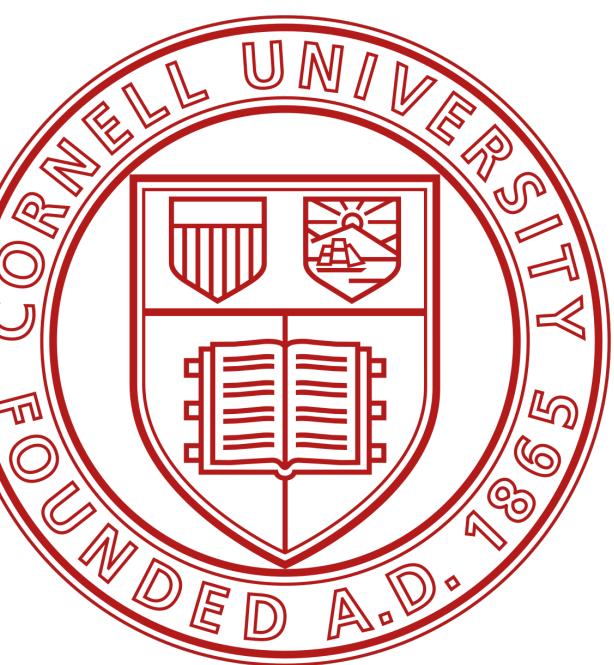


What will you learn ?

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



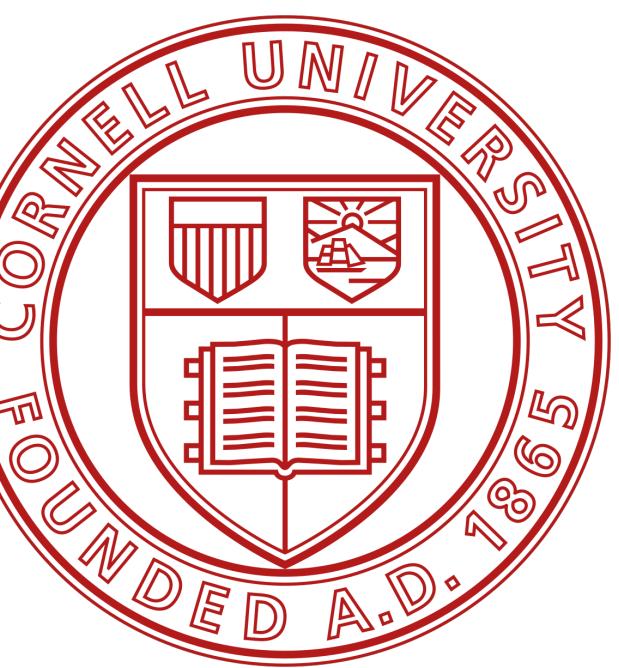


What won't you learn ?

Details

- Modeling
- Big data
- Python, Julia and others

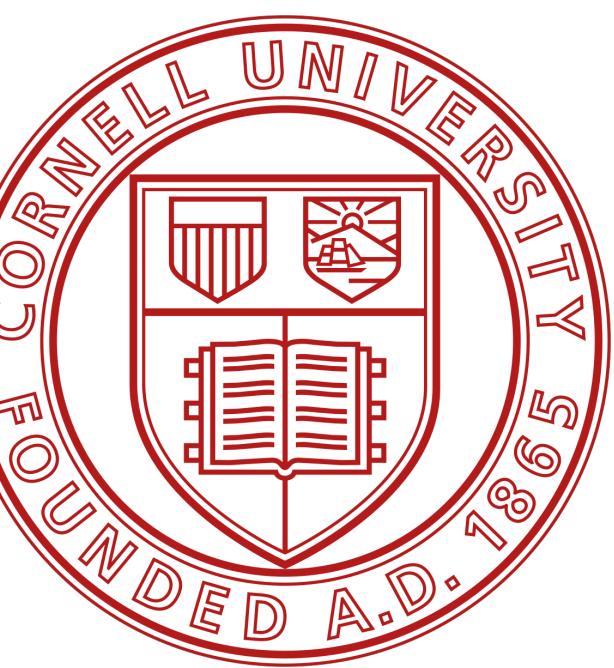




7 weeks

Overview

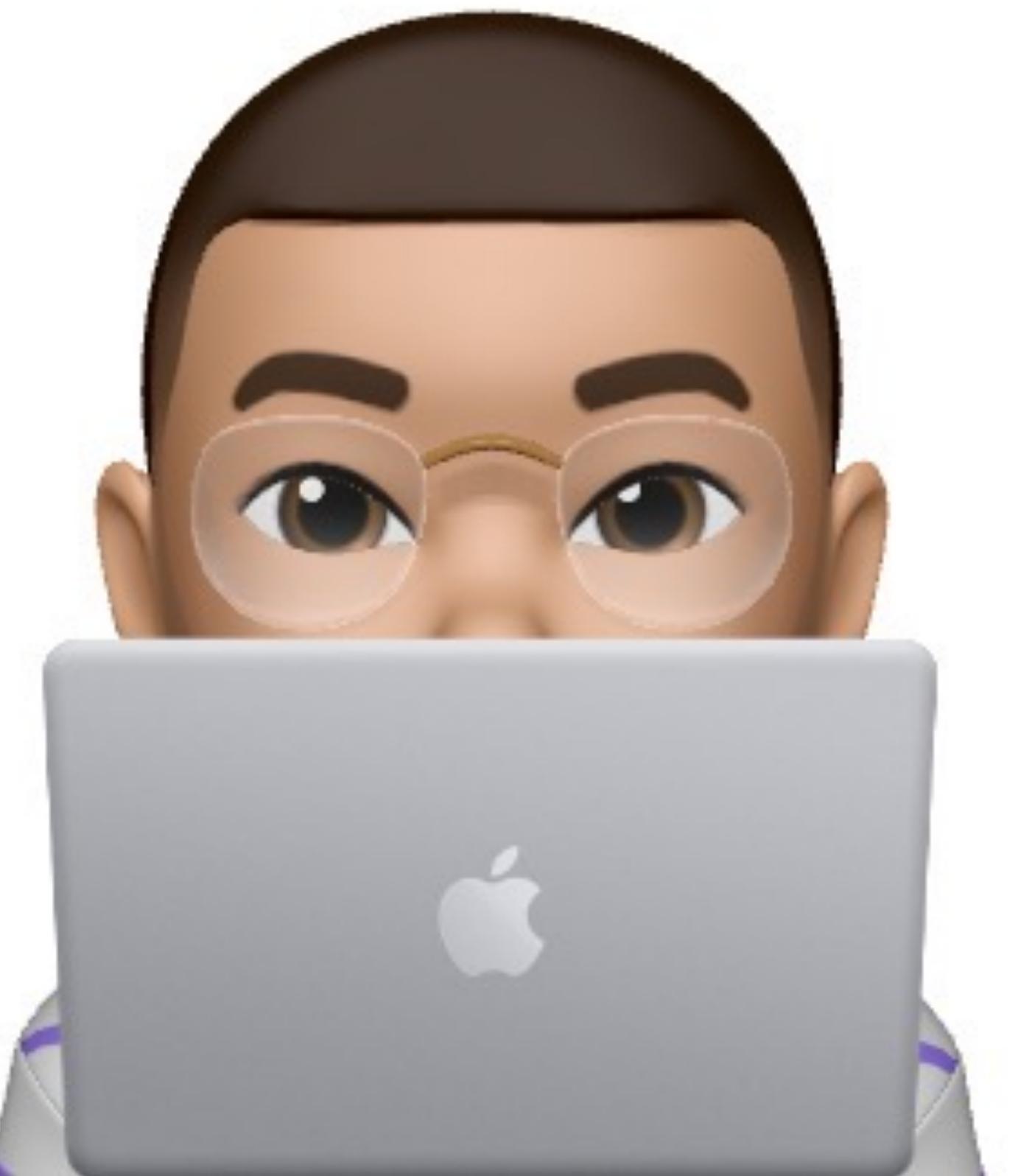


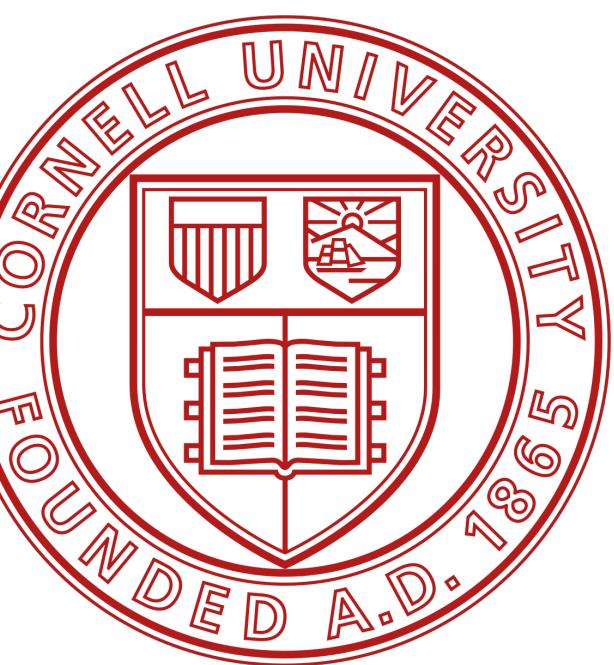


7 weeks

Overview

- W1: Introduction to R

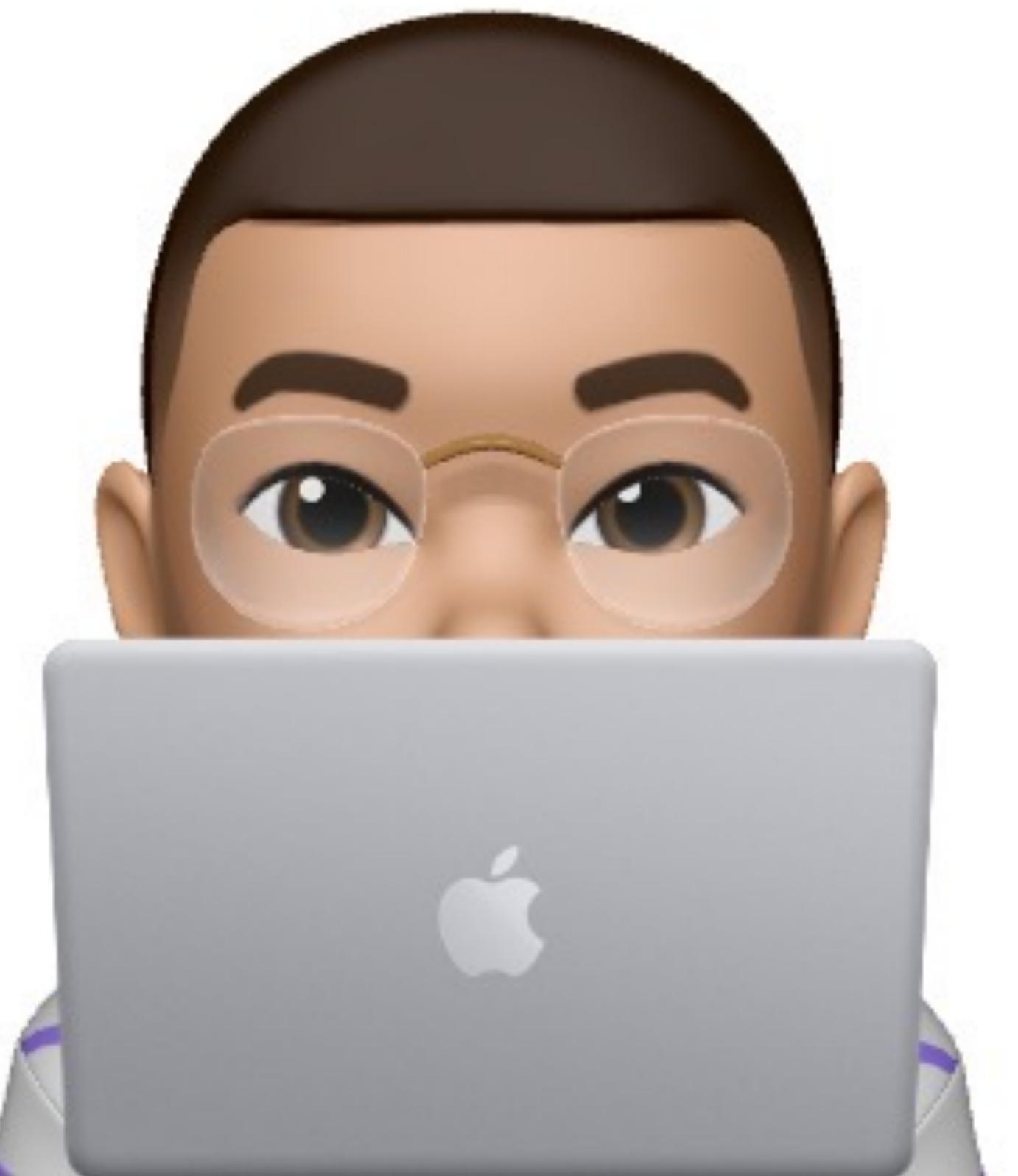


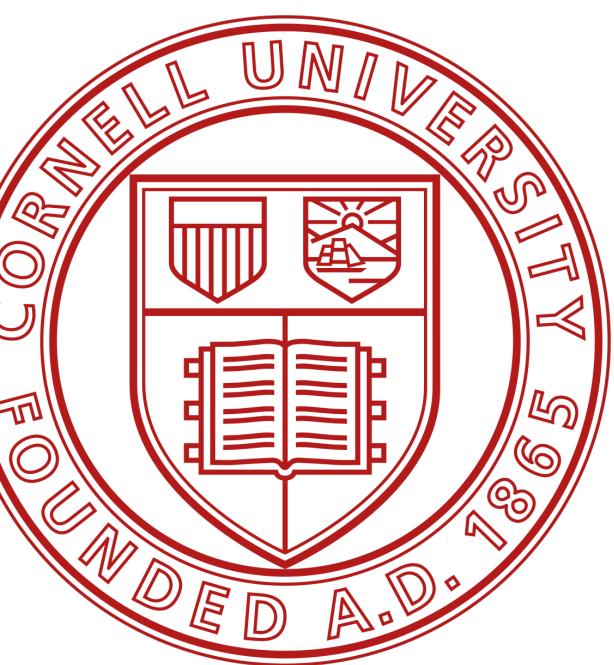


7 weeks

Overview

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

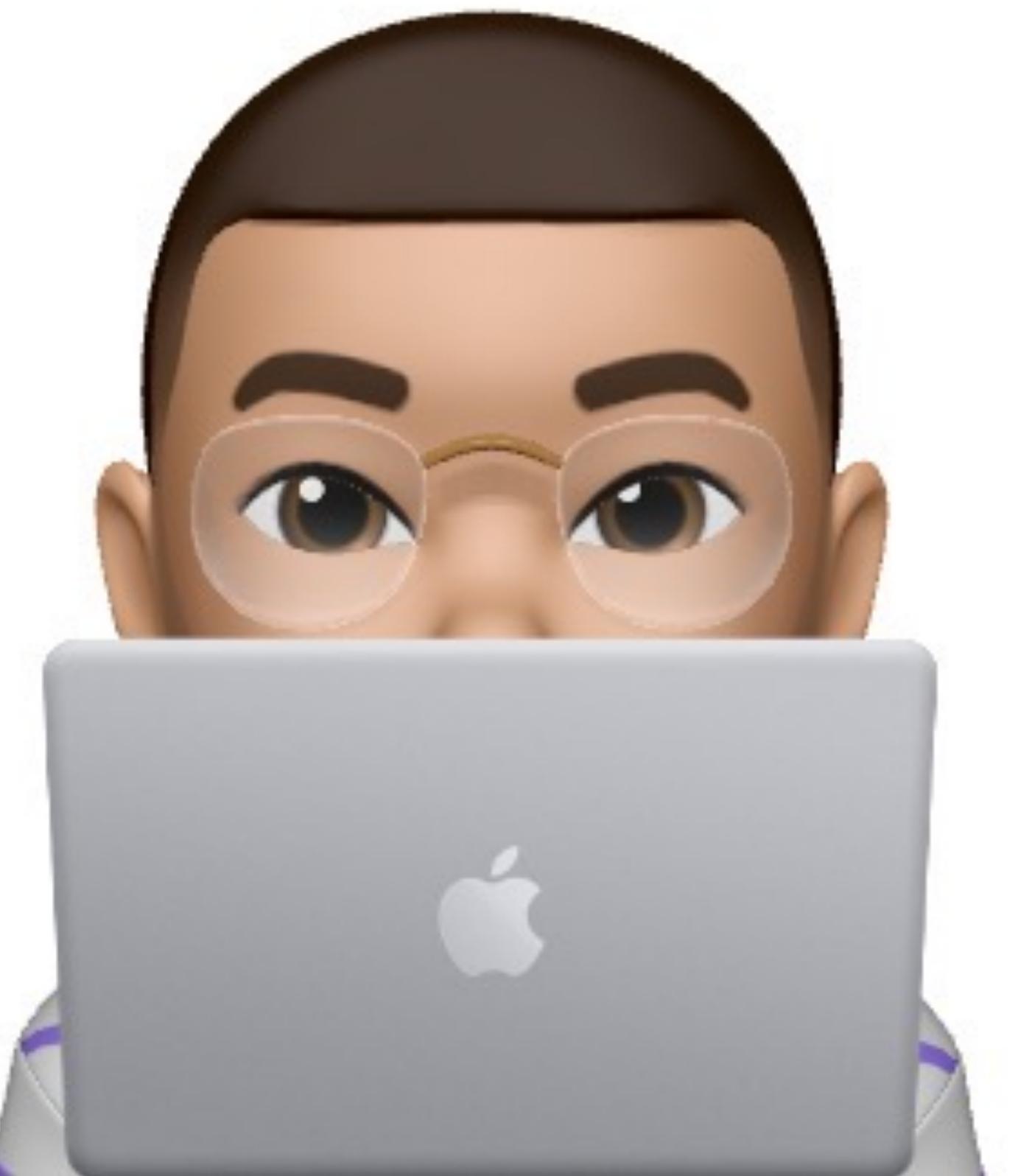


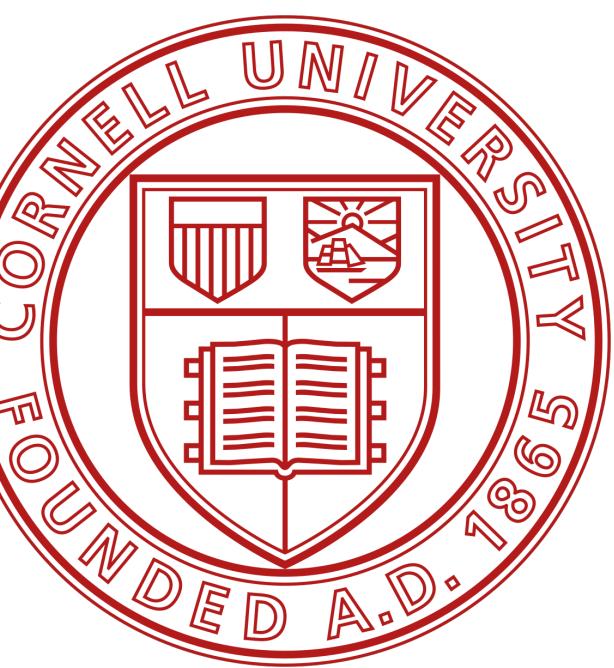


7 weeks

Overview

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

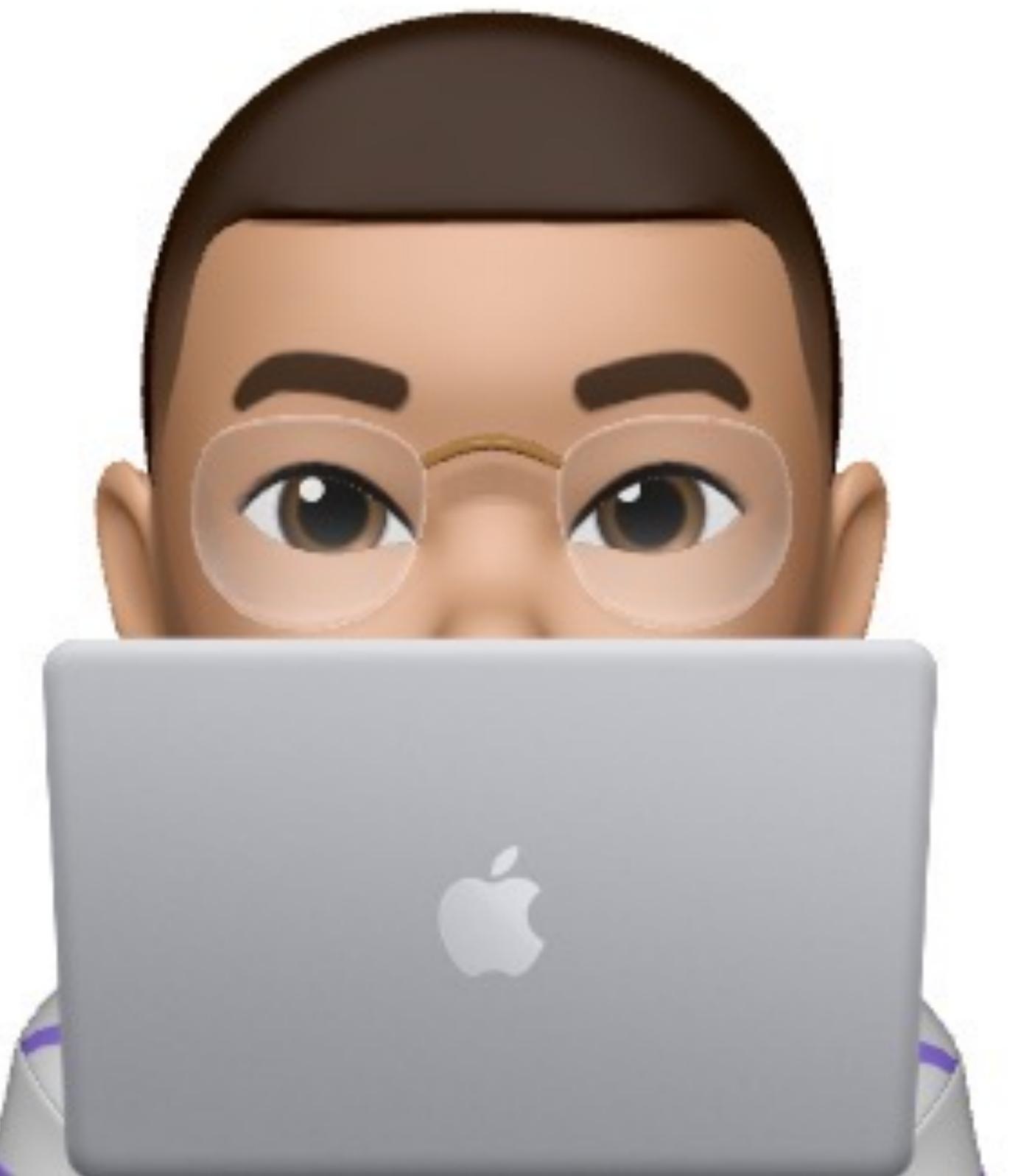


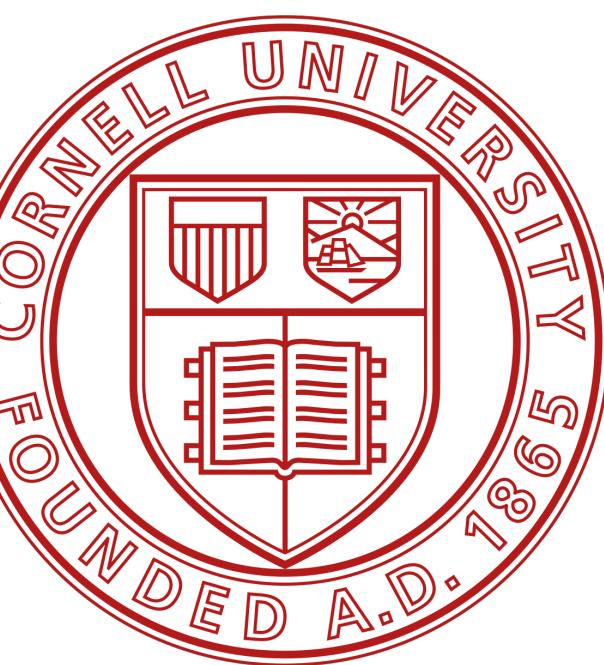


7 weeks

Overview

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

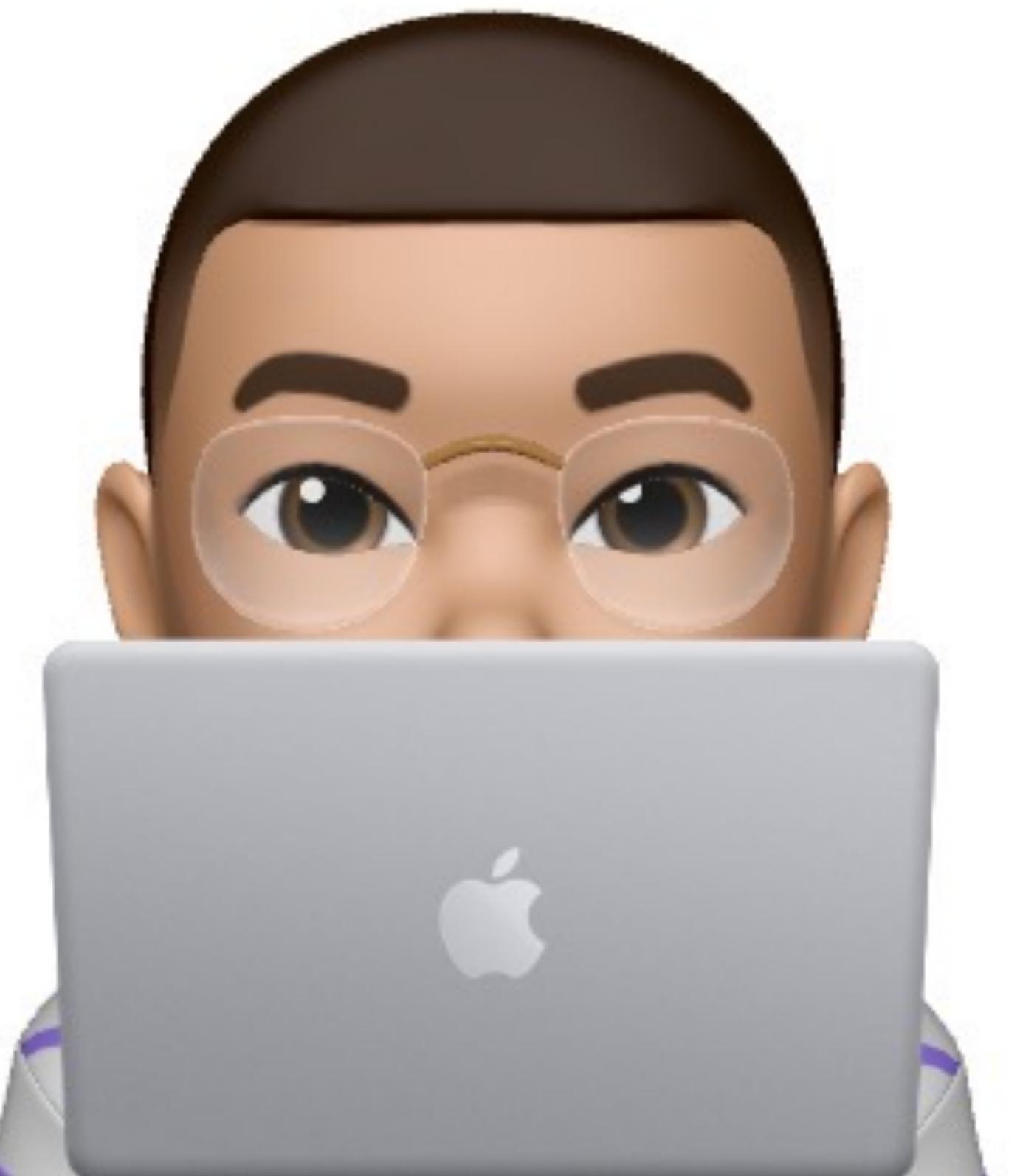


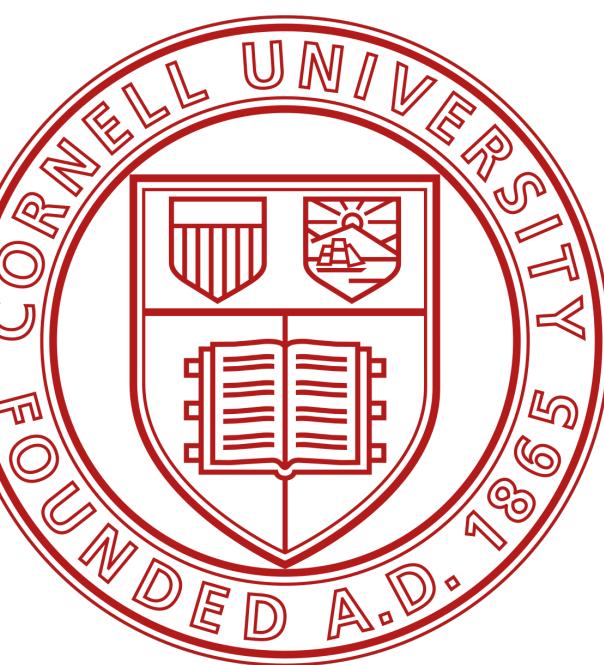


7 weeks

Overview

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

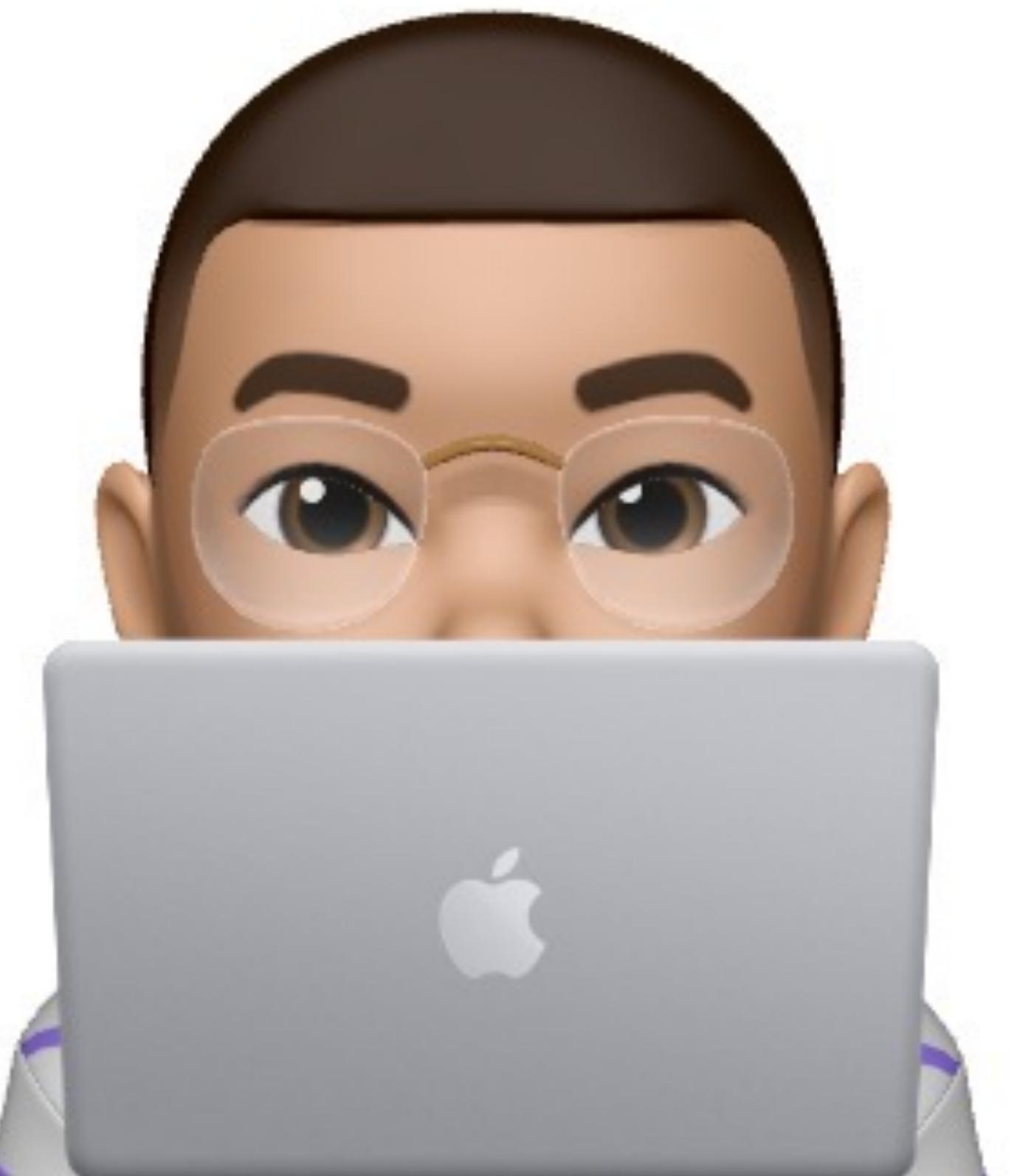


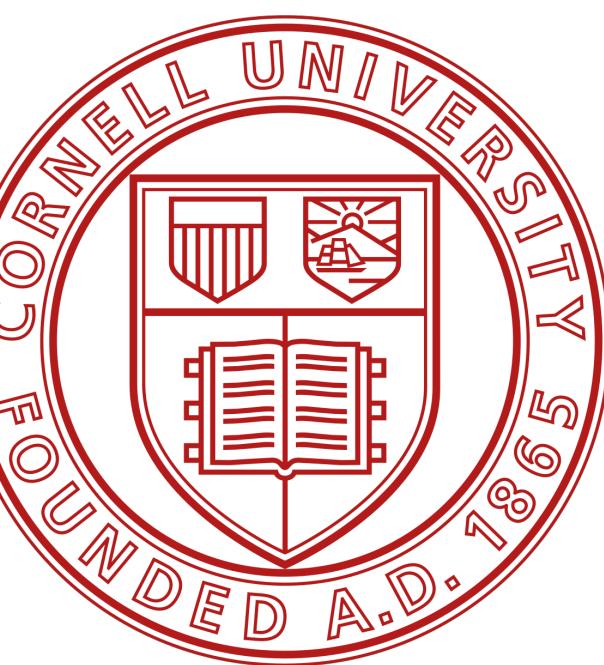


7 weeks

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

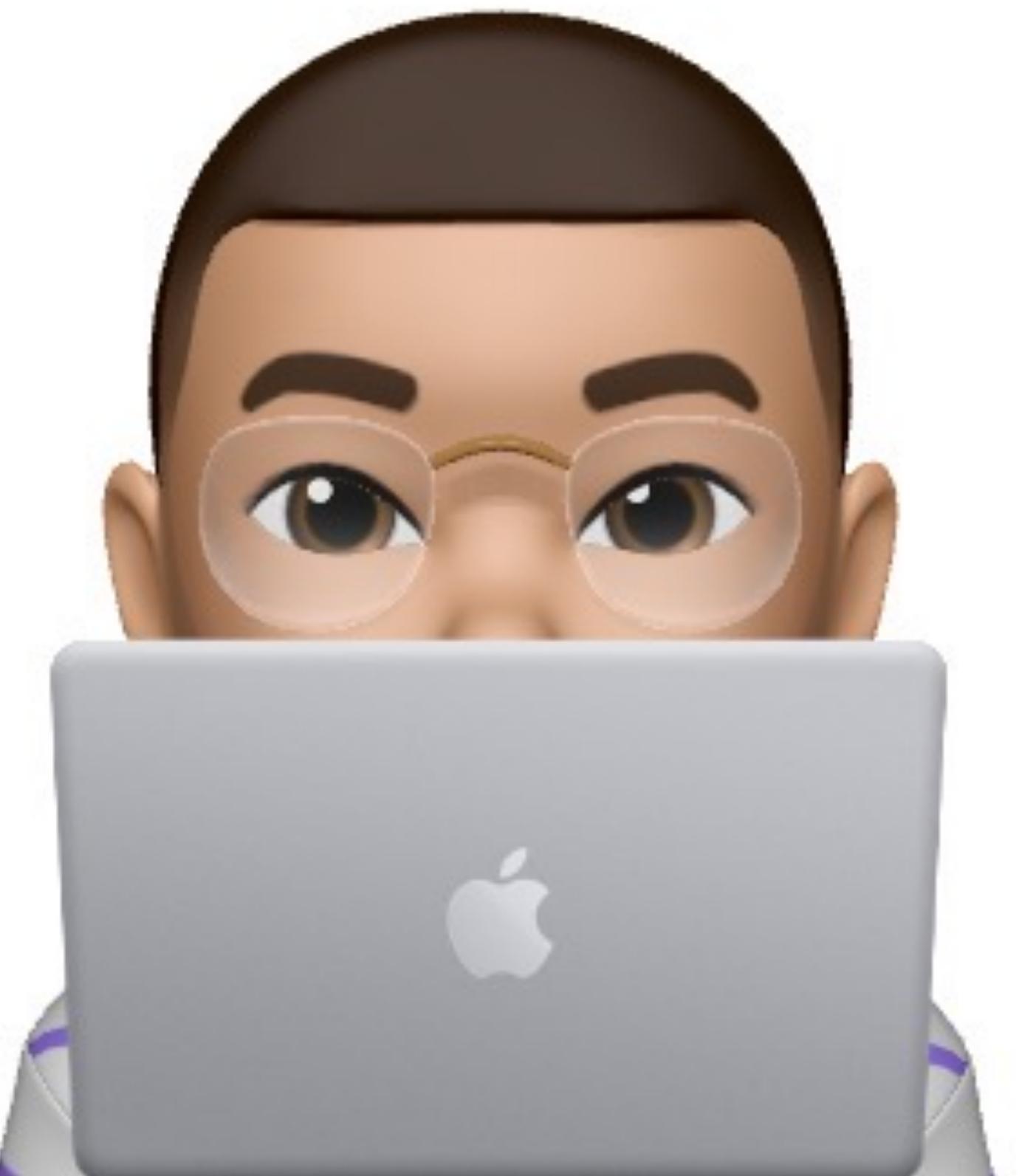


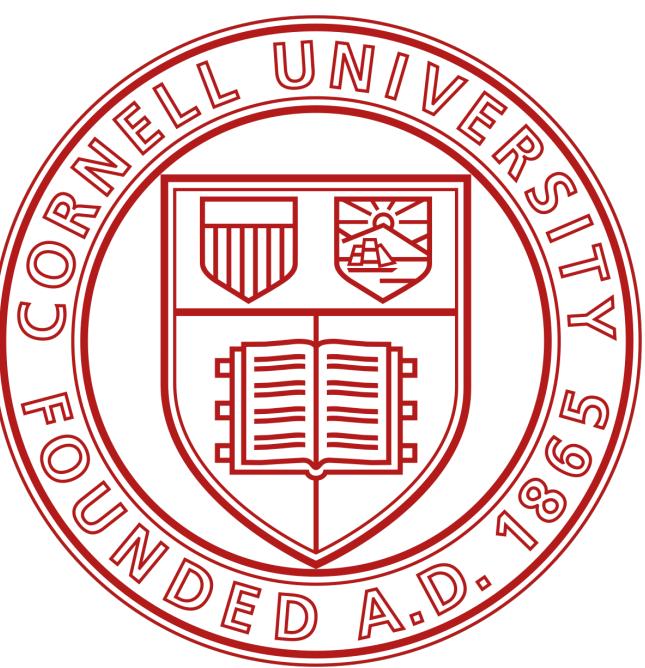


7 weeks

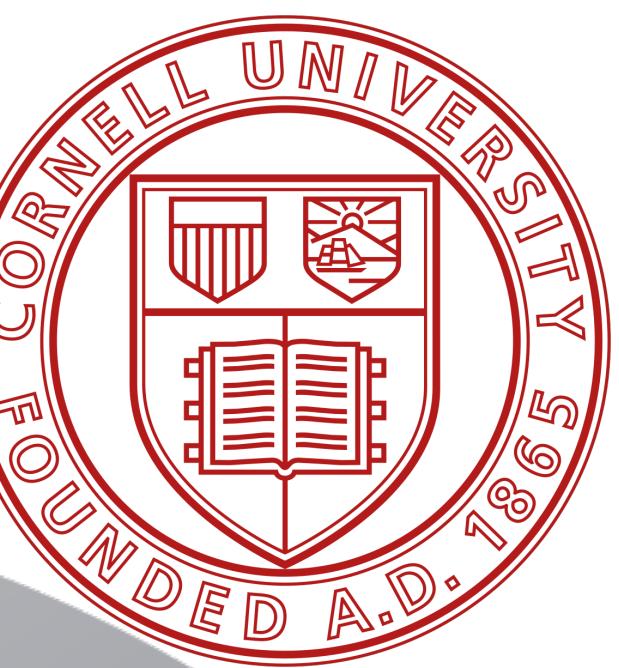
Overview

- 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





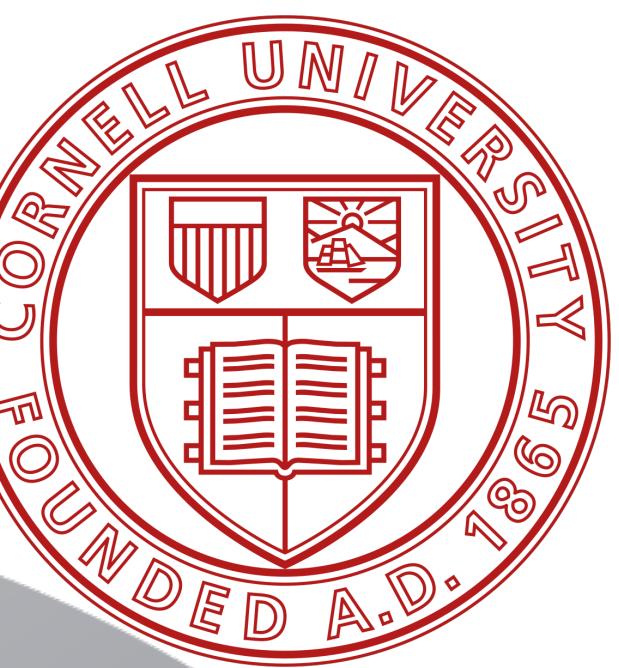
Introduction to R and RStudio



Install R

What is it ?



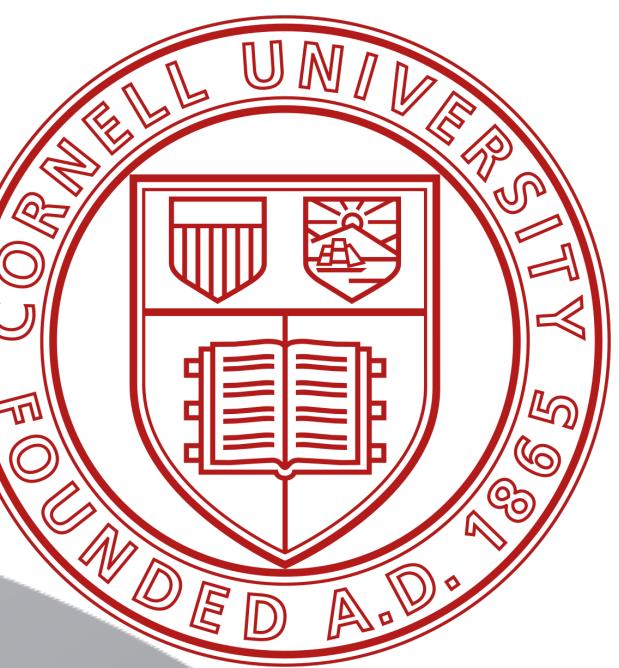


Install R

What is it ?

- R is a computer language



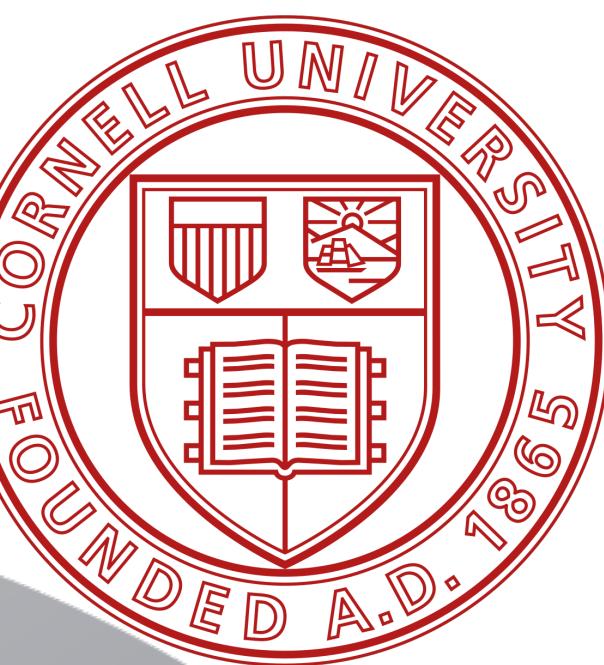


Install R

What is it ?

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



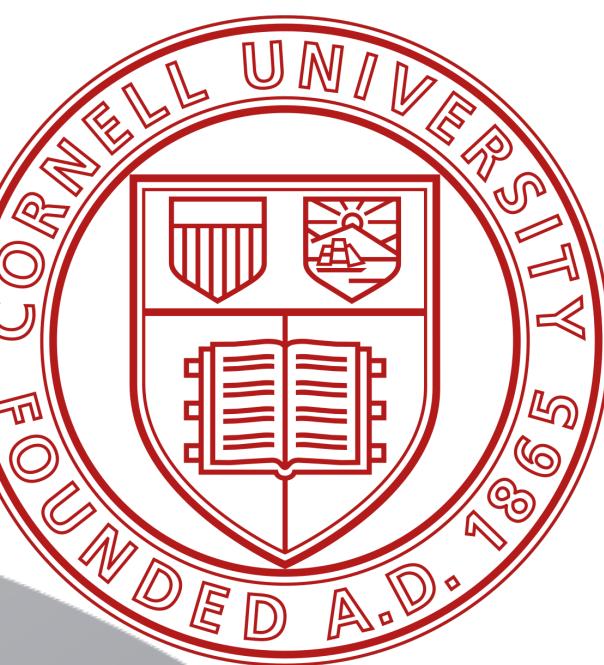


Install R

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



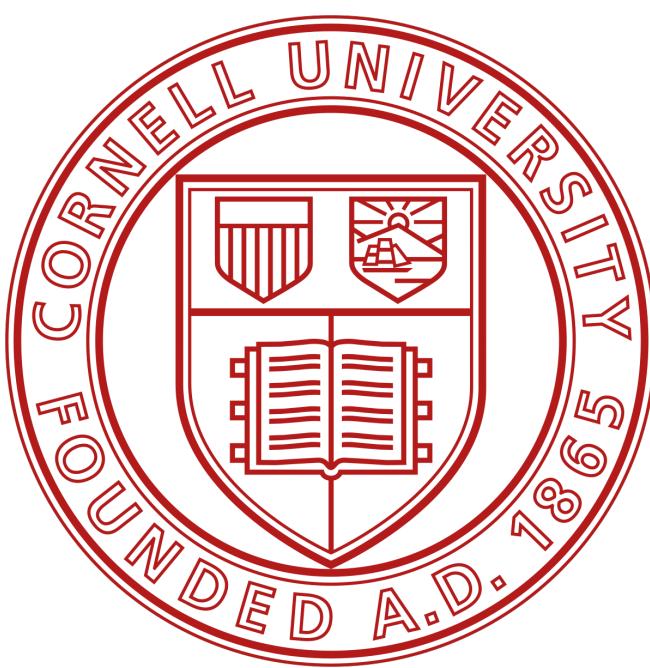


Install R

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:

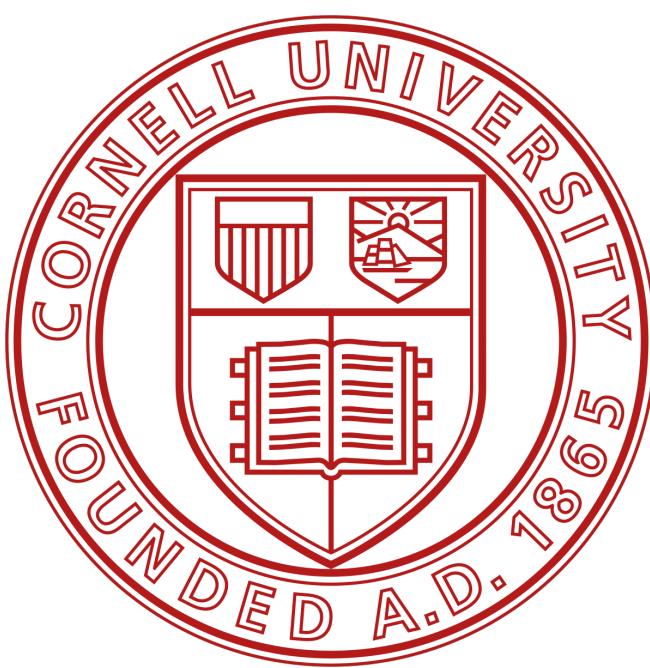
- [Download R for Linux \(Debian, Fedora/Redhat, Ubuntu\)](#)
- [Download R for macOS](#)
- [Download R for Windows](#)

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:

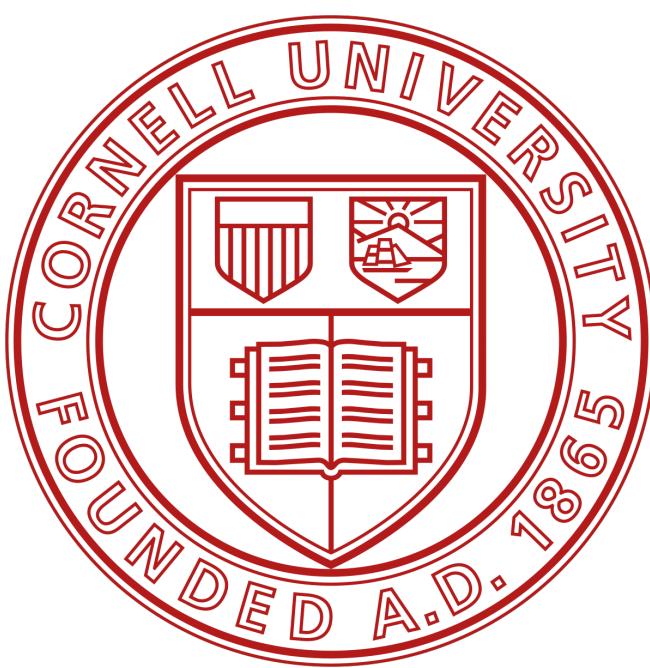
- [Download R for Linux \(Debian, Fedora/Redhat, Ubuntu\)](#)
- [Download R for macOS](#)
- [Download R for Windows](#)

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.

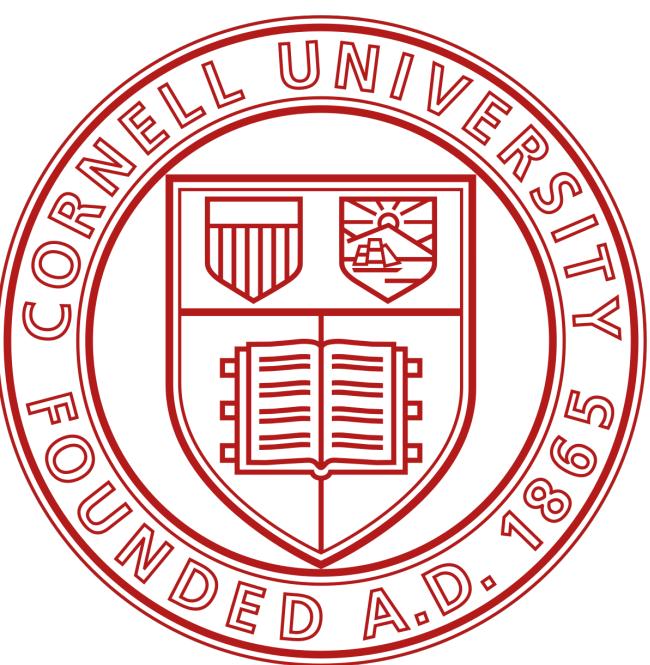
Source Code for all Platforms

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](#)

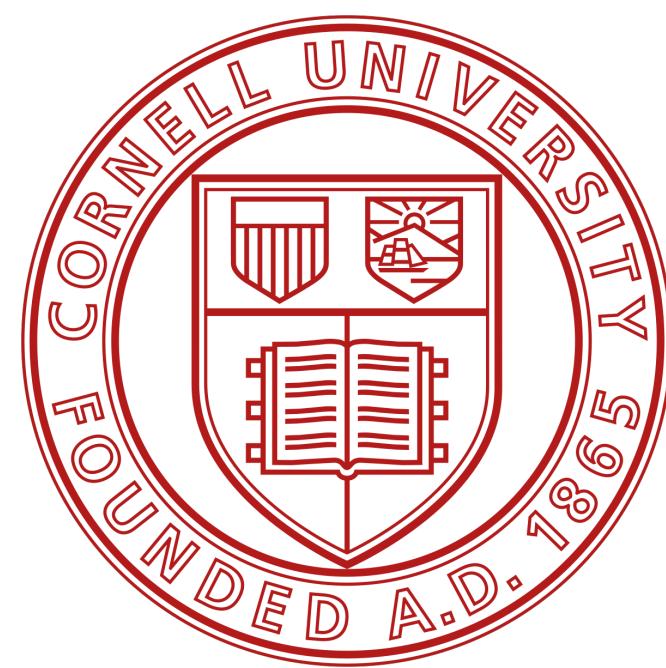
Install RStudio

What is it ?



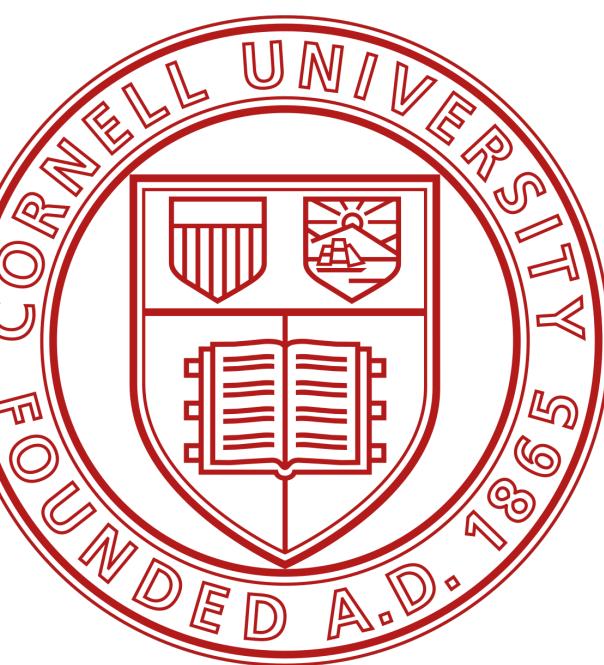
Install RStudio

What is it ?



- RStudio is an application like Microsoft Word.



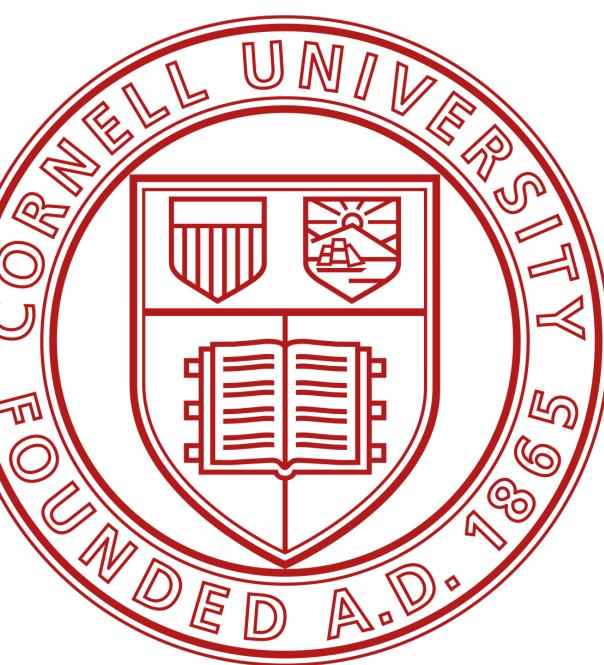


Install RStudio

What is it ?

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



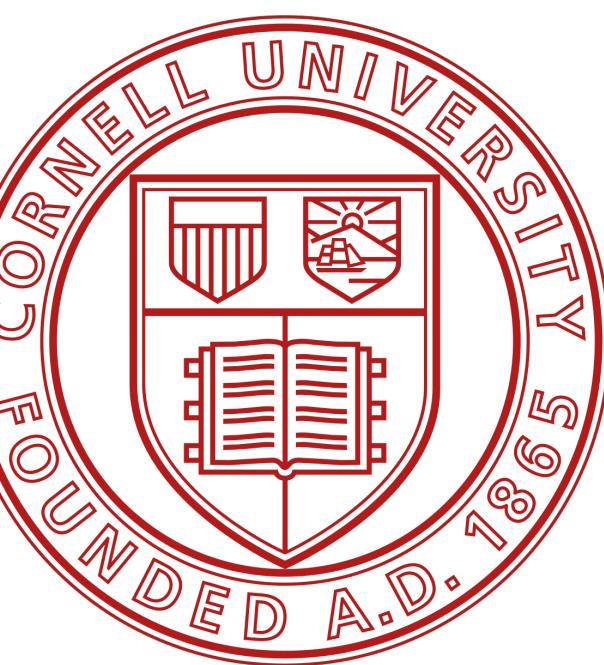


Install RStudio

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

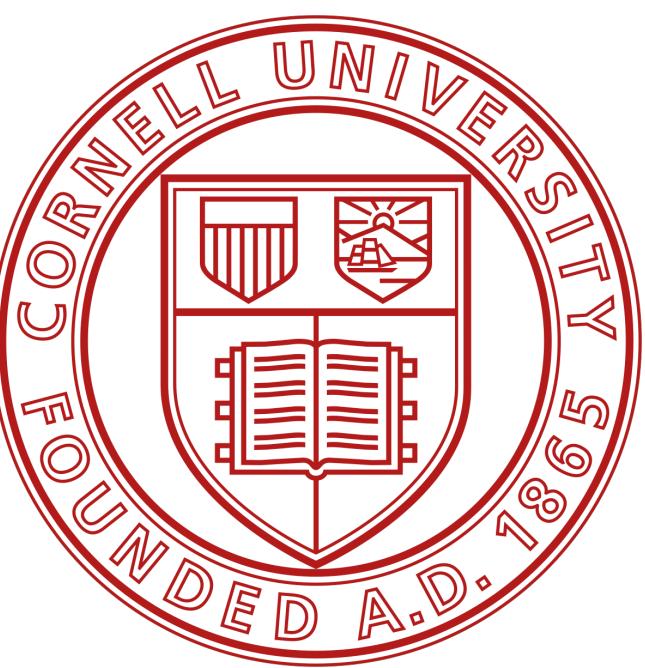
What is it ?

- 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 ?

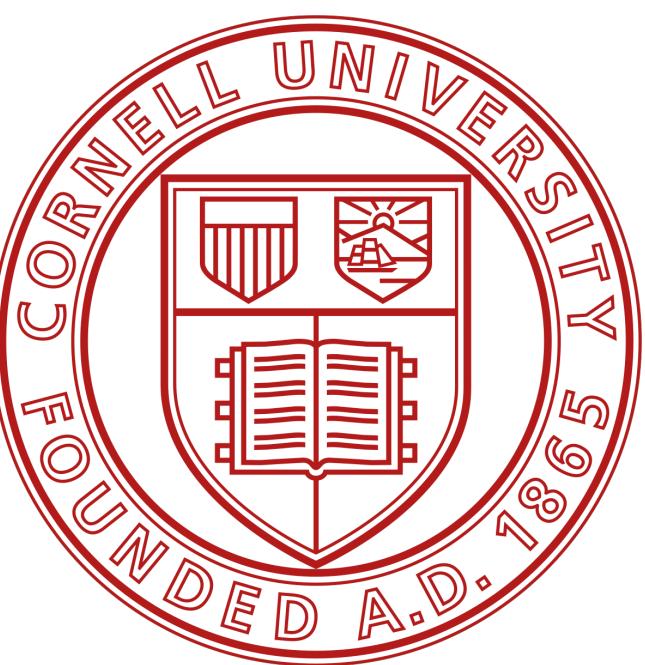


The screenshot displays the RStudio interface with the following components:

- Code Editor:** Shows an R script named "flights-example.R" with code for loading the "nycflights13" package, creating a daily dataset, and generating a boxplot.
- Console:** Shows the execution of the R script, displaying the structure of the "daily" dataset and its first few rows.
- Environment:** Shows the "Global Environment" pane with the "daily" dataset listed.
- Plots:** Shows a boxplot titled "Number of 2013 New York Flights Each Weekday" with the y-axis labeled "Flights" and the x-axis labeled "Weekday" (Sun, Mon, Tue, Wed, Thu, Fri, Sat).

Install RStudio

How to install it ?



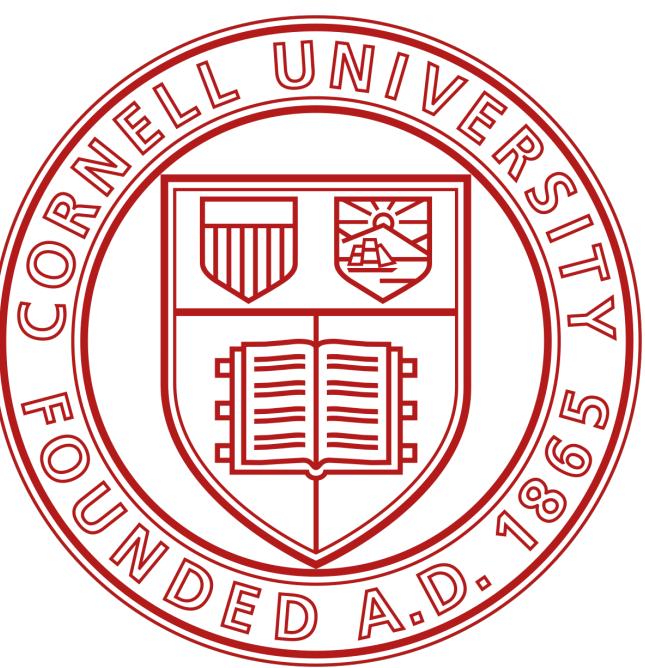
- Download RStudio for free.

The screenshot shows the RStudio interface with the following components:

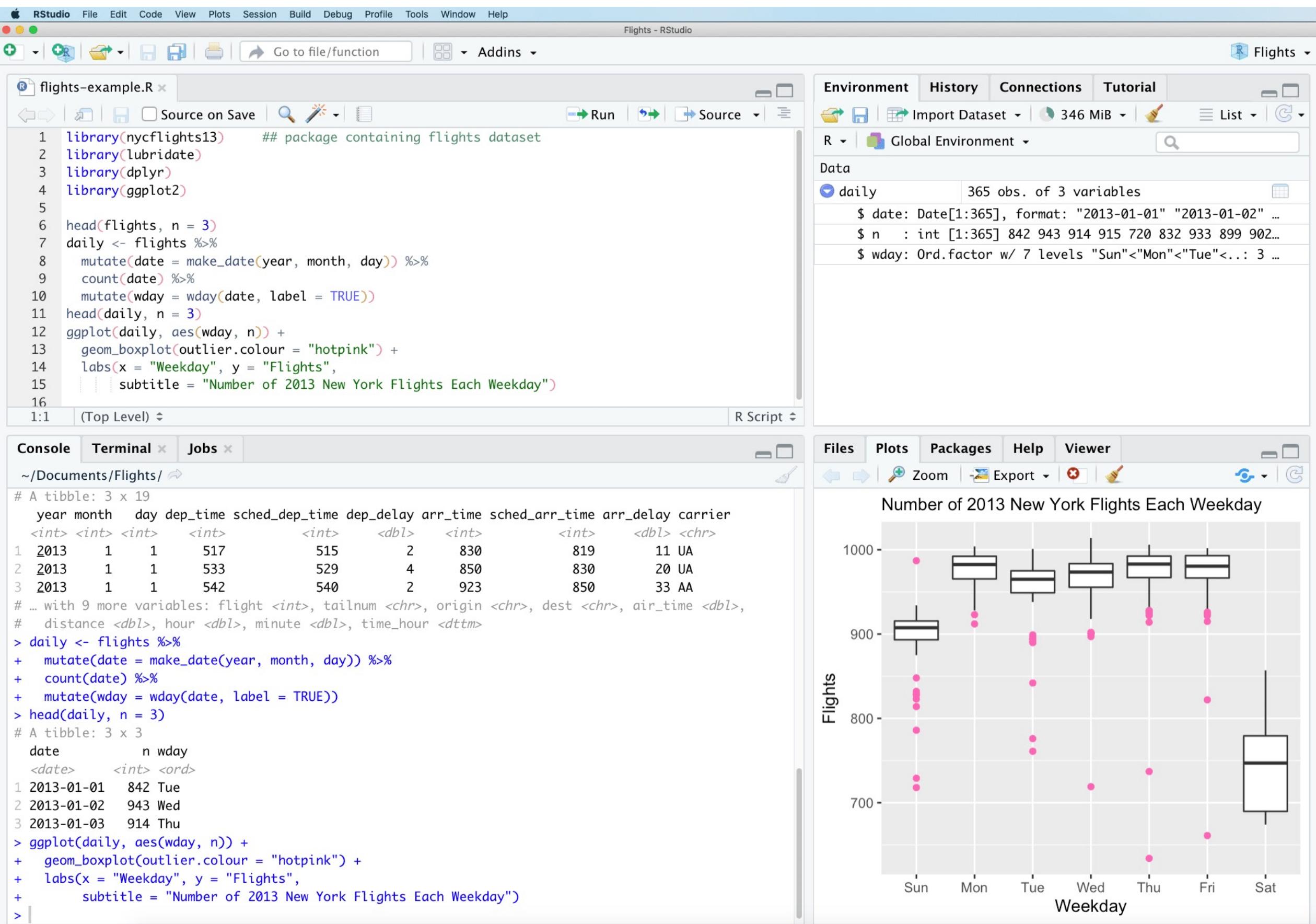
- Code Editor:** The "flights-example.R" script window contains R code for data manipulation and visualization. The code includes loading packages (nycflights13, lubridate, dplyr, ggplot2), creating a daily dataset from flights, and generating a boxplot for flight counts by weekday.
- Console:** The "Console" tab shows the R session history, including the execution of the R code and its output. It displays the structure of the "daily" tibble and the resulting boxplot data.
- Environment:** The "Environment" tab shows the global variables defined in the session, specifically the "daily" dataset.
- Plots:** The "Plots" tab displays a boxplot titled "Number of 2013 New York Flights Each Weekday". The x-axis is labeled "Weekday" and lists Sunday through Saturday. The y-axis is labeled "Flights" and ranges from 700 to 1000. The plot shows that flight volumes are highest on Monday and Friday, and lowest on Saturday.

Install RStudio

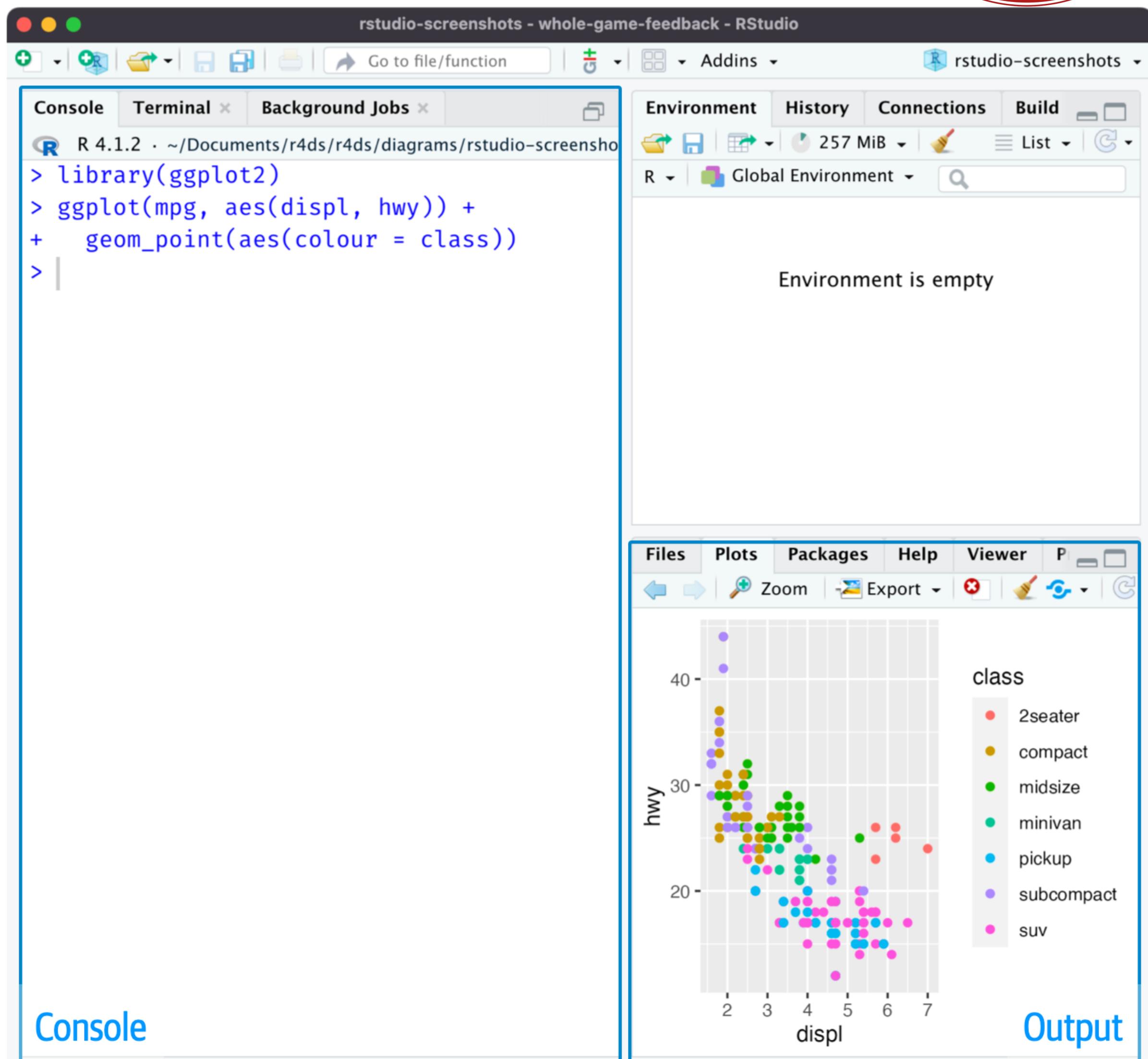
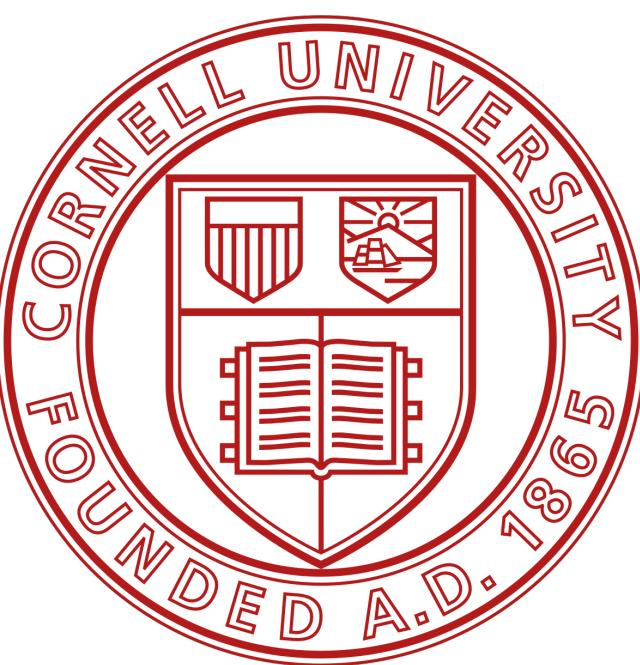
How to install it ?



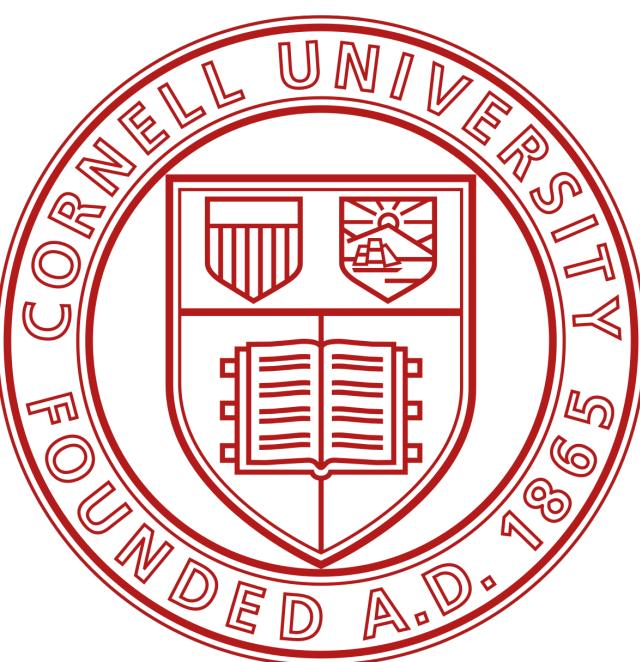
- [Download RStudio for free.](#)
- Use Studio on [Posit Cloud](#).



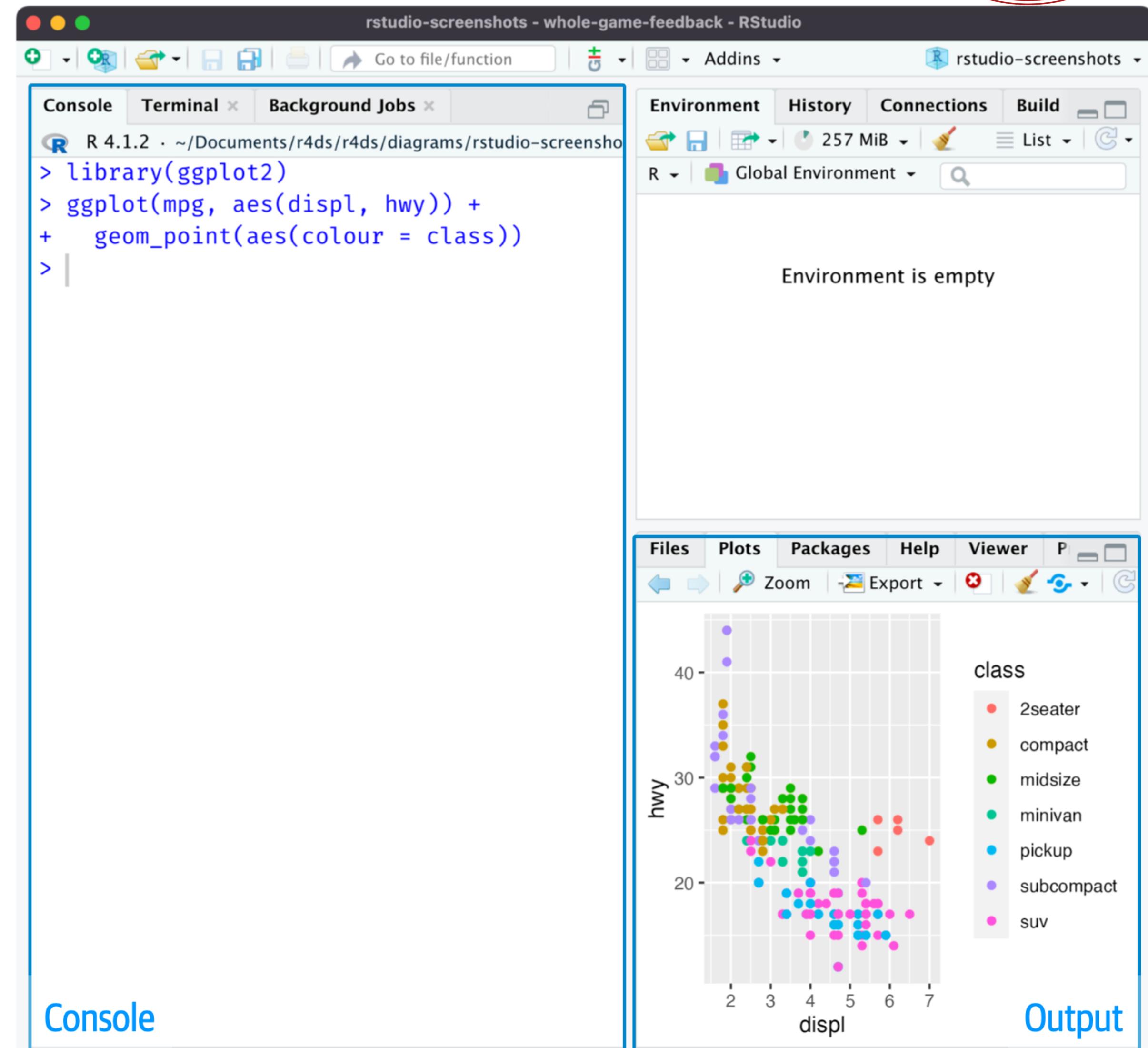
Install RStudio Interface



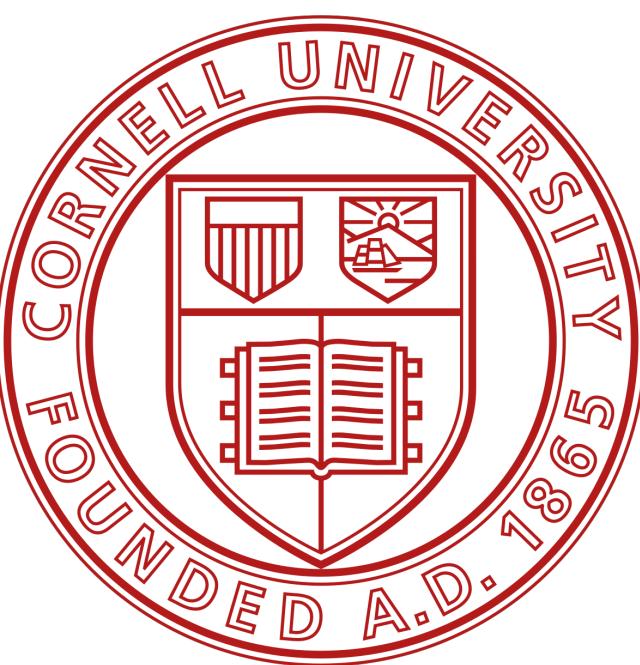
Install RStudio Interface



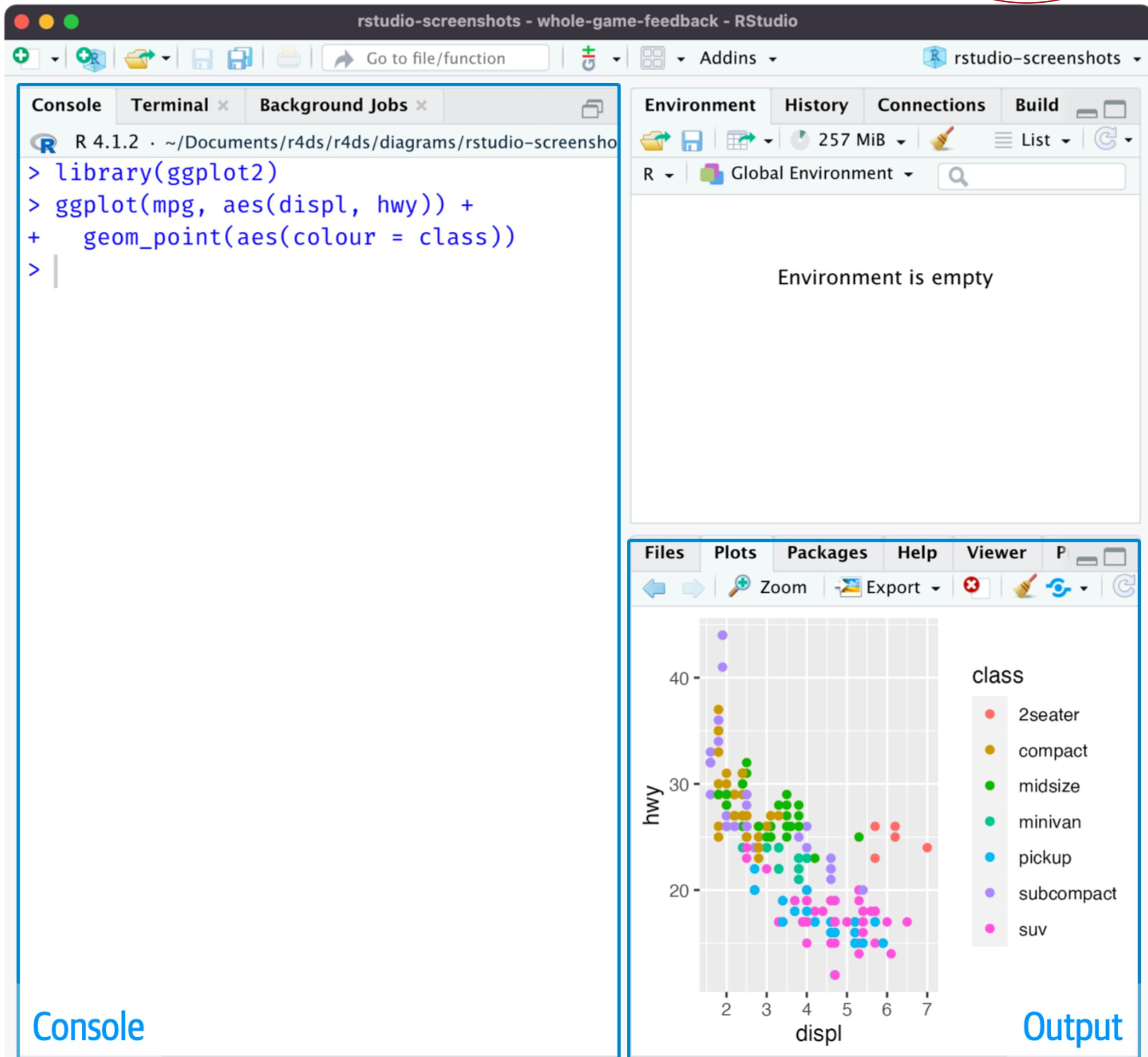
- Two key regions in the interface: the console pane and the output pane.



Install RStudio Interface

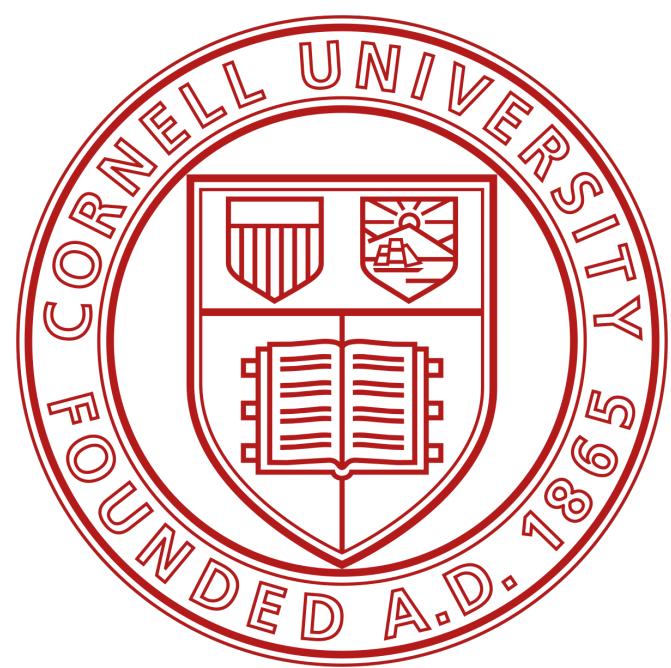


- Two key regions in the interface: the console pane and the output pane.
- You type the R code in the console pane and press enter to run it.

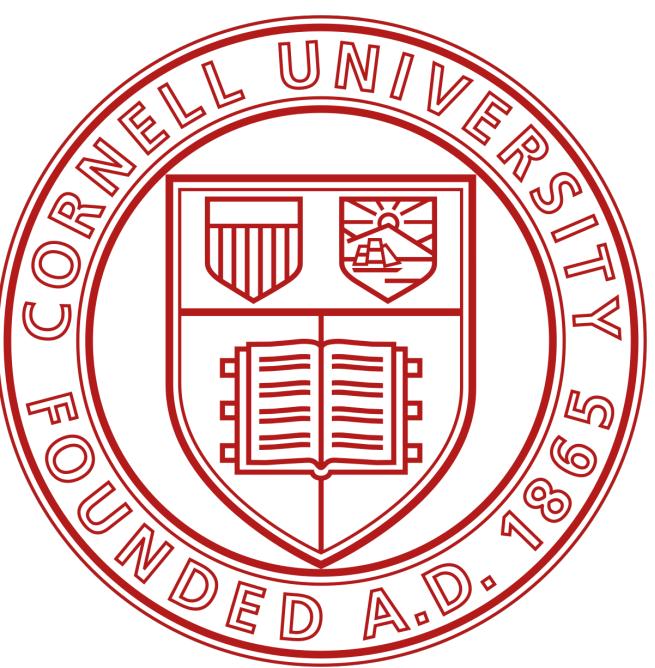


R packages

What is it ?



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

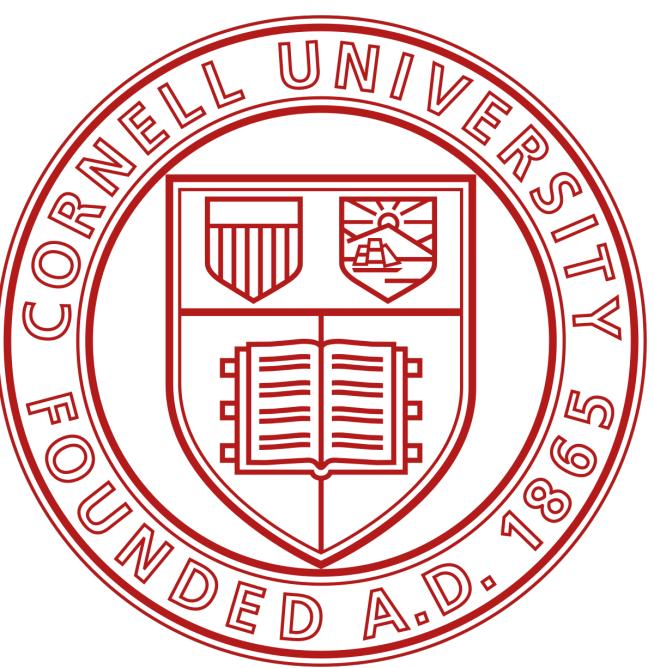


R packages

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>")
```

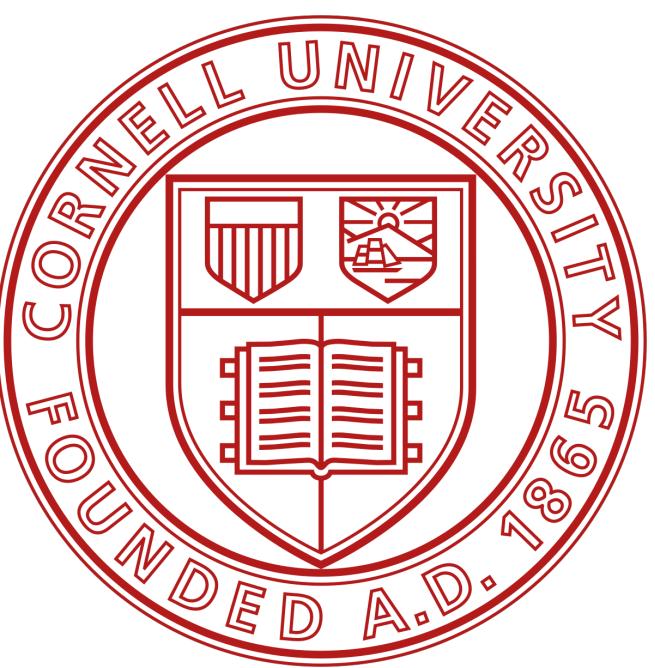


R packages

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

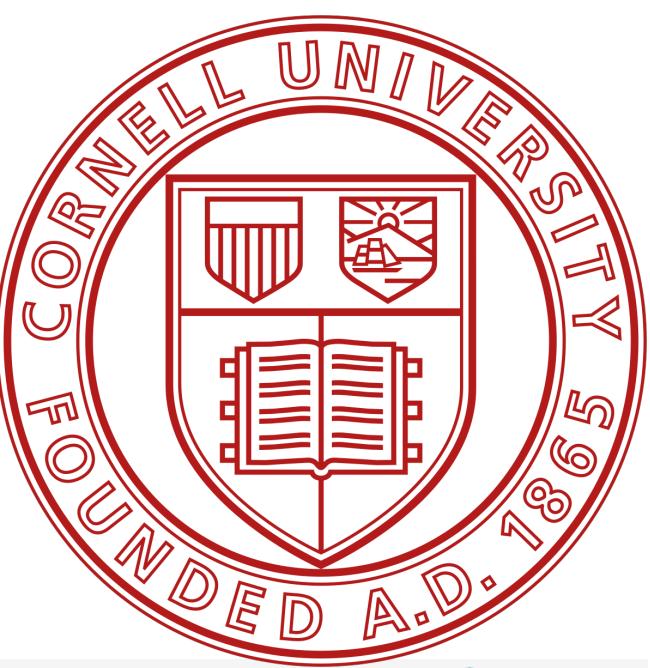
What is it ?

- 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

A screenshot of the RStudio interface. The top menu bar includes "Project: (None)", "File", "Edit", "View", "Tools", "Help", and "Presentation". The main workspace shows a script editor with the code "1 install.packages("tidyverse")" and a console window displaying the R startup message and license information.

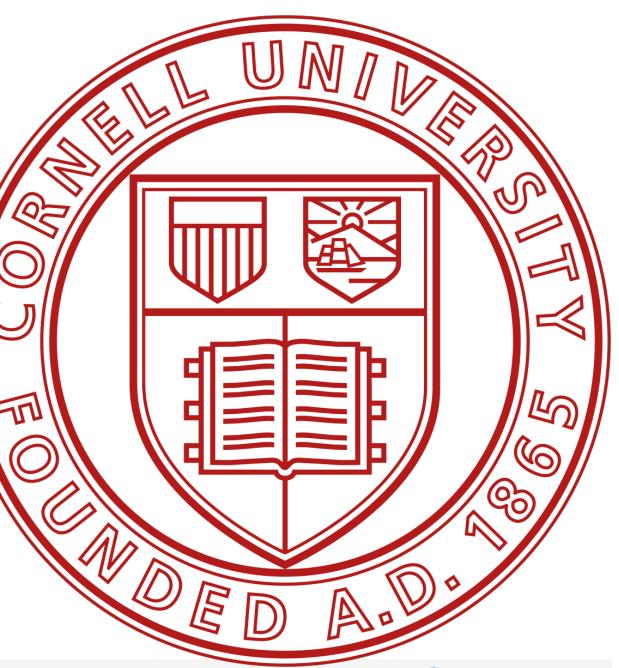
```
1:30 (Top Level) R Script
Console Terminal Background Jobs
R 4.4.1 - ~/~

R version 4.4.1 (2024-06-14) -- "Race for Your Life"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin20

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or S in publications.
```



R packages

Installing packages

- Open a new R script.

The screenshot shows the RStudio interface. In the top-left pane, there is an untitled R script window containing the command `install.packages("tidyverse")`. In the bottom-left pane, the R console displays the R startup message and license information. The right-hand sidebar shows the Global Environment tab, which is currently empty.

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

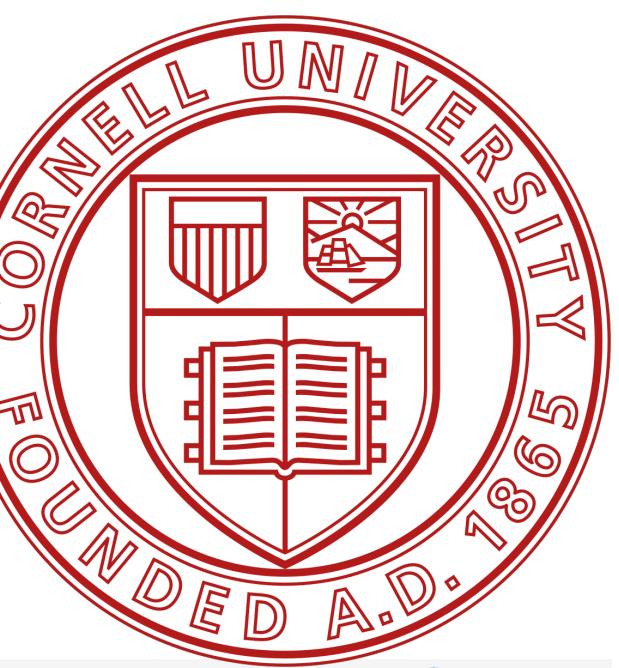
```
1:30 (Top Level) R Script
Console Terminal Background Jobs
R 4.4.1 - ~/~

R version 4.4.1 (2024-06-14) -- "Race for Your Life"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin20

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or S in publications.
```



R packages

Installing packages

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

A screenshot of the RStudio interface. The left pane shows an R script editor with a single line of code: `install.packages("tidyverse")`. The right pane shows the RStudio environment, which is currently empty. Below the script editor is the R console window, which displays the standard R startup message and license information.

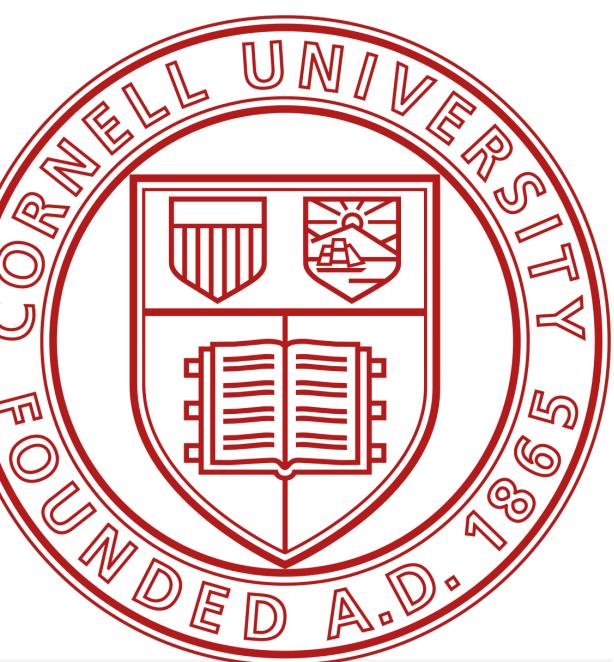
```
1:30 (Top Level) R Script
Console Terminal Background Jobs
R 4.4.1 - ~/~

R version 4.4.1 (2024-06-14) -- "Race for Your Life"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin20

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

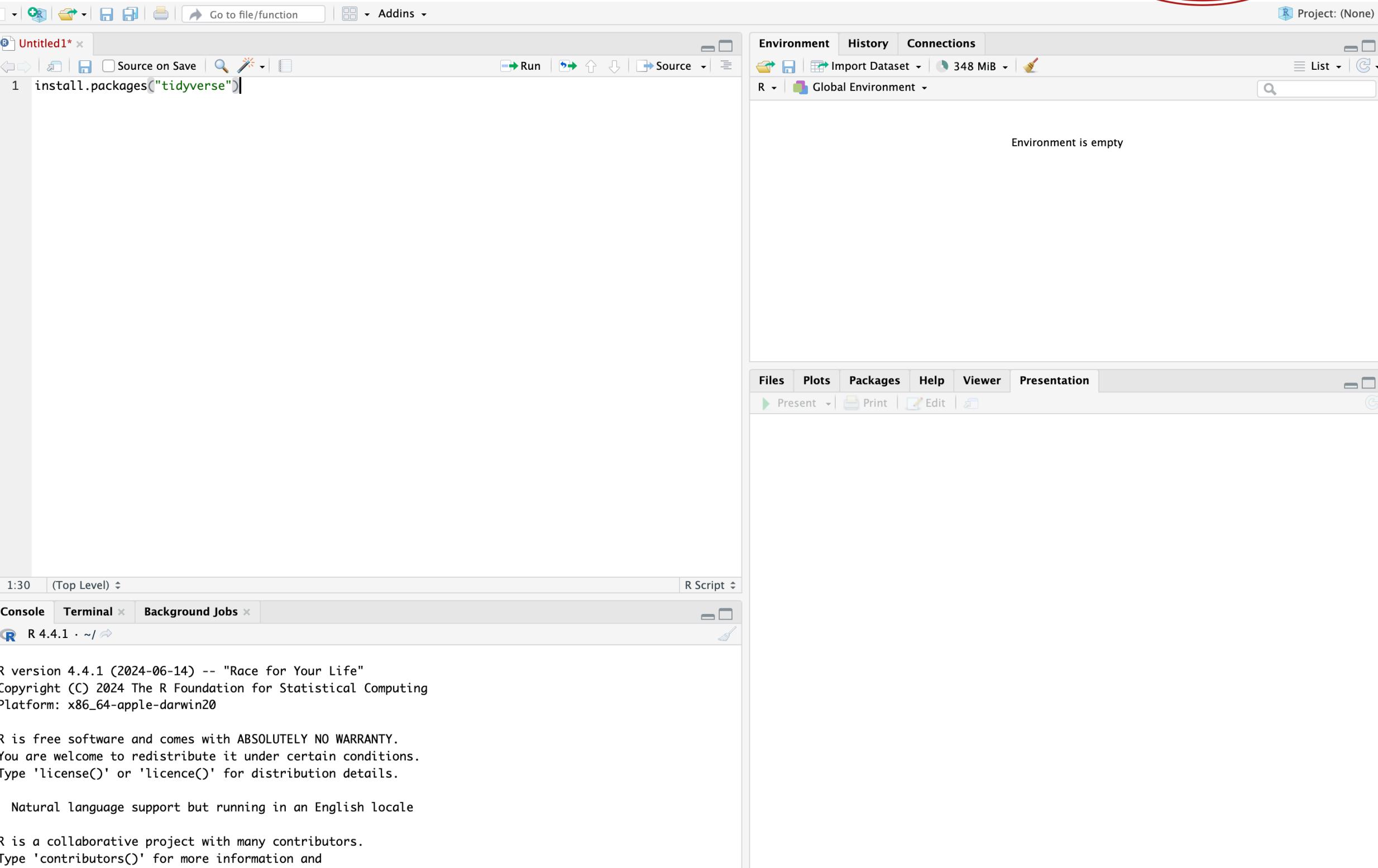
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or S in publications.
```



R packages

Installing packages

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



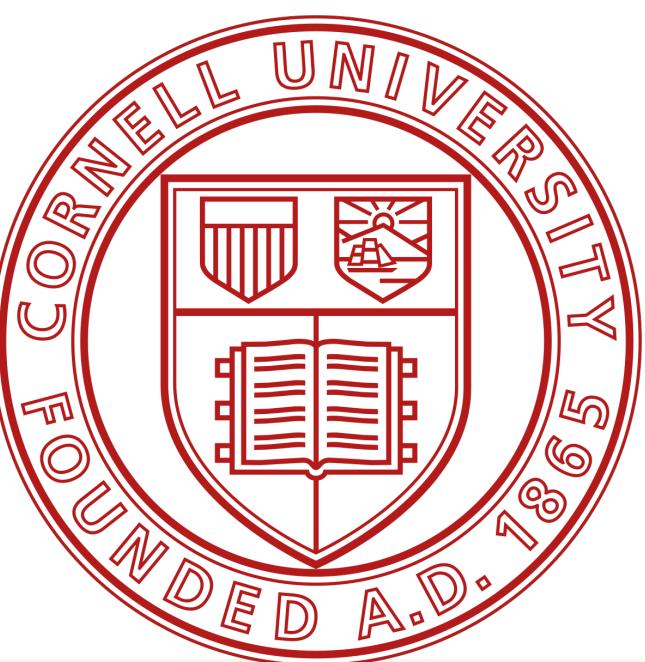
The screenshot shows the RStudio interface. In the top-left pane, there is an untitled R script with the line of code:

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

In the bottom-left pane, the R console displays the R startup message and license information:

```
R version 4.4.1 (2024-06-14) -- "Race for Your Life"  
Copyright (C) 2024 The R Foundation for Statistical Computing  
Platform: x86_64-apple-darwin20  
  
R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.  
  
Natural language support but running in an English locale  
  
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or S in publications.
```

The right-hand sidebar shows the Global Environment, which is currently empty.



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.

A screenshot of the RStudio interface. The left pane shows an R script editor with the code `install.packages("tidyverse")`. The right pane shows the R console output. The console window displays the R startup message: "R version 4.4.1 (2024-06-14) -- 'Race for Your Life' Copyright (C) 2024 The R Foundation for Statistical Computing Platform: x86_64-apple-darwin20". It also shows the standard R license and copyright information, followed by "Natural language support but running in an English locale" and "R is a collaborative project with many contributors. Type 'contributors()' for more information and 'citation()' on how to cite R or S in publications".

```
1:30 (Top Level) R Script
Console Terminal Background Jobs
R 4.4.1 - ~/ ~

R version 4.4.1 (2024-06-14) -- "Race for Your Life"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin20

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

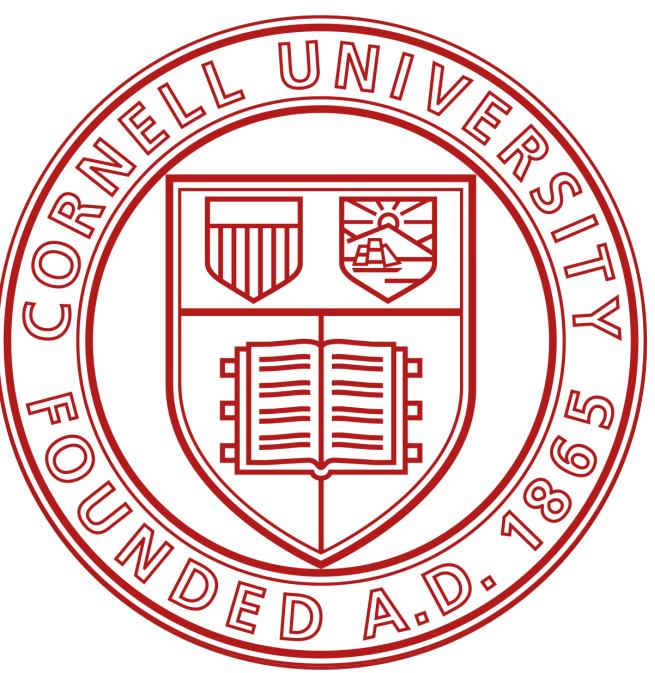
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'terms()' for the legal terms of distribution.

Environment History Connections
Import Dataset 348 MiB
R Global Environment
Environment is empty

Files Plots Packages Help Viewer Presentation
Present Print Edit
```

The screenshot shows the RStudio interface with several panes open:

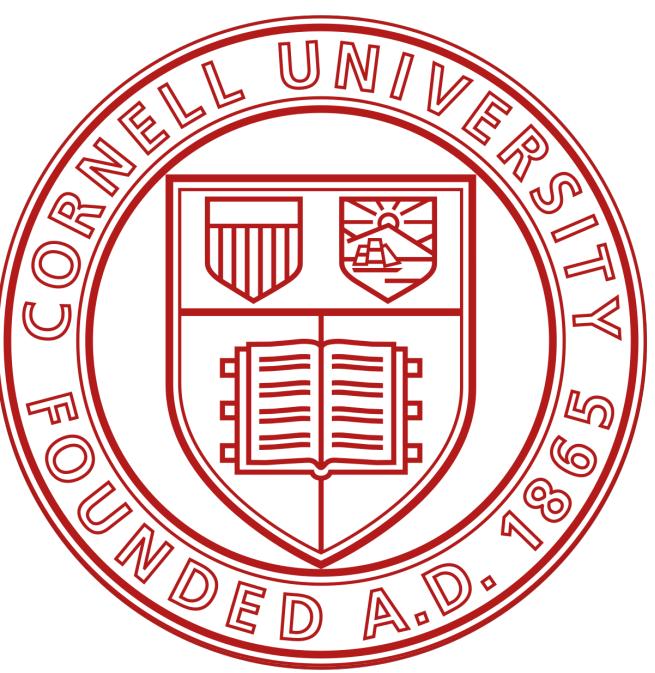
- Environment** pane: Shows the "Global Environment" tab with a message "Environment is empty".
- Global Environment** pane: Shows the "Global Environment" tab.
- Console** pane: Shows the command `install.packages("tidyverse")` being run. The output indicates that two packages are being downloaded from CRAN, totaling 515 KB and 418 KB respectively. The packages are saved to the directory `/var/folders/2r/9nmfdty958xcz1d179s1fxgw0000gn/T//RtmpKNCKL4/downloaded_packages`.
- Source** pane: Shows the R script `Untitled1.R` with the code `install.packages("tidyverse")`.



R packages

Loading packages

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

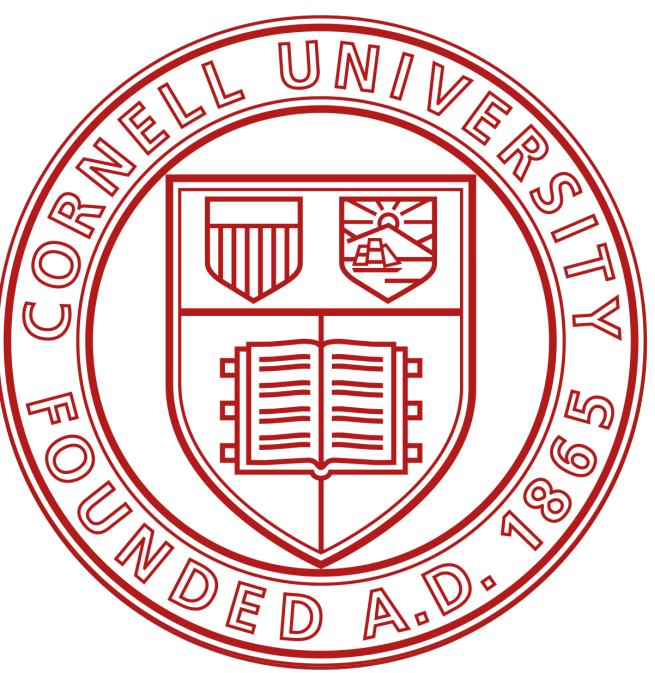


R packages

Loading packages

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

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

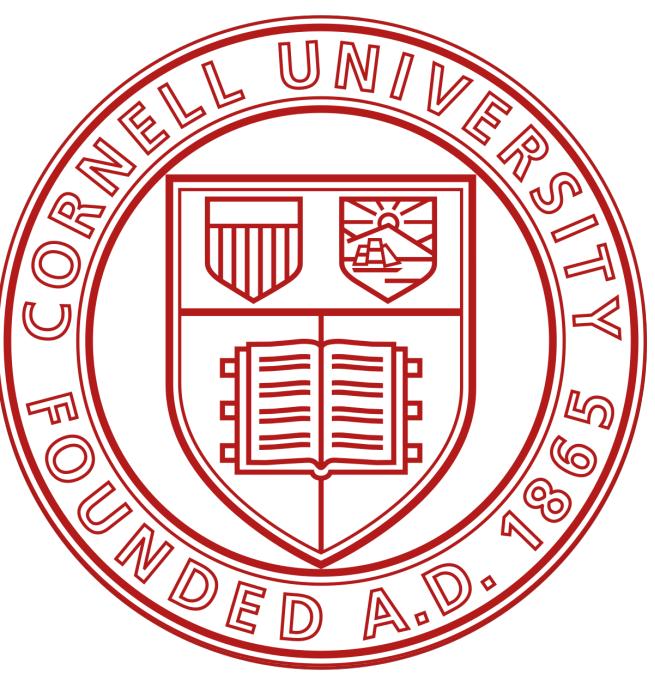


R packages

Loading packages

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

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

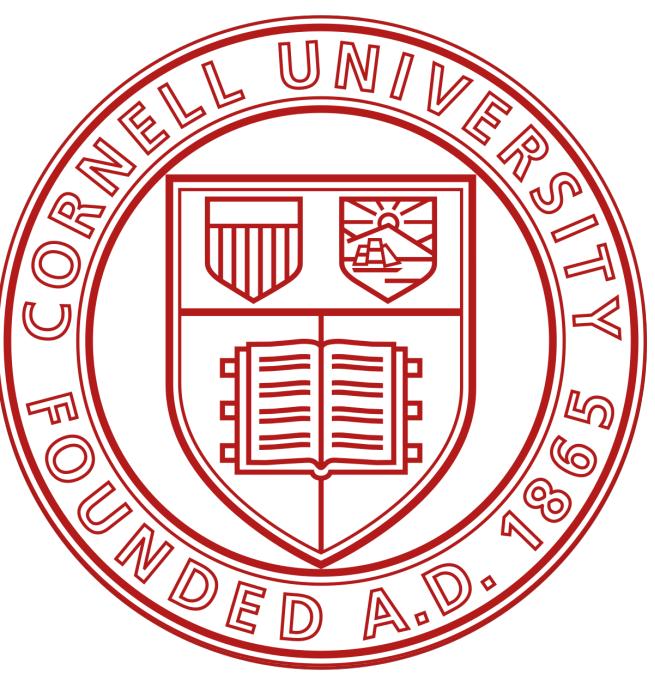


R packages

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>)
```

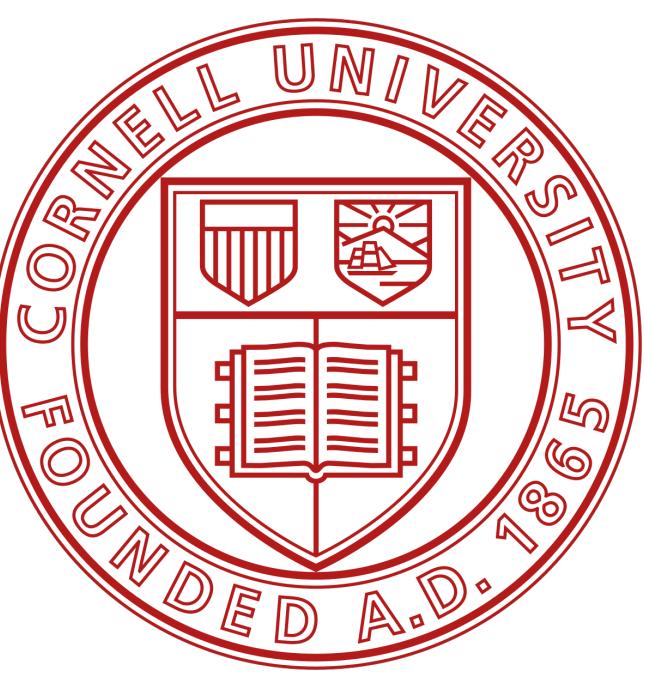


R packages

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>)
```

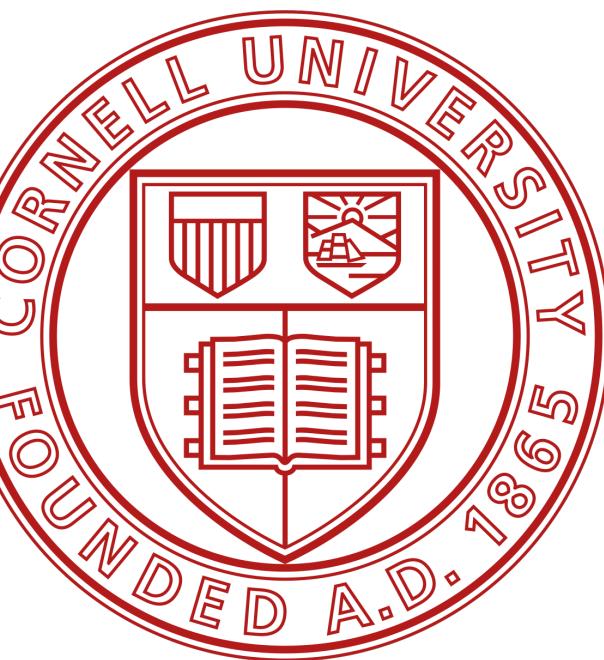


Updating R

R language



UPDATE...

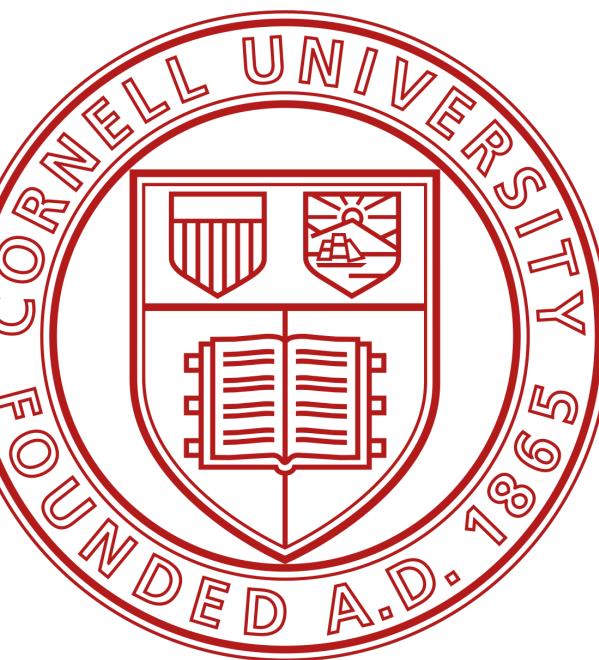


Updating R

R language

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





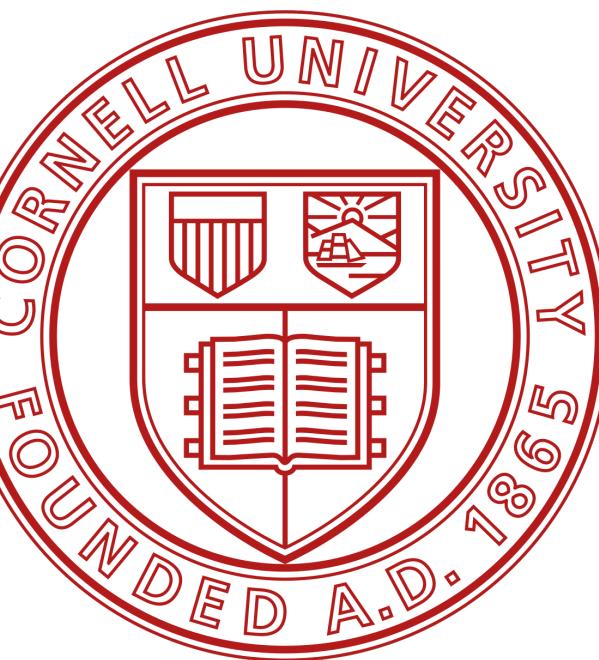
Updating R

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](#).



UPDATE...

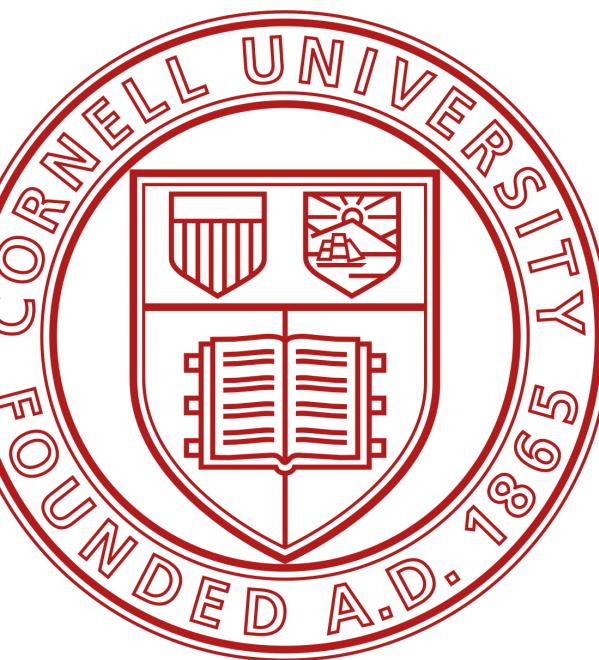


Updating R

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.





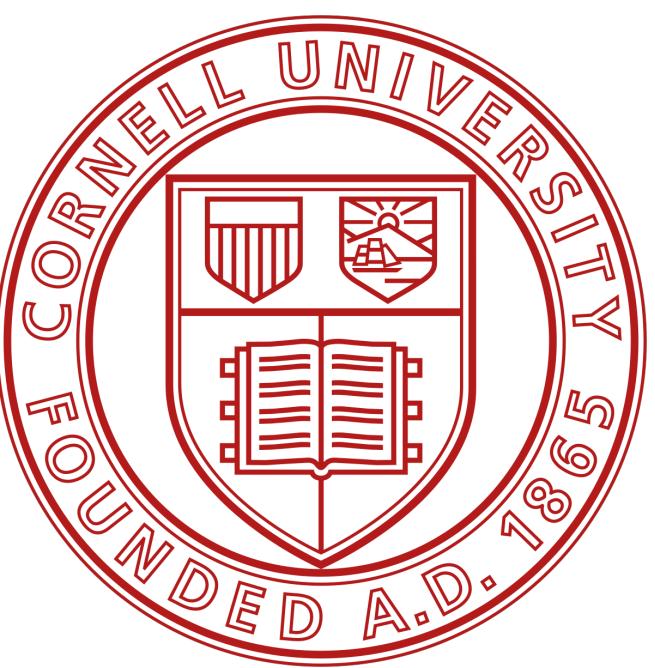
Updating R

R language

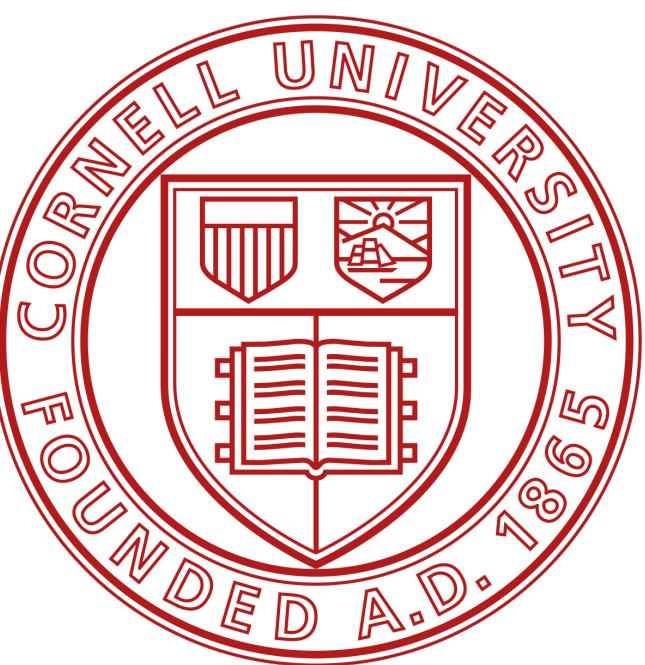
- New versions of R are released several times a year.
- The easiest way to stay current with R is to periodically check [the CRAN website](#).
- The process is the same as when you first installed R.
- Updating to the current version of R is a good place to start if you ever encounter a bug that you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



Updating R R packages



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

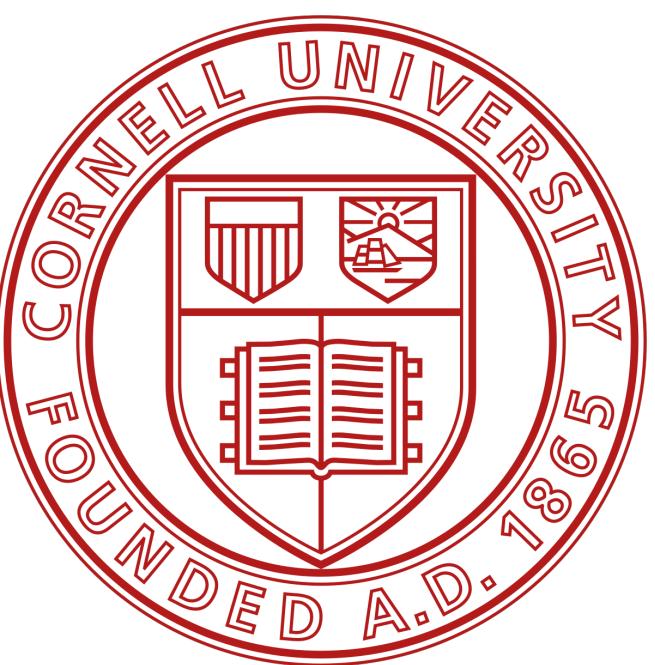


Updating R

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"))
```

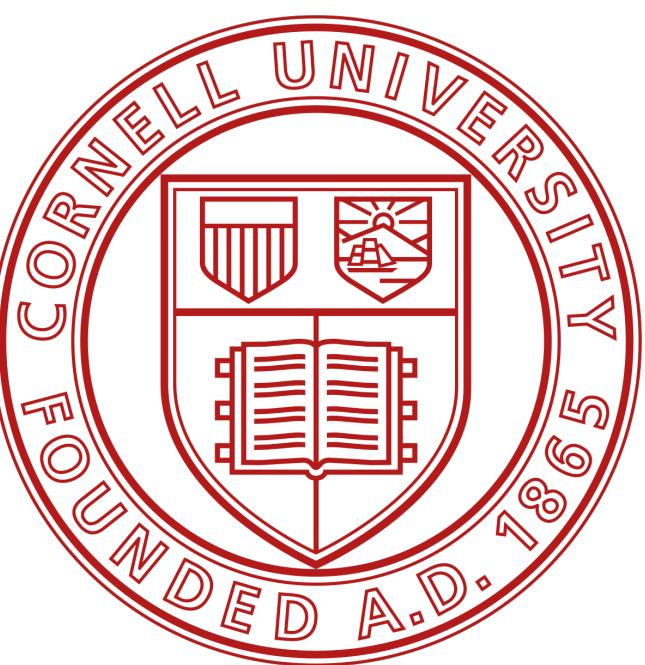


Updating R

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"))
```

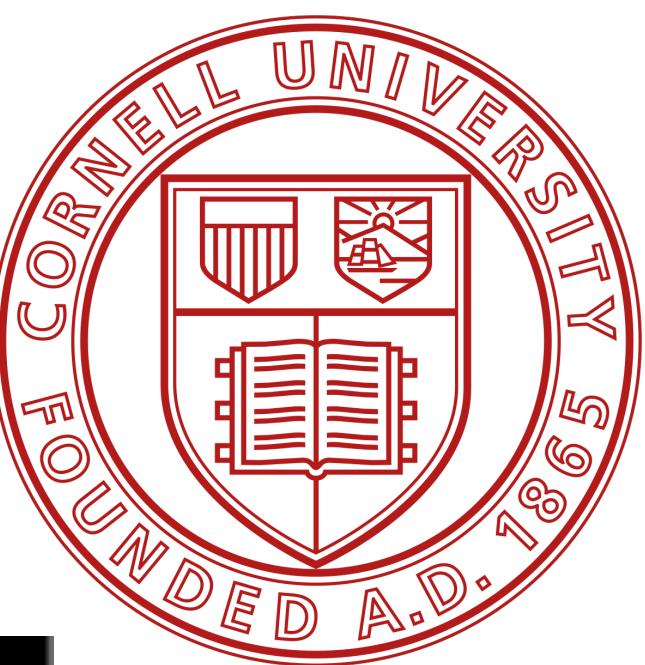


Updating R

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"))
```

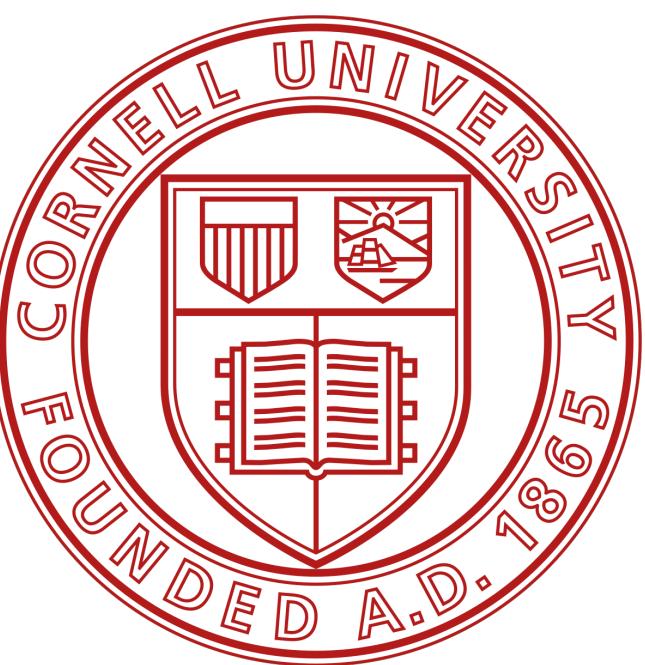


User guide

Comprehensive overview

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



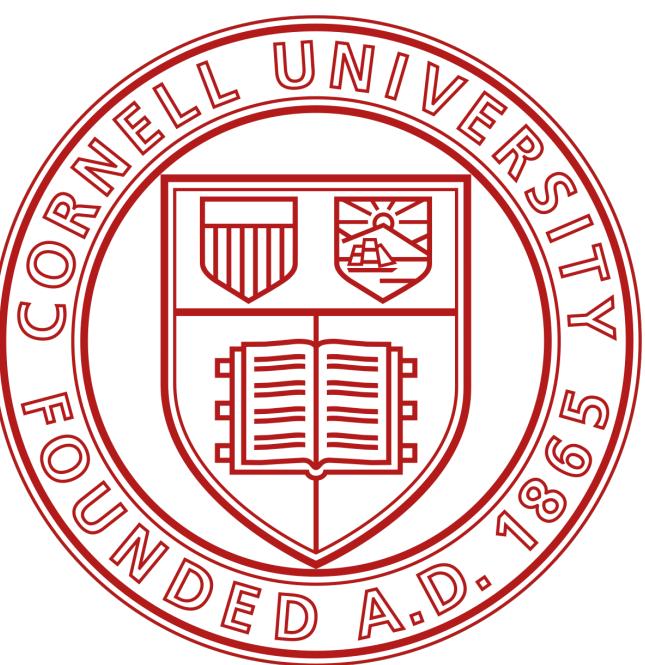


Running R code

Basic command

A screenshot of an R console window. The title bar shows two tabs: "Console" and "Terminal". The "Console" tab is active. The R logo icon is visible next to the version number. The console output shows the command `> 1+1` followed by the result `[1] 2`. A blue cursor arrow is positioned below the last line of output.

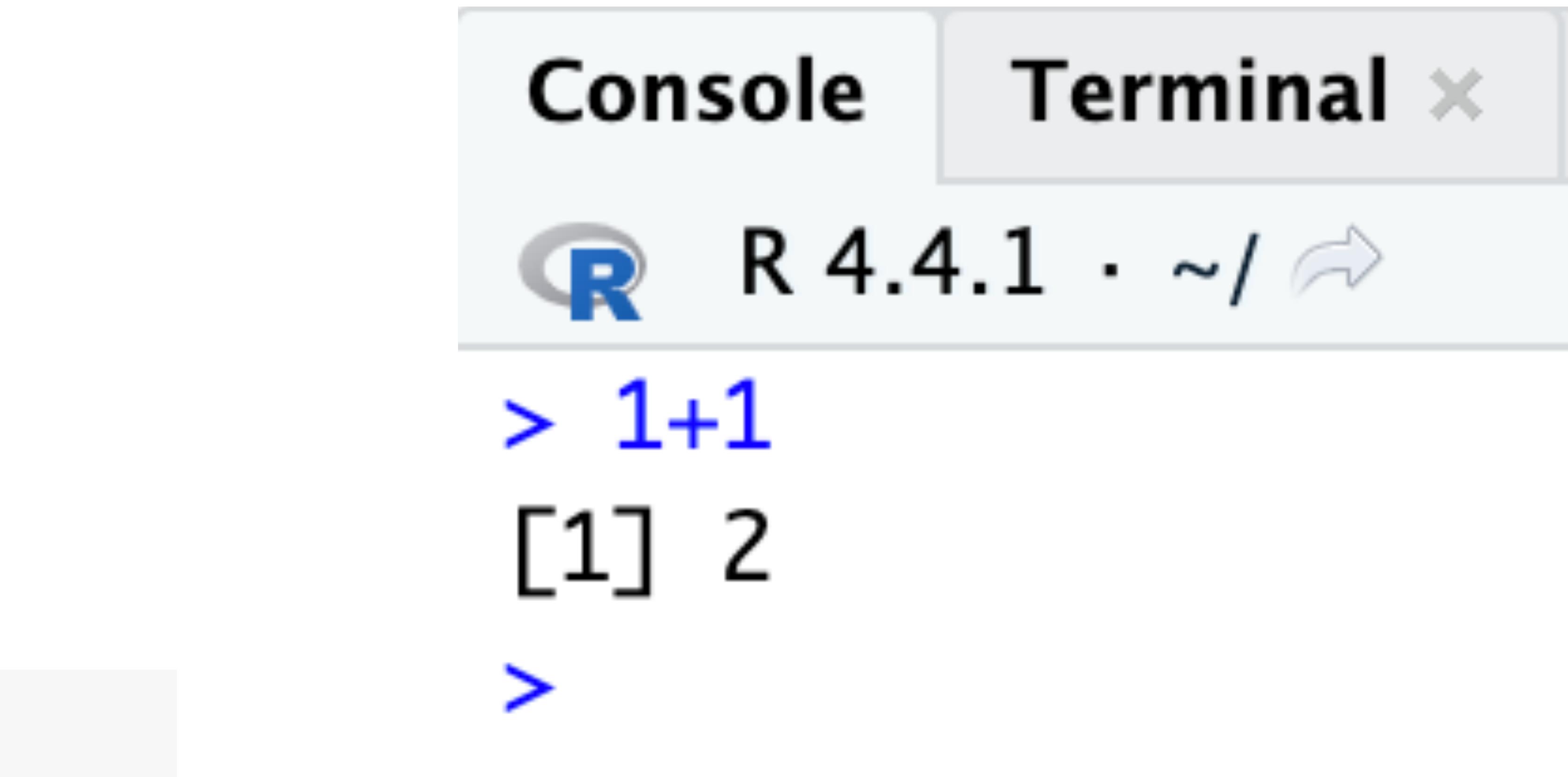
```
R 4.4.1 · ~/ →
> 1+1
[1] 2
>
```



Running R code

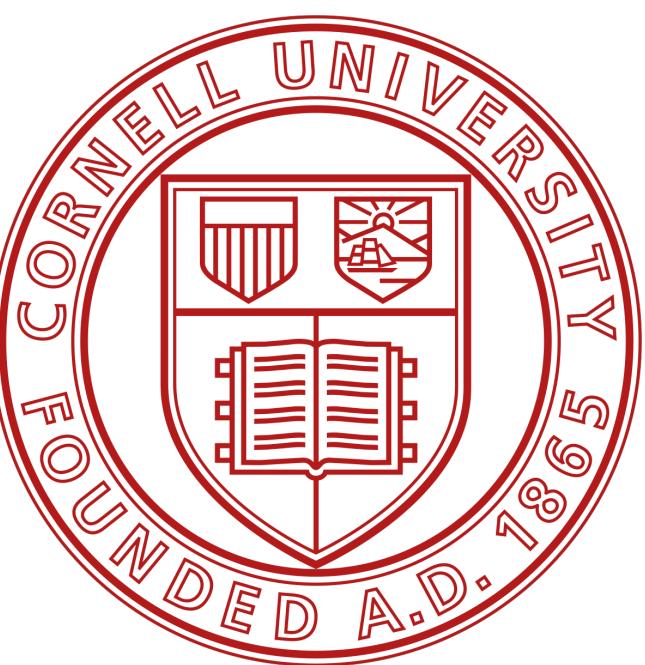
Basic command

- Open RStudio.



A screenshot of the RStudio interface showing the Console tab active. The R logo icon is visible next to the text "R 4.4.1 · ~/". Below the logo, the console history shows the command "`> 1+1`" followed by its output "[1] 2". A blue cursor arrow is positioned below the "[1] 2" line, indicating where the next command will be entered.

```
R 4.4.1 · ~/> 1+1
[1] 2
```

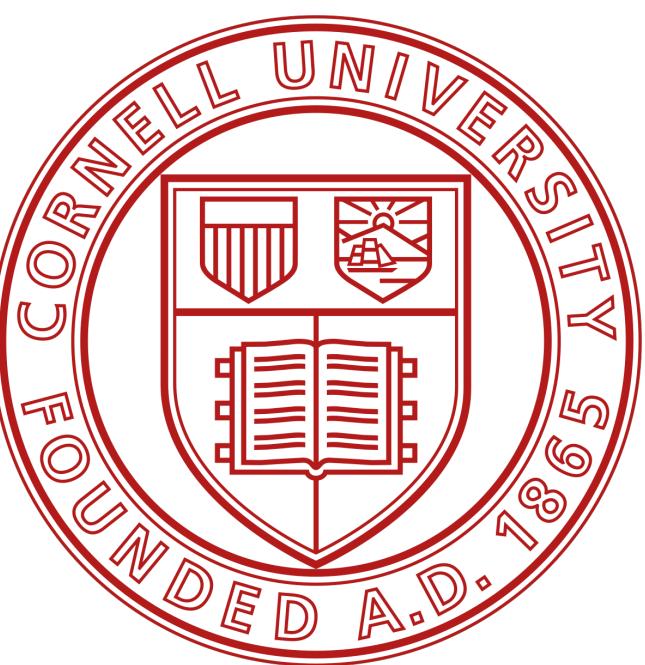


Running R code

Basic command

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

A screenshot of the RStudio interface. On the left, there's a small gray square. In the center, the 'Console' tab is active, indicated by a dark gray background. To its right is the 'Terminal' tab, which has a light gray background. The 'Console' tab shows the R logo icon followed by 'R 4.4.1 · ~ /' and a blue arrow icon. Below this, the R command '`> 1+1`' is typed in blue, and the result '[1] 2' is displayed in black. At the bottom of the console pane, there's another blue arrow icon. The overall background of the RStudio window is white.

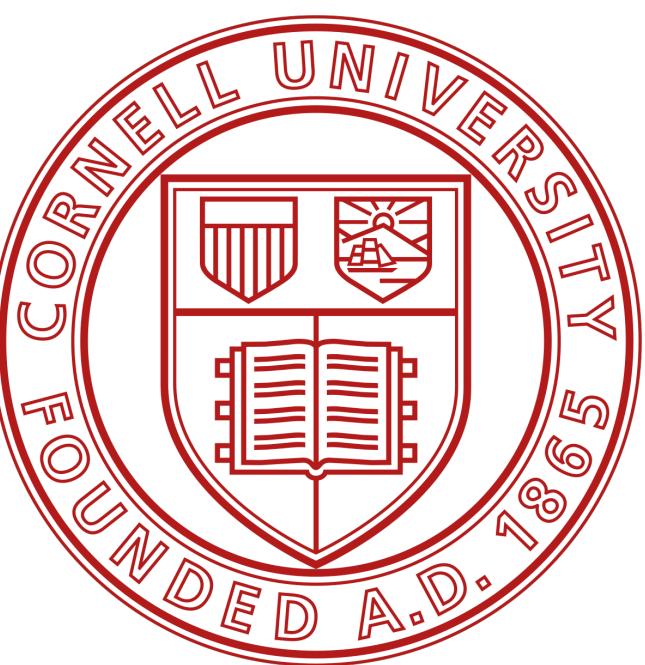


Running R code

Basic command

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

A screenshot of the RStudio interface. On the left, there's a small gray square. In the center, the 'Console' tab is active, showing the R logo icon, the text 'R 4.4.1 · ~ /' followed by a copy icon, and then the R code '1+1' in blue and its output '[1] 2' in black. Below the code, there's another blue '1+1'. On the right, the 'Terminal' tab is visible but inactive, showing a close button.

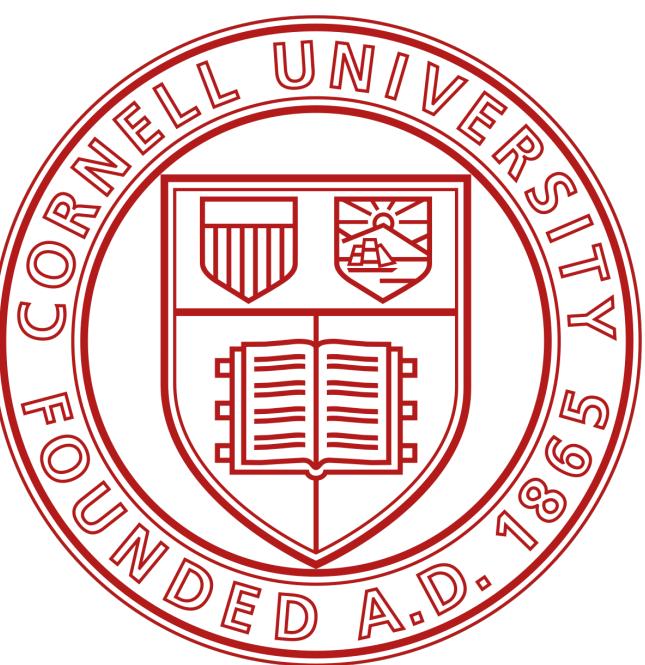


Running R code

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*.

A screenshot of the RStudio interface. On the left, there's a small gray square. In the center, the 'Console' tab is active, showing the R logo icon, the text 'R 4.4.1 · ~ /' followed by a copy icon, and then the R command '> 1+1' which is highlighted in blue. Below that, the output '[1] 2' is shown in black. At the bottom of the console window, there's a blue '>' symbol. To the right of the console, the 'Terminal' tab is visible but inactive, showing a gray 'x' icon.

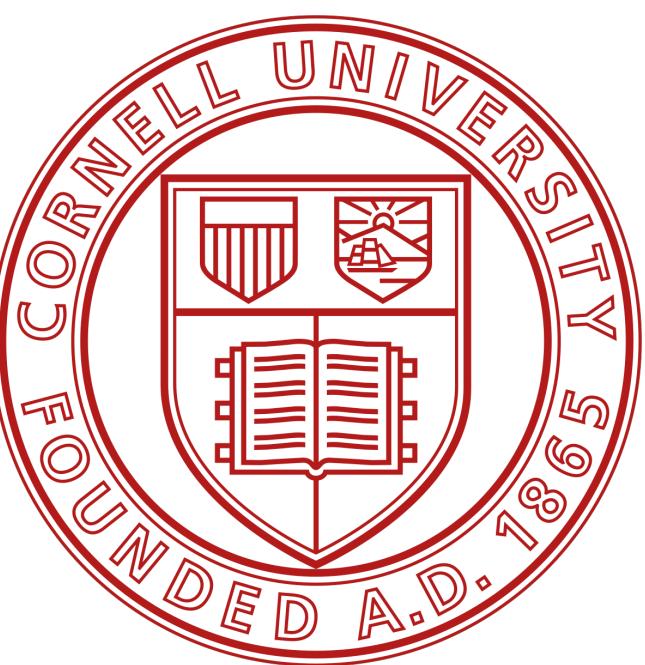


Running R code

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*.

A screenshot of the RStudio interface. On the left, there's a small gray square icon. In the center, the 'Console' tab is active, showing the R logo and the text 'R 4.4.1 · ~/'. Below that, a blue prompt 'R>' is followed by the command '1+1' in blue. The output, '[1] 2', is displayed in black. At the bottom of the console window, there's a blue arrow pointing right. To the right of the console, the 'Terminal' tab is visible with a close button.

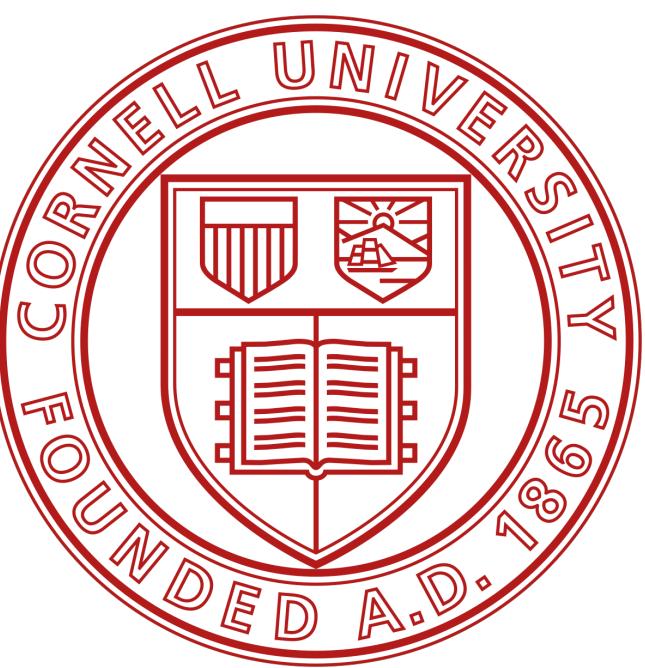


Running R code

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:

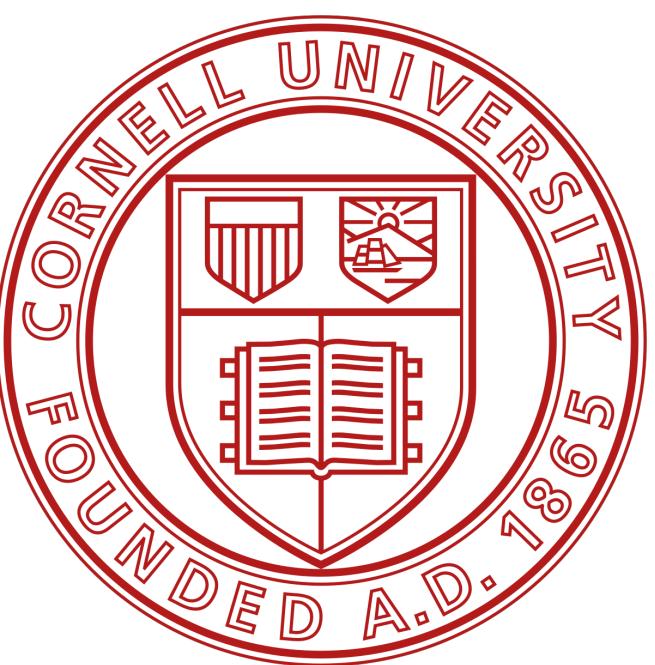
A screenshot of the RStudio interface. On the left, there's a "Console" tab and a "Terminal" tab; the "Console" tab is active. The R logo icon is visible next to the text "R 4.4.1 · ~/". In the console window, the command `> 1+1` is typed in blue, followed by the output `[1] 2`. A blue arrow points from the text "For example, if you type 1 + 1 and hit Enter, RStudio will display:" in the previous slide towards the console window.



Basics

Sequence of numbers

```
Console Terminal x
R 4.4.1 · ~/ ↗
> 100:130
[1] 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122
[24] 123 124 125 126 127 128 129 130
> |
```



Basics

Sequence of numbers

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

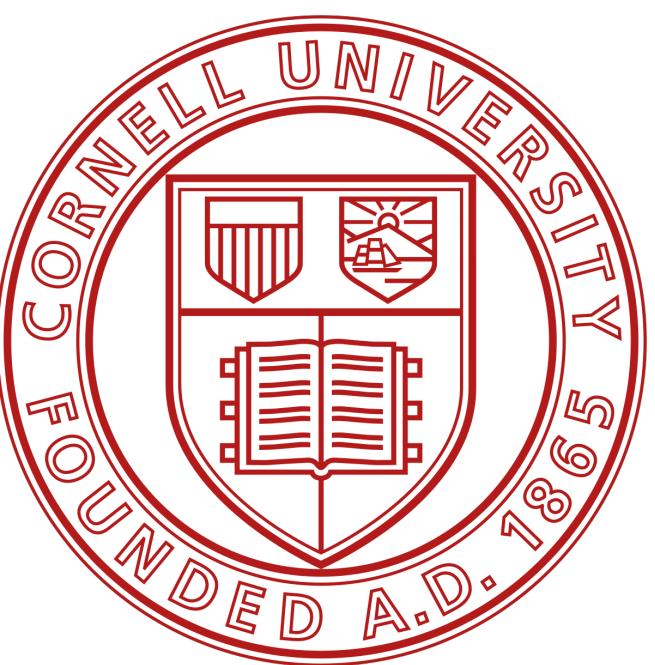
A large rectangular area of the R console output has been redacted with a light gray color.

Console Terminal x

R 4.4.1 · ~/ ↗

> 100:130

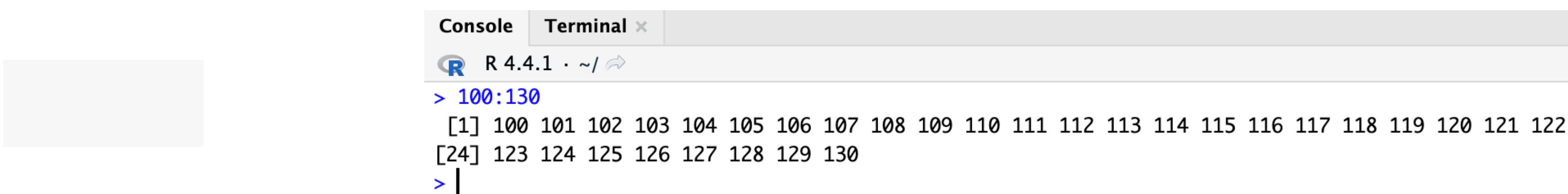
```
[1] 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122
[24] 123 124 125 126 127 128 129 130
> |
```



Basics

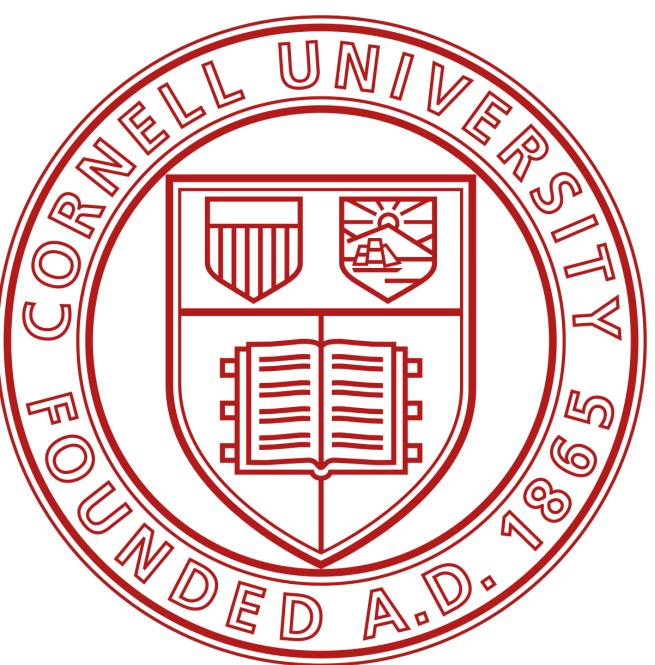
Sequence of numbers

- 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.



A screenshot of an R console window. The title bar says "Console Terminal x". The R logo icon and "R 4.4.1 · ~/..." are visible. The console output shows:

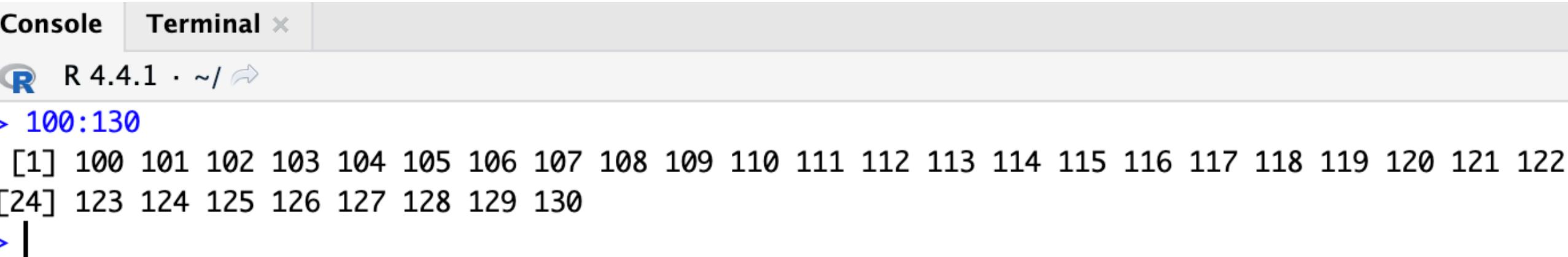
```
R 4.4.1 · ~/...
> 100:130
[1] 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122
[24] 123 124 125 126 127 128 129 130
> |
```



Basics

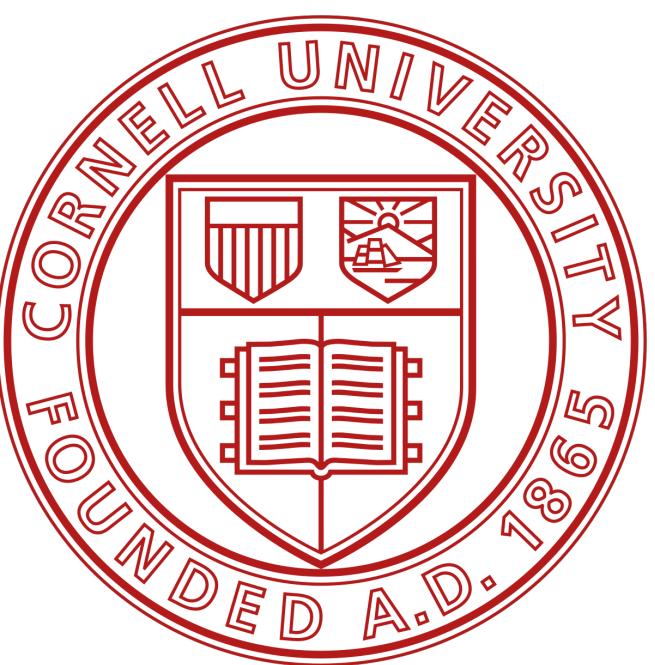
Sequence of numbers

- 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.



A screenshot of an R console window. The title bar says "Console Terminal". The R logo and version "R 4.4.1" are visible. The console output shows:

```
R 4.4.1 · ~/ ↗
> 100:130
[1] 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122
[24] 123 124 125 126 127 128 129 130
> |
```



Basics

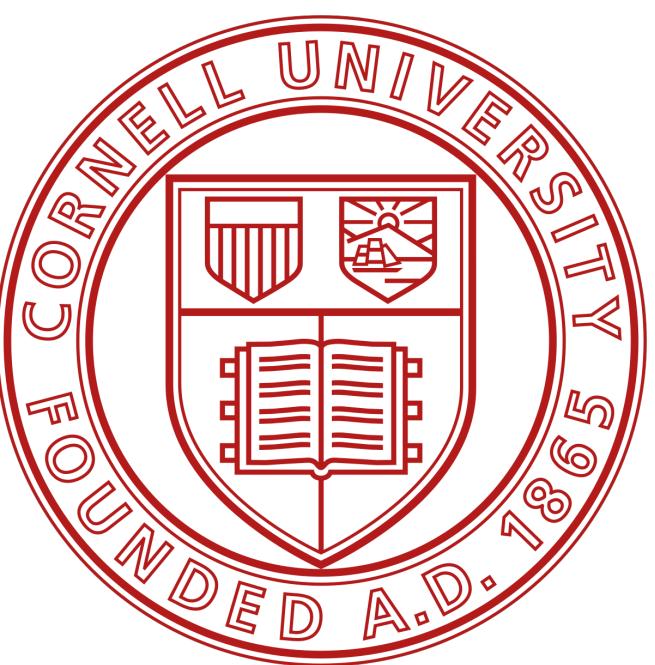
Sequence of numbers

- 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.

A screenshot of an R console window. The title bar says "Console Terminal". The R logo icon and "R 4.4.1 · ~/..." are visible. Below the title bar, there is a blue link-like text "100:130". Underneath, two lines of code output are shown:

```
[1] 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122  
[24] 123 124 125 126 127 128 129 130
```

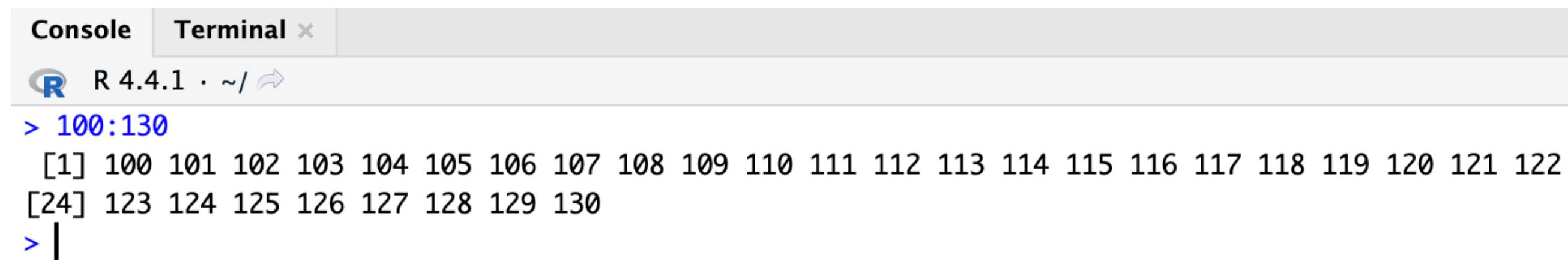
A cursor arrow points to the right after the first line of output.



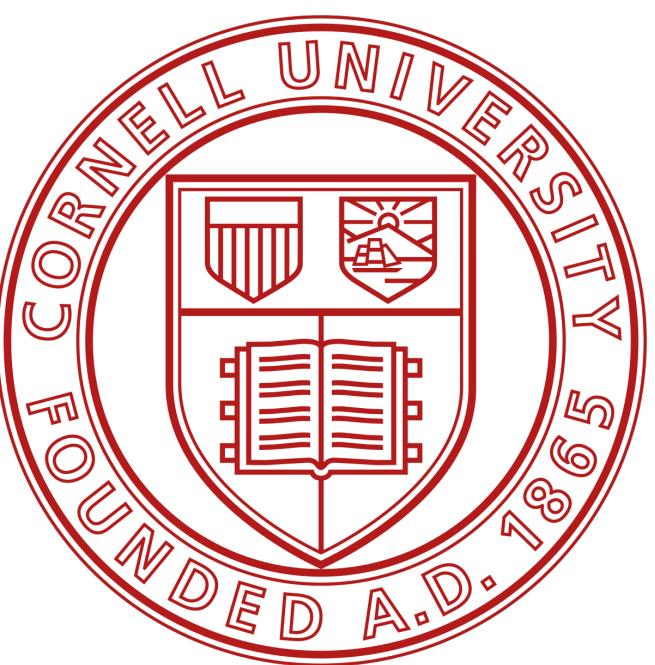
Basics

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.



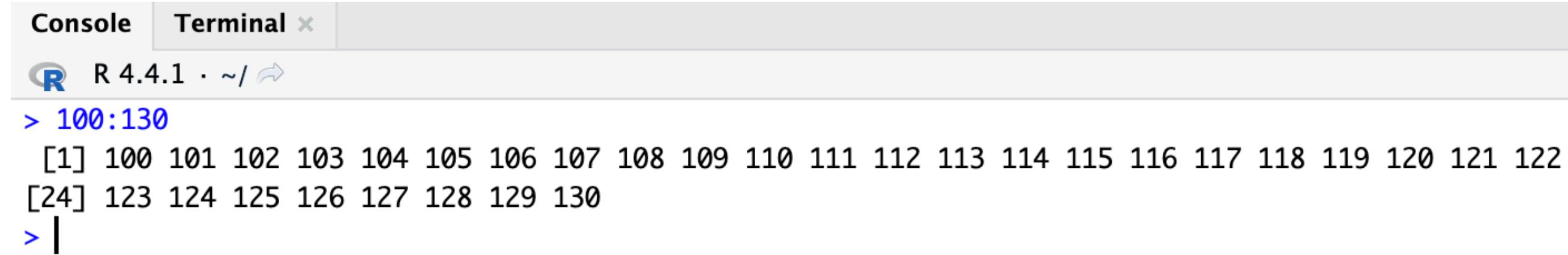
A screenshot of an R console window. The title bar says "Console Terminal". The R logo and version "R 4.4.1 · ~/..." are visible. The command "> 100:130" is entered. The output shows two lines: "[1] 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122" and "[24] 123 124 125 126 127 128 129 130". A cursor is shown at the end of the second line.



Basics

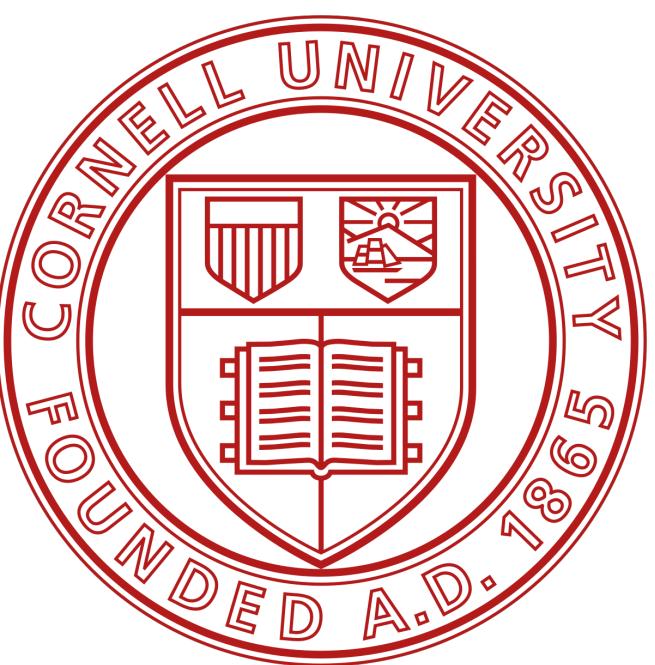
Sequence of numbers

- 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.



A screenshot of an R console window. The title bar says "Console Terminal". The R logo icon and "R 4.4.1 · ~/..." are visible. The console output shows:

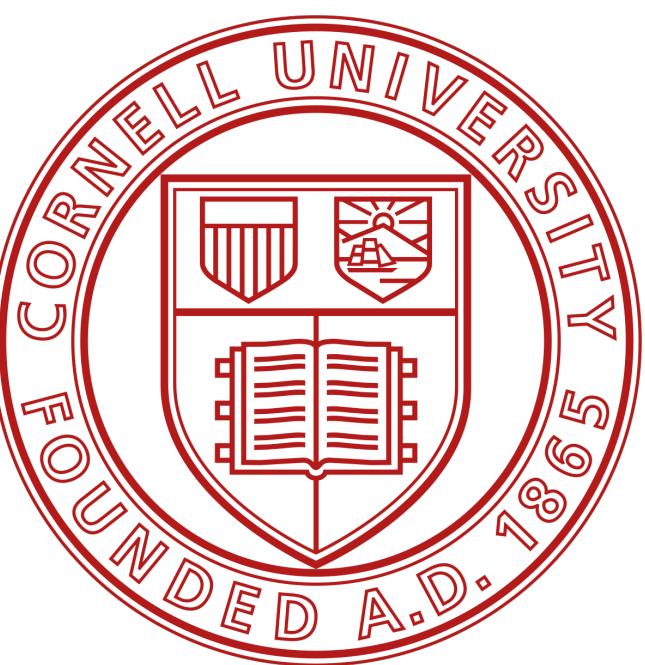
```
R 4.4.1 · ~/ 
> 100:130
[1] 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122
[24] 123 124 125 126 127 128 129 130
> |
```



Basics

Incomplete command

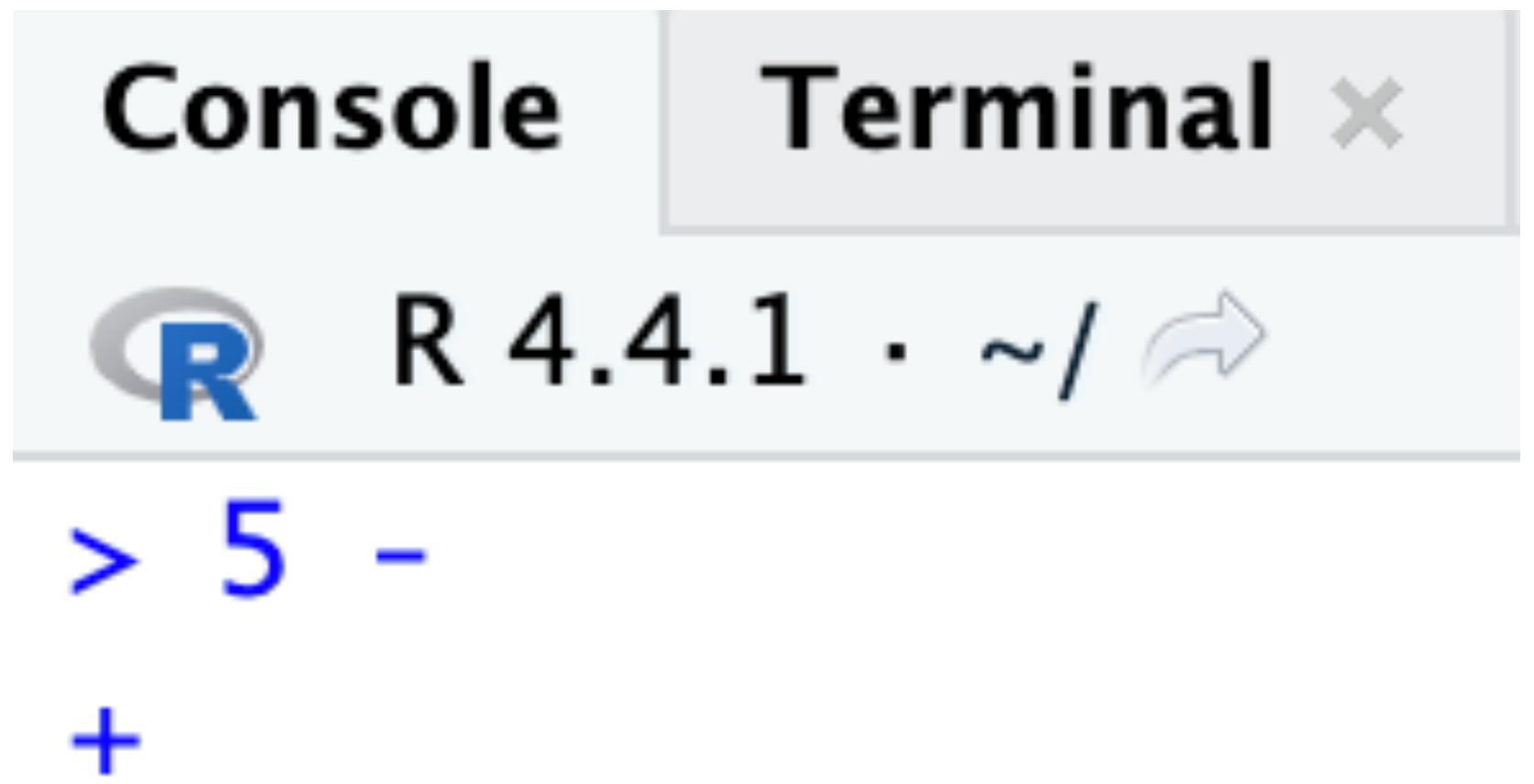
```
Console Terminal ×  
R 4.4.1 · ~/ ↗  
> 5 -  
+  
  
R 4.4.1 · ~/ ↗  
> 5 -  
+ 1  
[1] 4
```



Basics

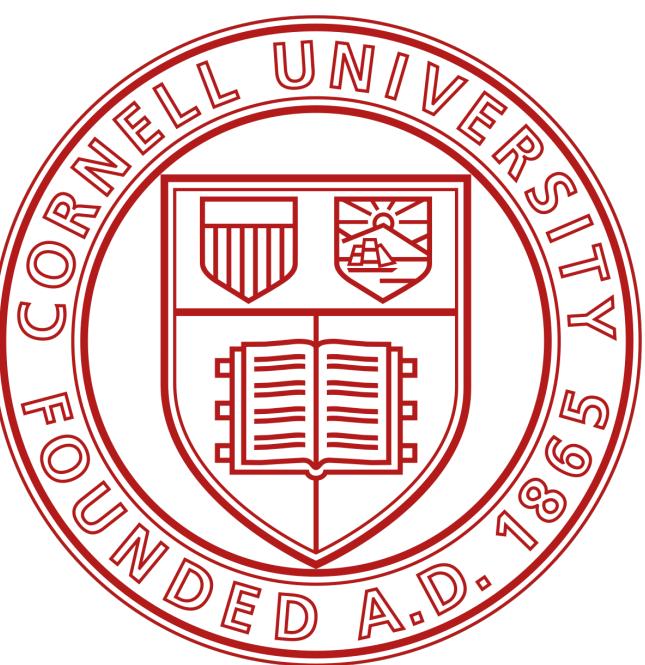
Incomplete command

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



The screenshot shows two panels of an R console interface. The top panel is labeled "Console" and the bottom panel is labeled "Terminal". Both panels show the R logo and the text "R 4.4.1 · ~/". In the "Console" panel, the user has typed "5 -" and pressed Enter, which is followed by a blue "+". In the "Terminal" panel, the user has typed "5 -" and pressed Enter, which is followed by a blue "+". This illustrates that R displays a + prompt after an incomplete command.

```
R 4.4.1 · ~/ 
> 5 -
+
R 4.4.1 · ~/ 
> 5 -
+
[1] 4
```



Basics

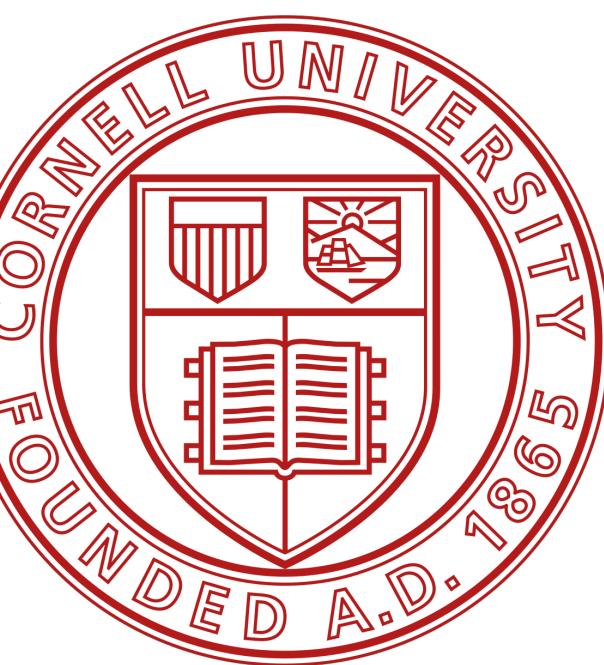
Incomplete command

- 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.



The screenshot shows two panels of an R console interface. The top panel is titled "Console" and the bottom panel is titled "Terminal". Both panels show the R logo and the text "R 4.4.1 · ~/". In the "Console" panel, the command "> 5 -" is entered, followed by a blue "+". In the "Terminal" panel, the command "> 5 -" is entered, followed by a blue "+", and then the result "[1] 4" is displayed in black text.

```
R 4.4.1 · ~/ 
> 5 -
+
R 4.4.1 · ~/ 
> 5 -
+
[1] 4
```



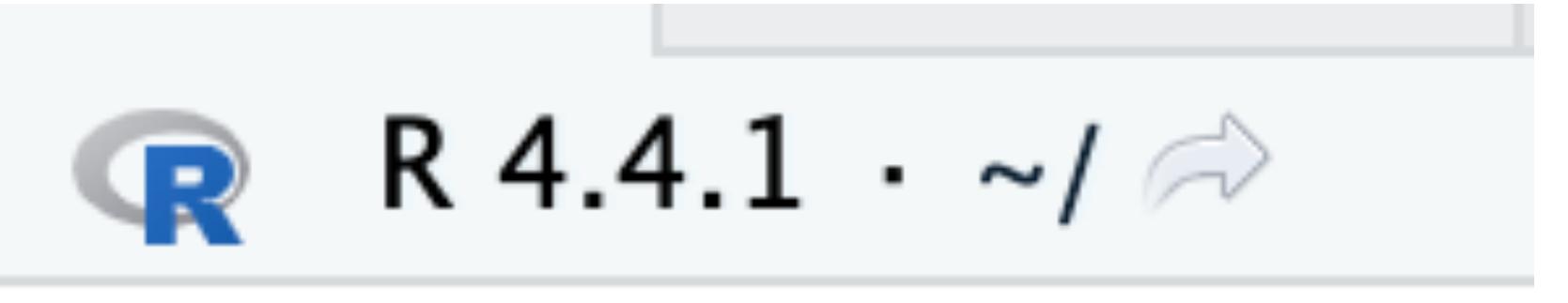
Basics

Incomplete command

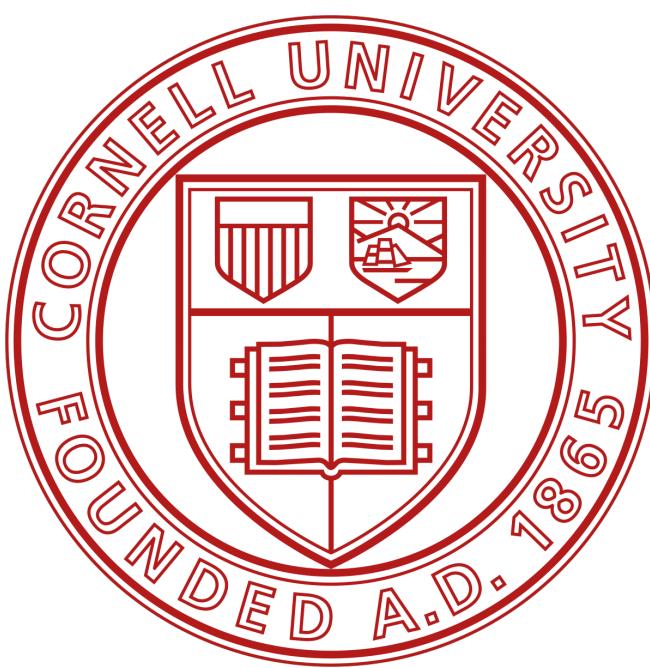
- If you type an incomplete command and press Enter, R will display a + prompt.
- It means R is waiting for you to type the rest of your command.
- Either finish the command or hit Escape to start over.



The screenshot shows the R console interface. The title bar says "Console Terminal". The main area displays the R logo followed by "R 4.4.1 · ~/". Below this, there are two lines of text: "> 5 -" and "+".



The second screenshot shows the R console after the user has completed the command. The text now reads "> 5 -" on the first line and "[1] 4" on the second line, indicating the result of the computation.



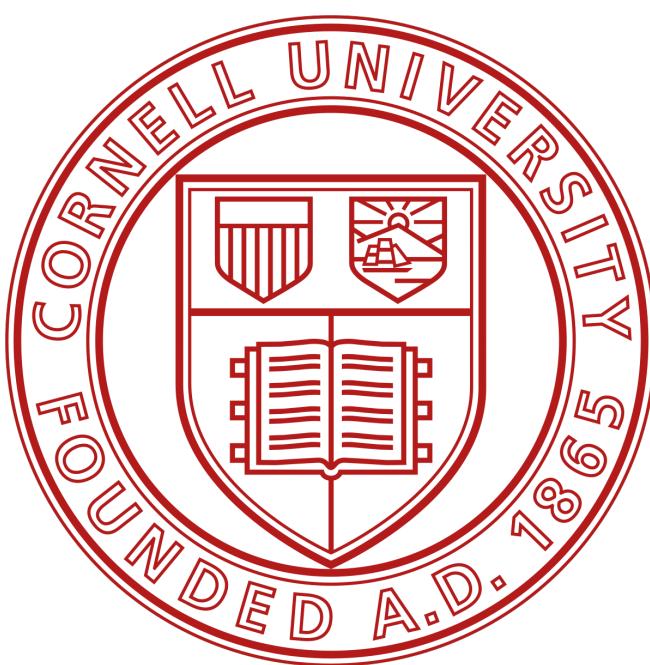
Basics

Error message

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

A screenshot of an R console window. The window has tabs for "Console" and "Terminal", with "Console" selected. The R logo icon is visible next to the text "R 4.4.1 · ~/". In the console area, the user has typed the command "> 3%5" and received the error message "Error: unexpected input in \"3%5\"". A cursor is shown at the end of the line where the command was entered.

```
Console Terminal ×
R 4.4.1 · ~/ 
> 3%5
Error: unexpected input in "3%5"
> |
```

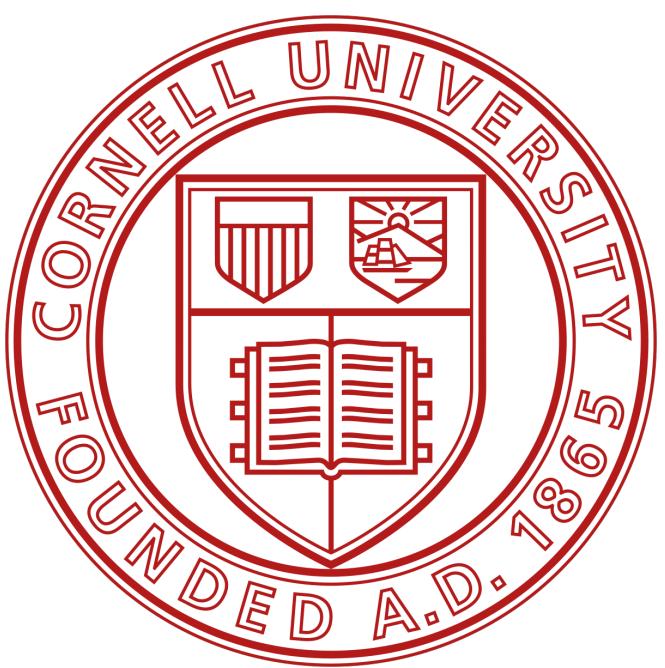


Basics

Calculator

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

```
Console Terminal ×
R 4.4.1 · ~/ ↗
> 2*3
[1] 6
> 4-1
[1] 3
> 6/(4-1)
[1] 2
>
```



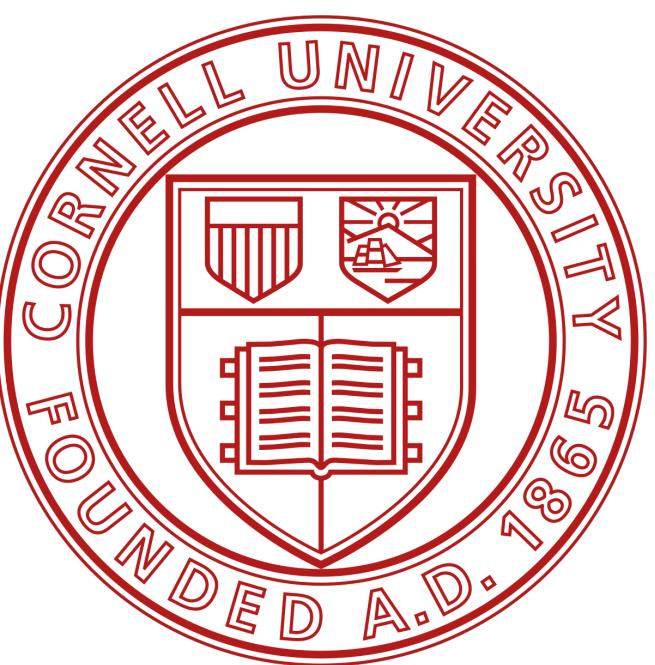
Basics

Comments

Console Terminal ×

R 4.4.1 · ~/ ➔

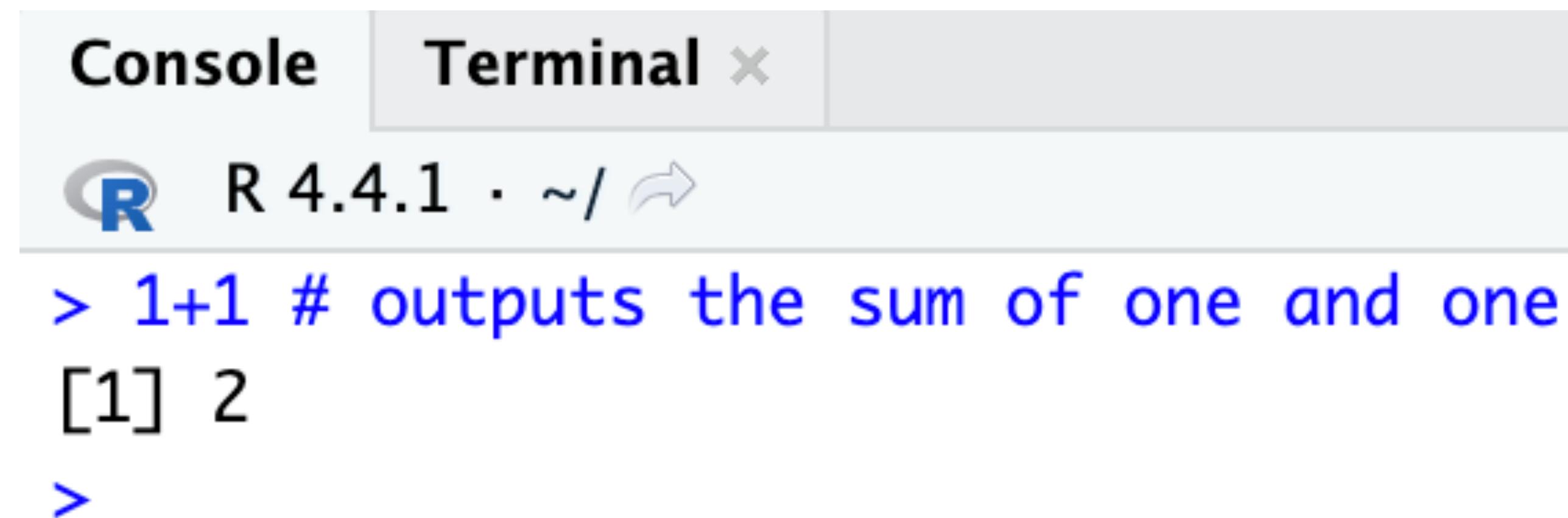
```
> 1+1 # outputs the sum of one and one
[1] 2
>
```



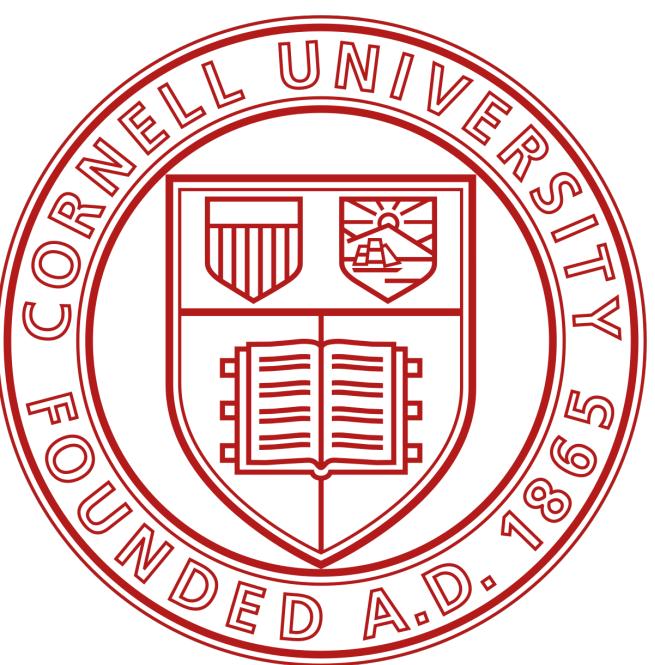
Basics

Comments

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

A screenshot of the RStudio interface showing the Terminal tab. The R logo icon is visible next to the text "R 4.4.1 · ~/". Below it, a command is entered: > `1+1 # outputs the sum of one and one`. The output shows the result: [1] 2. A final prompt > is shown at the bottom.

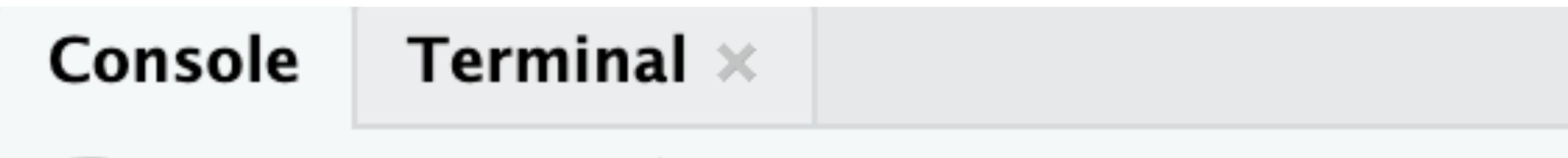
```
Console Terminal ×
R 4.4.1 · ~/ 
> 1+1 # outputs the sum of one and one
[1] 2
>
```



Basics

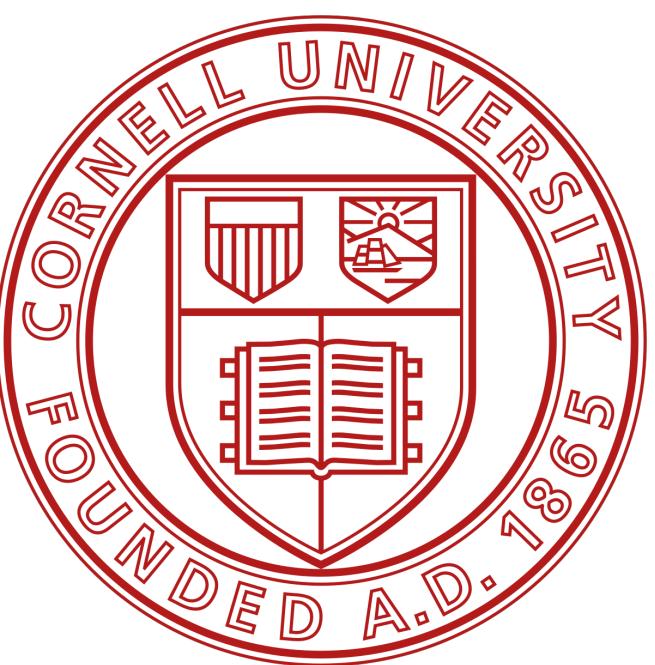
Comments

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



A screenshot of an RStudio interface showing the Terminal tab. The title bar says "Console Terminal". The R logo icon is visible next to the R version "R 4.4.1 · ~/". The terminal window displays the following R session:

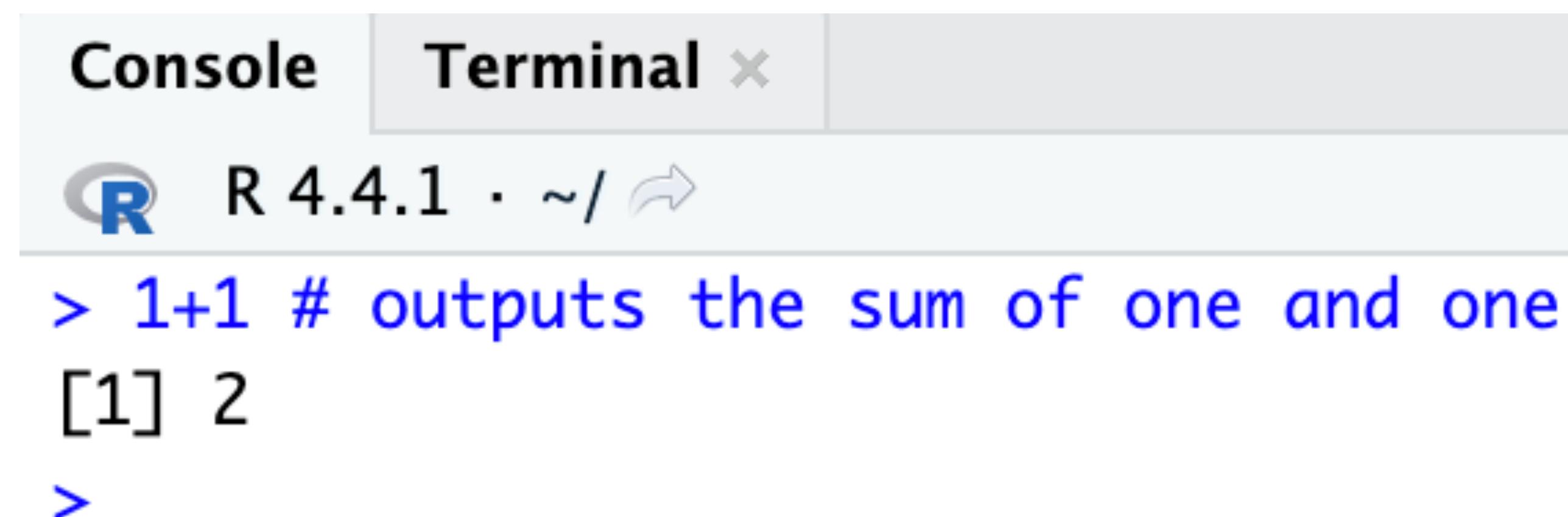
```
R 4.4.1 · ~/ 
> 1+1 # outputs the sum of one and one
[1] 2
>
```



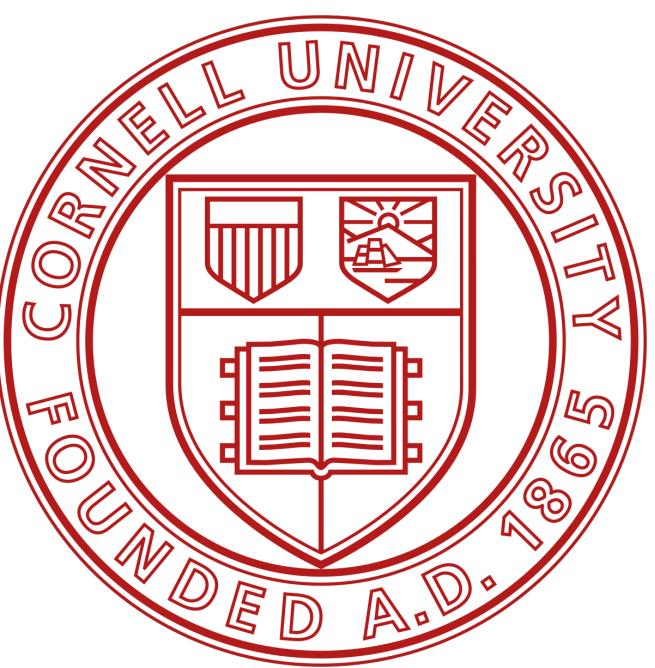
Basics

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.

A screenshot of the RStudio interface showing the Terminal tab. The R logo is visible, followed by "R 4.4.1 · ~/". In the terminal window, the command "`> 1+1 # outputs the sum of one and one`" is entered, and the output "[1] 2" is shown. The "# outputs the sum of one and one" part is highlighted in blue, indicating it is a comment.

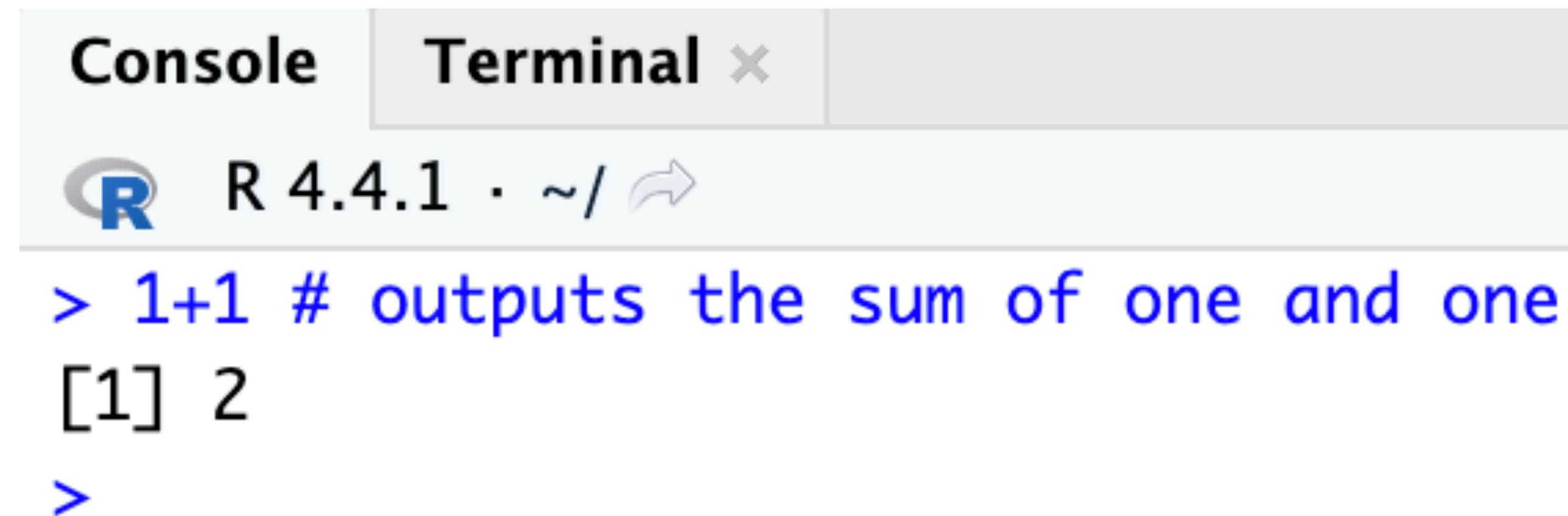
```
R 4.4.1 · ~/ 
> 1+1 # outputs the sum of one and one
[1] 2
```



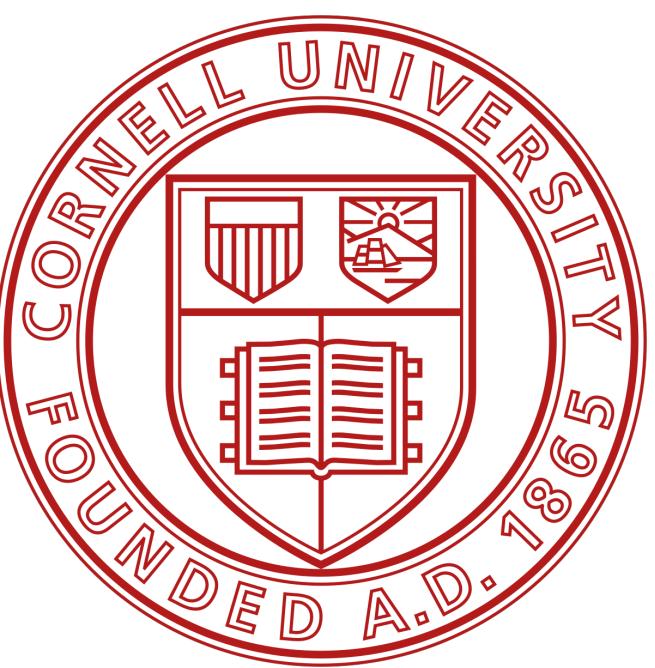
Basics

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.



The screenshot shows an RStudio interface with the 'Terminal' tab selected. The R logo is visible, followed by 'R 4.4.1 · ~/'. In the terminal window, the command `> 1+1 # outputs the sum of one and one` is entered, and the output `[1] 2` is shown. Below the command, a single greater-than sign (`>`) indicates the prompt for the next command.



Basics

Console vs R. file

The screenshot shows the RStudio interface. At the top, there is a script editor window titled "First_lesson.R". The code in the editor is:

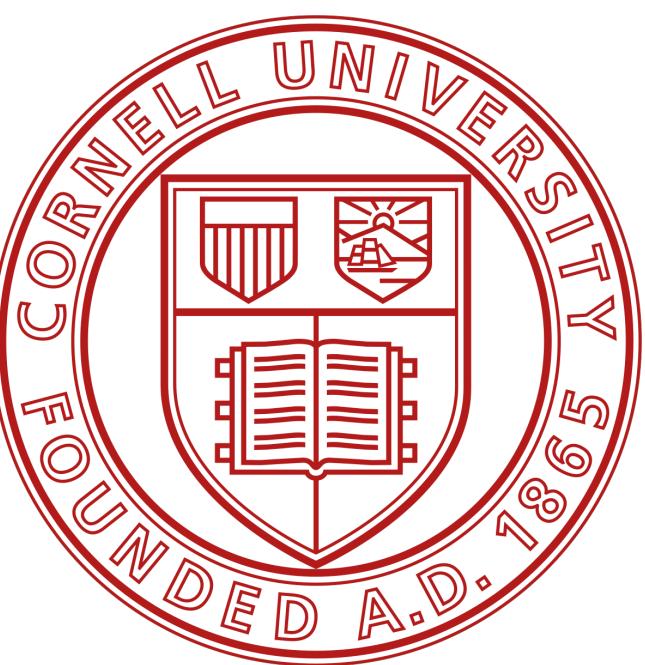
```
1 1+1  
2 2*2  
3 3/5  
4
```

Below the editor, the status bar indicates "4:1 (Top Level) R Script".

At the bottom, there is a console window titled "Console". The console output is:

```
R 4.4.1 · ~/>
```

A blue arrow icon is located at the bottom center of the console window.



Basics

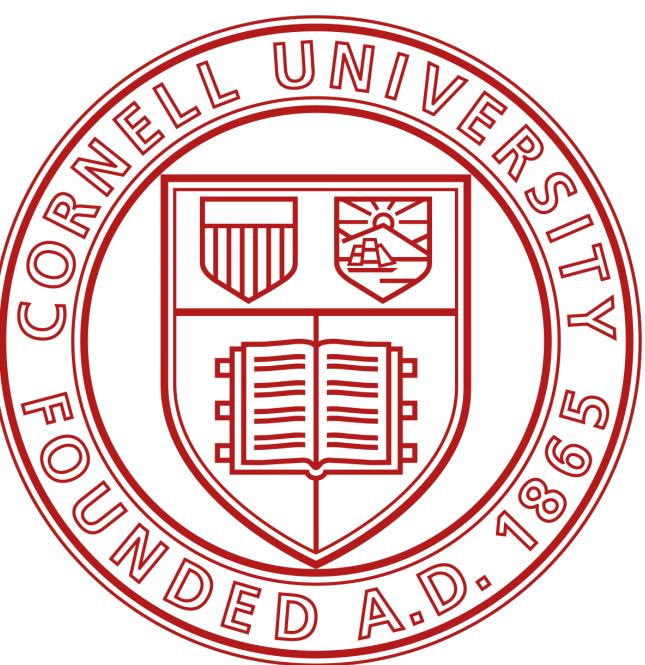
Console vs R. file

- 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 screenshot shows the RStudio interface. The top panel is a script editor titled "First_lesson.R" containing the following code:

```
1 1+1
2 2*2
3 3/5
4
```

The status bar at the bottom of the editor panel indicates "4:1 (Top Level) R Script". The bottom panel is a console window titled "Console" showing the R logo and the text "R 4.4.1 · ~/". A blue arrow cursor is visible at the bottom of the console window.



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.

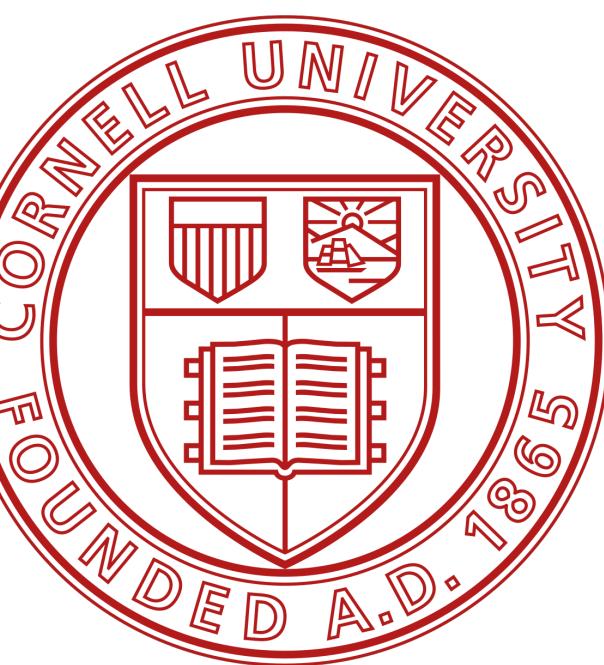
A screenshot of the RStudio interface. The top panel shows a script editor titled "First_lesson.R" containing four lines of R code: "1 1+1", "2 2*2", "3 3/5", and "4". Below the script editor is a status bar with "4:1 (Top Level) R Script". The bottom panel shows a console tab titled "Console" which displays "R 4.4.1 · ~/".

```
1 1+1
2 2*2
3 3/5
4
```

4:1 (Top Level) R Script

Console Terminal

R 4.4.1 · ~/



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.

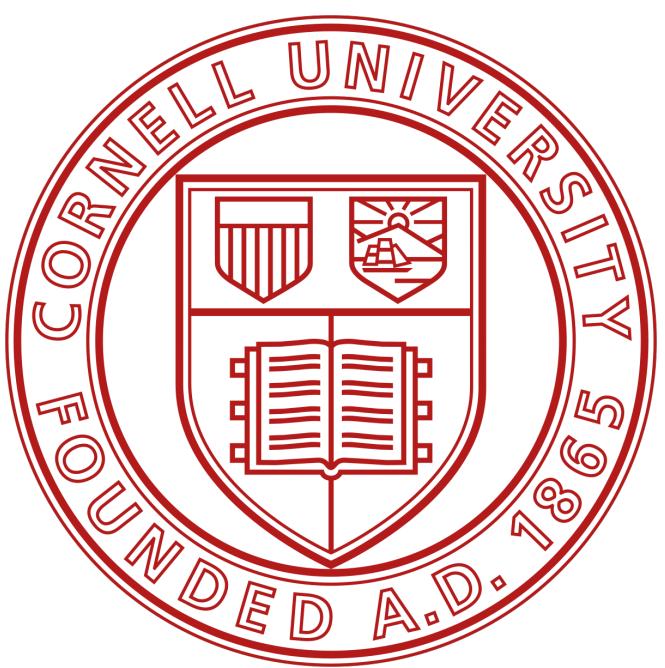
A screenshot of the RStudio interface. The top panel shows a script editor titled "First_lesson.R" containing four lines of R code: "1 1+1", "2 2*2", "3 3/5", and "4". The bottom panel shows a terminal window titled "Console" with the text "R 4.4.1 · ~/".

First_lesson.R

1 1+1
2 2*2
3 3/5
4

Console Terminal

R 4.4.1 · ~/



Basics

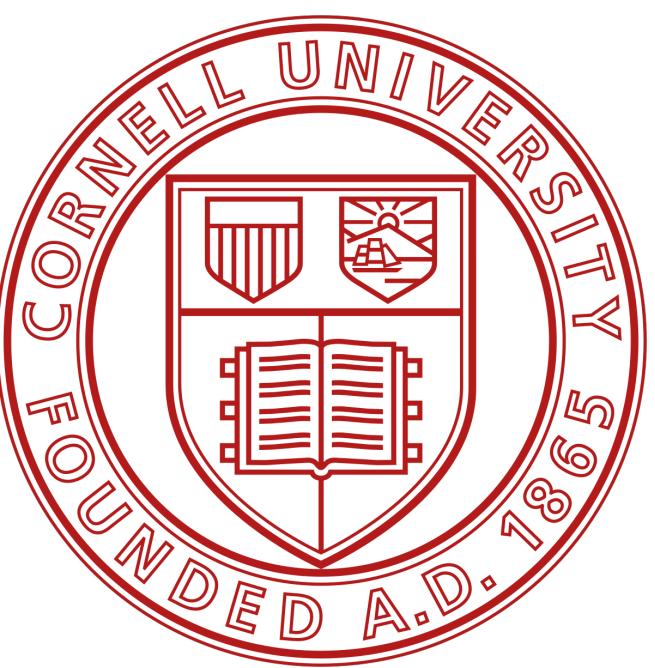
Run vs Source

A screenshot of the RStudio interface. The top menu bar shows two files: "First_lesson.R" and "Exercises_W1.R". The toolbar below includes icons for back, forward, search, and run/source. The main code editor area displays the following R code:

```
1 1+1
2 2*2
3 3/5
4
```

The status bar at the bottom indicates "4:1 (Top Level)".

R Script



Basics

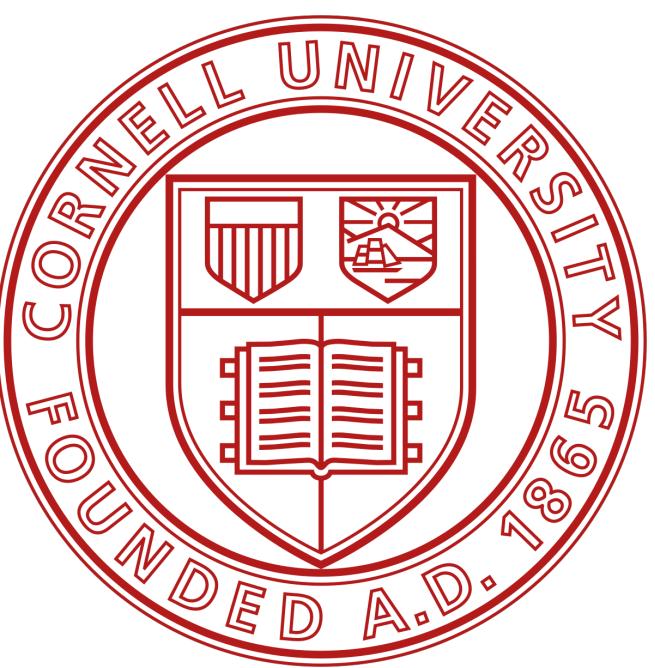
Run vs Source

- 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.

A screenshot of the RStudio interface showing a script editor window. The window title is "R Script". There are two tabs open: "First_lesson.R" and "Exercises_W1.R". The "Exercises_W1.R" tab is active. The code in the editor is:

```
1 1+1
2 2*2
3 3/5
4
```

The status bar at the bottom shows "4:1 (Top Level)". The toolbar above the editor includes icons for back, forward, search, and run/source functions.



Basics

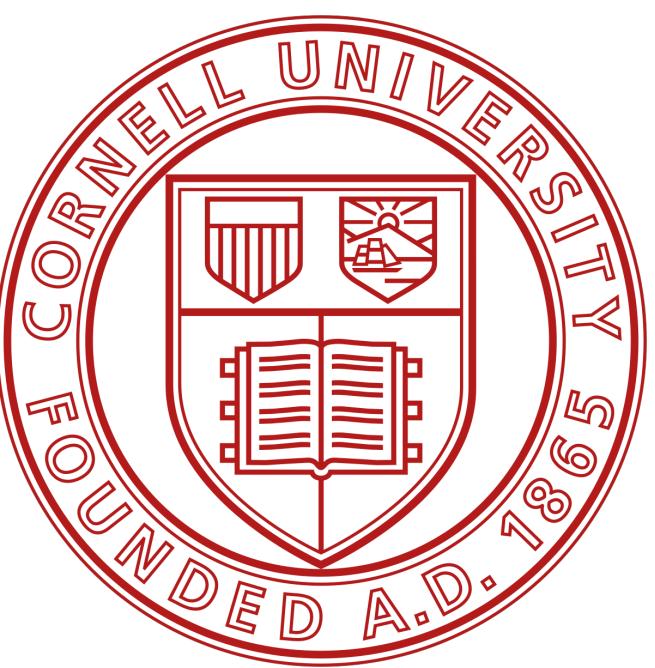
Run vs Source

- 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.

A screenshot of the RStudio interface showing a script editor. The title bar shows two files: "First_lesson.R" and "Exercises_W1.R". The "Exercises_W1.R" tab is active. The editor window displays the following R code:

```
1 1+1
2 2*2
3 3/5
4
```

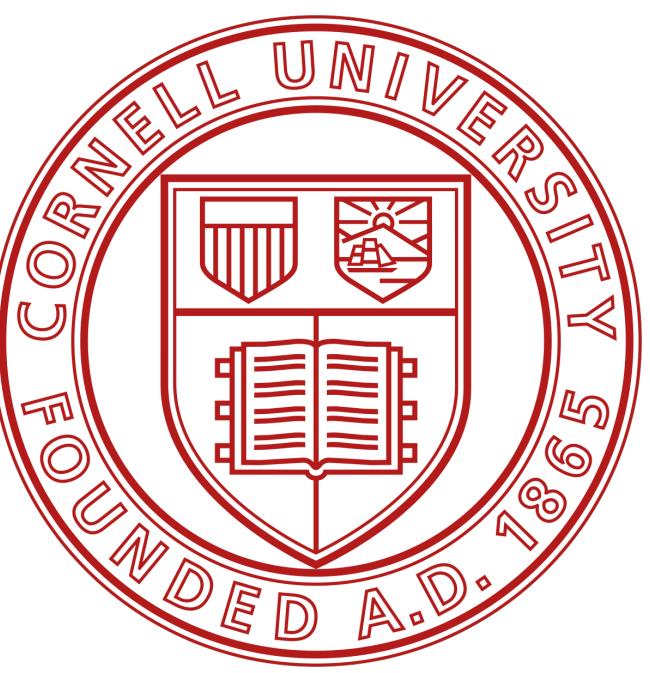
The lines 1, 2, and 3 are highlighted in blue, indicating they are selected for execution. The status bar at the bottom shows "4:1 (Top Level)".

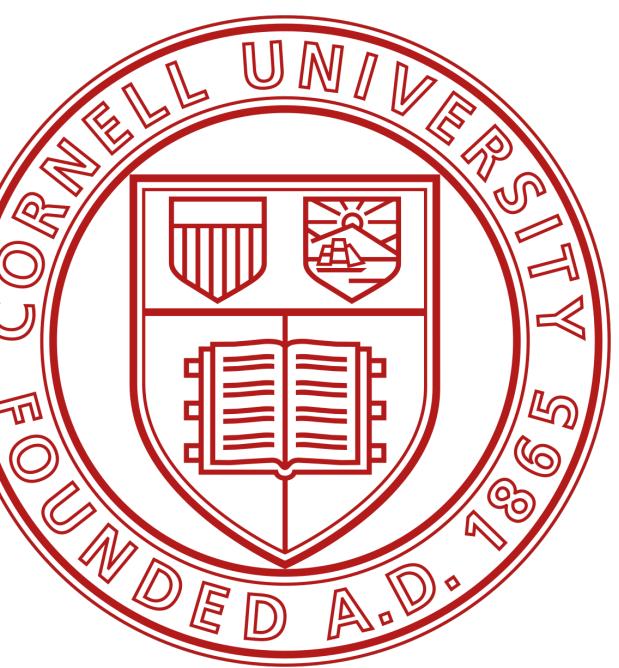


Playing with dices

Basics

Objects

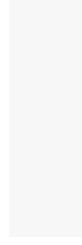


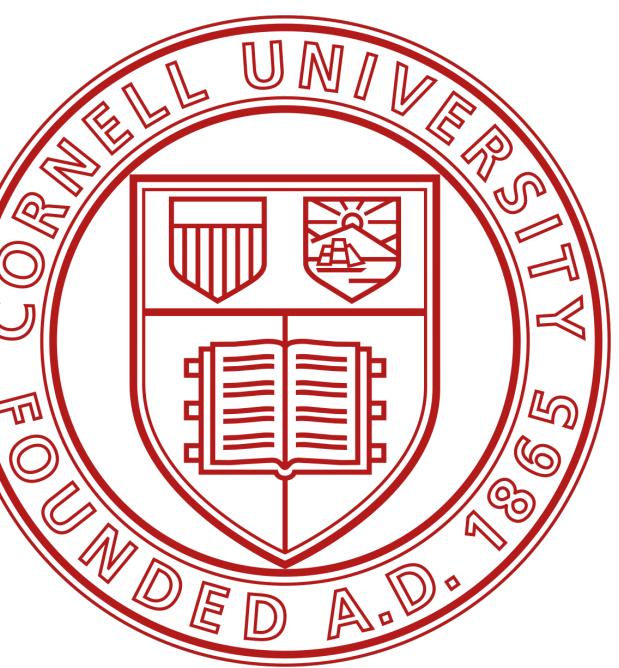


Basics

Objects

- Let's create a virtual die



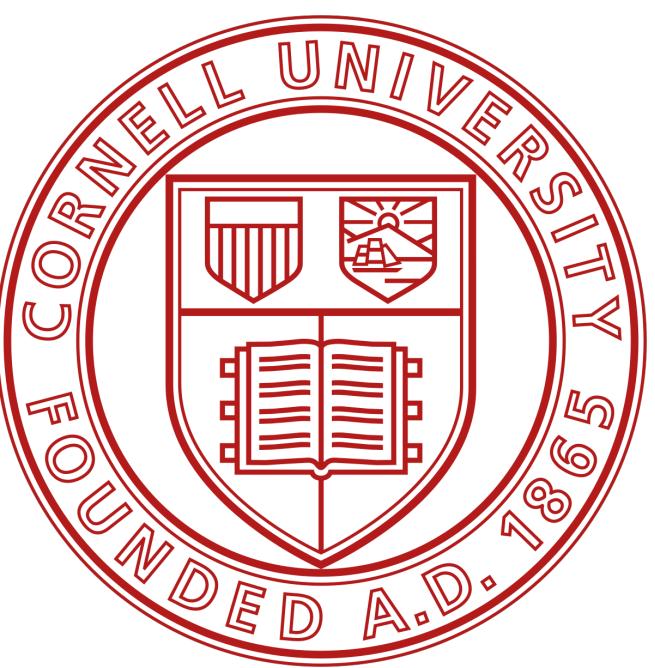


Basics

Objects

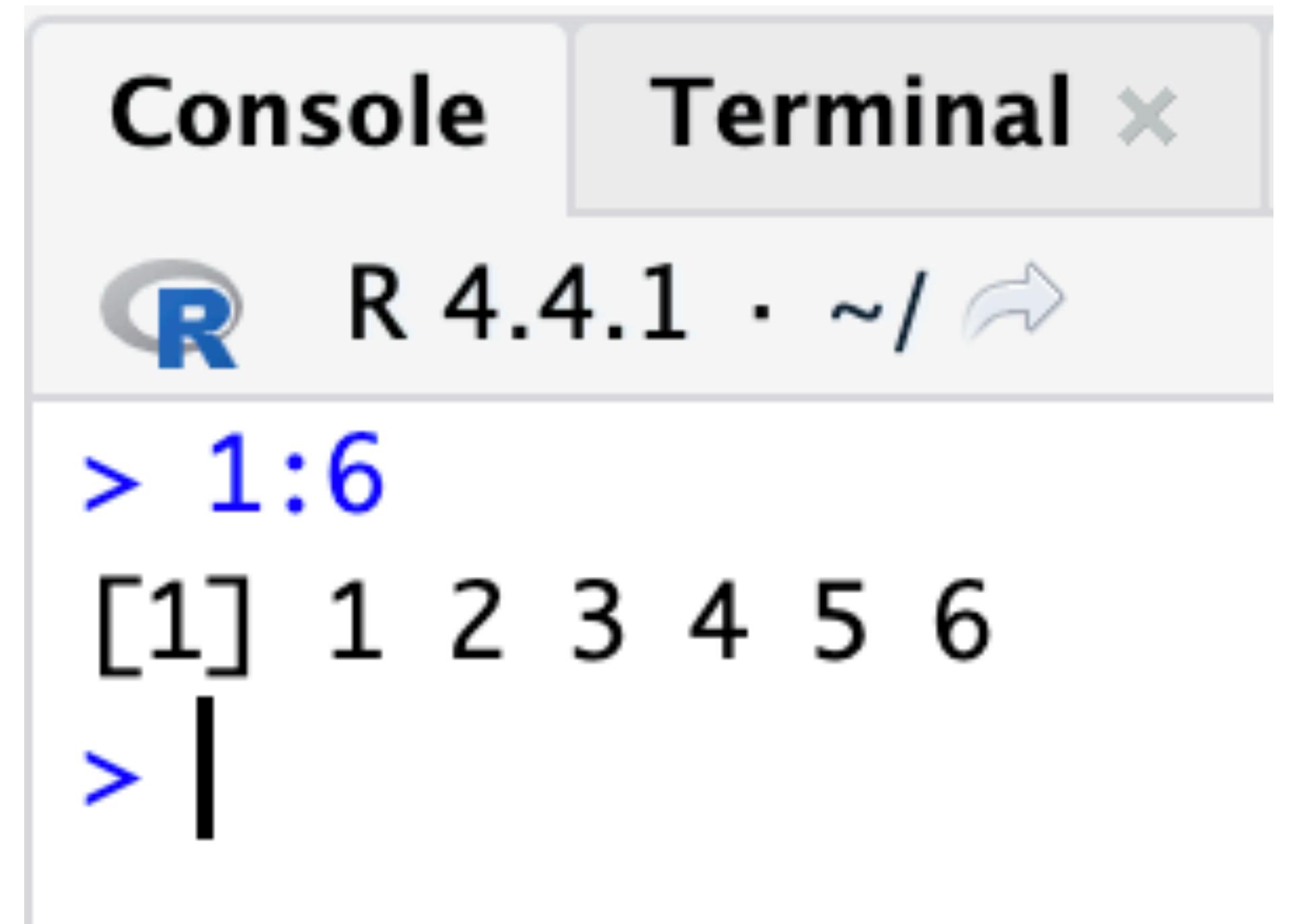
- Let's create a virtual die
- The `:` operator returns its results as a vector, a one-dimensional set of numbers.



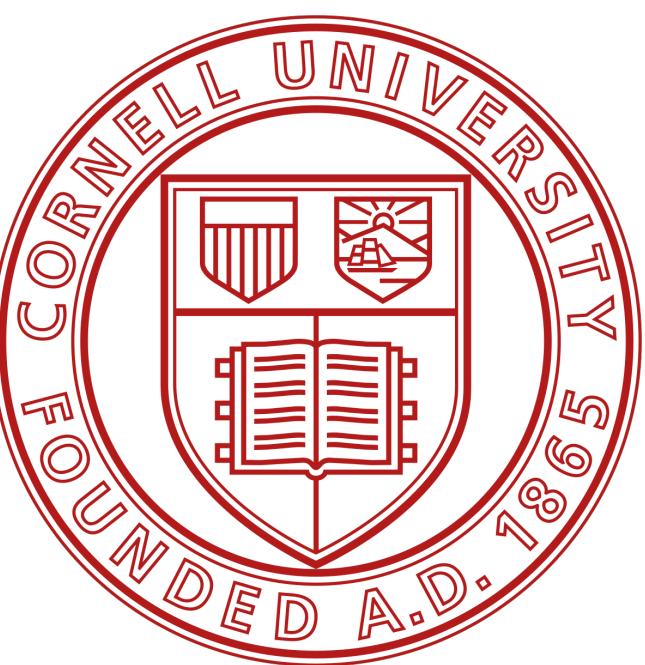


Basics

Vector

A screenshot of the RStudio interface showing a console window. The title bar includes tabs for "Console" and "Terminal". The main area displays R code and its output. The R logo icon is visible on the left. The output shows the creation of a vector from 1 to 6.

```
R 4.4.1 · ~/ ↗
> 1:6
[1] 1 2 3 4 5 6
> |
```



Basics

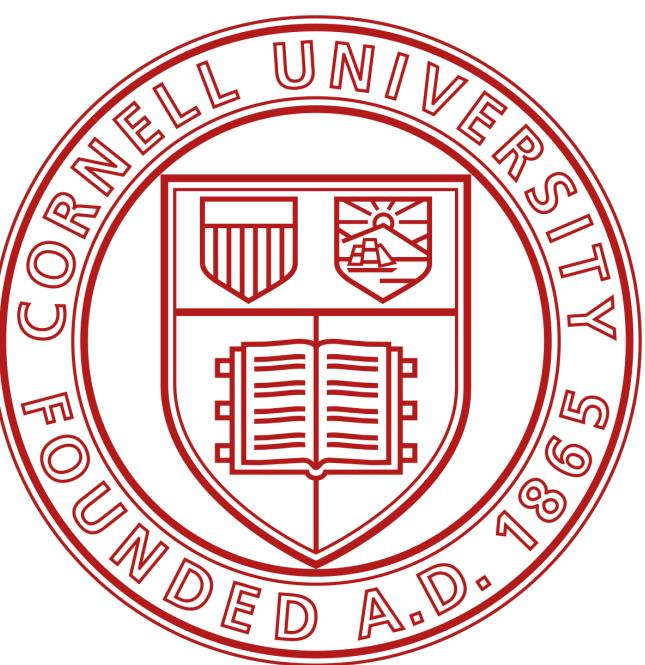
Vector

- 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.

A screenshot of the RStudio interface. The title bar shows "Console" and "Terminal". The "Console" tab is active, showing the R logo and "R 4.4.1 · ~/". Below the title bar, the console output is displayed:

```
> 1:6
[1] 1 2 3 4 5 6
> |
```

The "Terminal" tab is visible but inactive.



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.

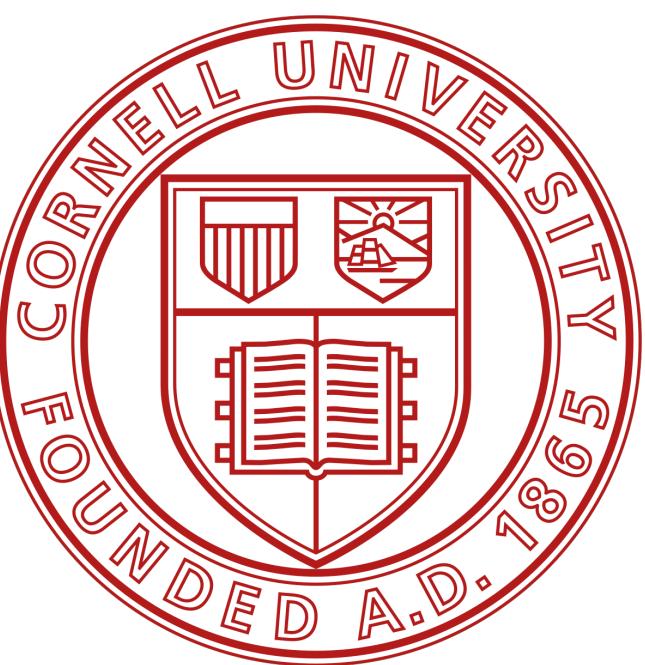
A screenshot of the RStudio interface. The title bar shows "Console" and "Terminal". The "Console" tab is active, showing the R logo and the text "R 4.4.1 · ~/". Below that, the command "`> 1:6`" is entered, followed by the output "[1] 1 2 3 4 5 6". A cursor is visible at the bottom of the console window.

```
R 4.4.1 · ~/
```

```
> 1:6
```

```
[1] 1 2 3 4 5 6
```

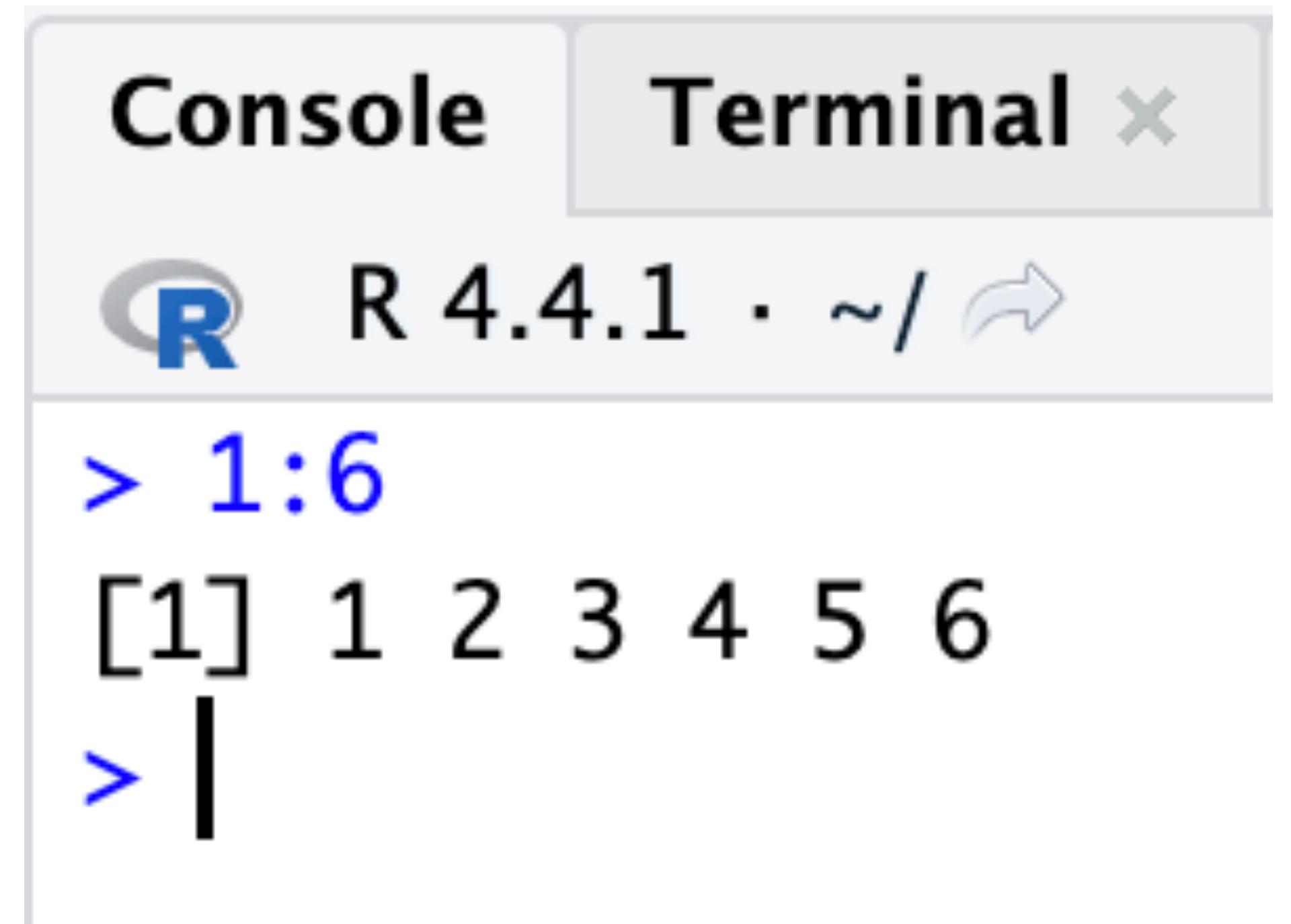
```
>
```



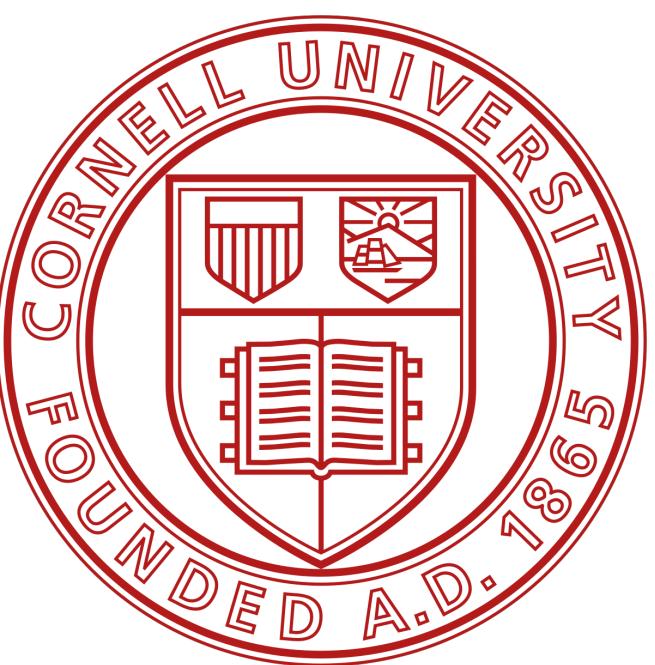
Basics

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.



The screenshot shows the RStudio interface with the 'Console' tab selected. The R logo icon is visible next to the text 'R 4.4.1 · ~/'. Below, the command `> 1:6` is entered, followed by its output: `[1] 1 2 3 4 5 6`. A blue cursor arrow is positioned below the output line.



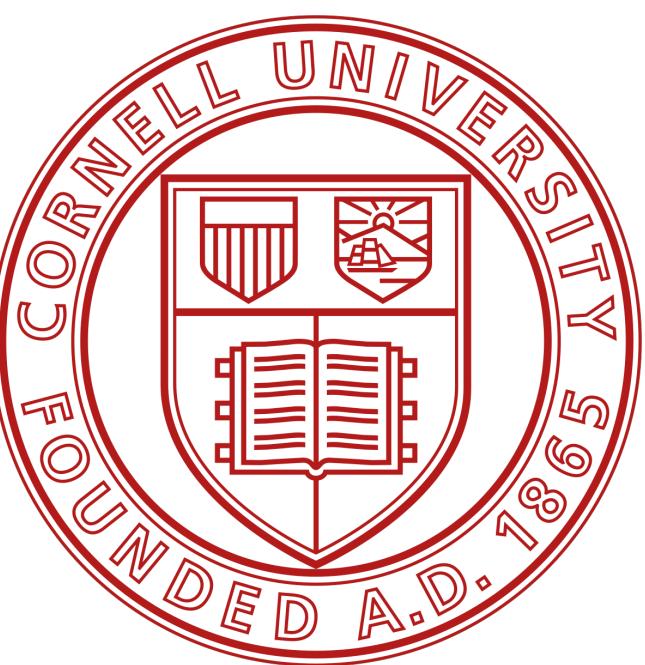
Basics

Storing data

Console Terminal ×

R 4.4.1 · ~/ ↻

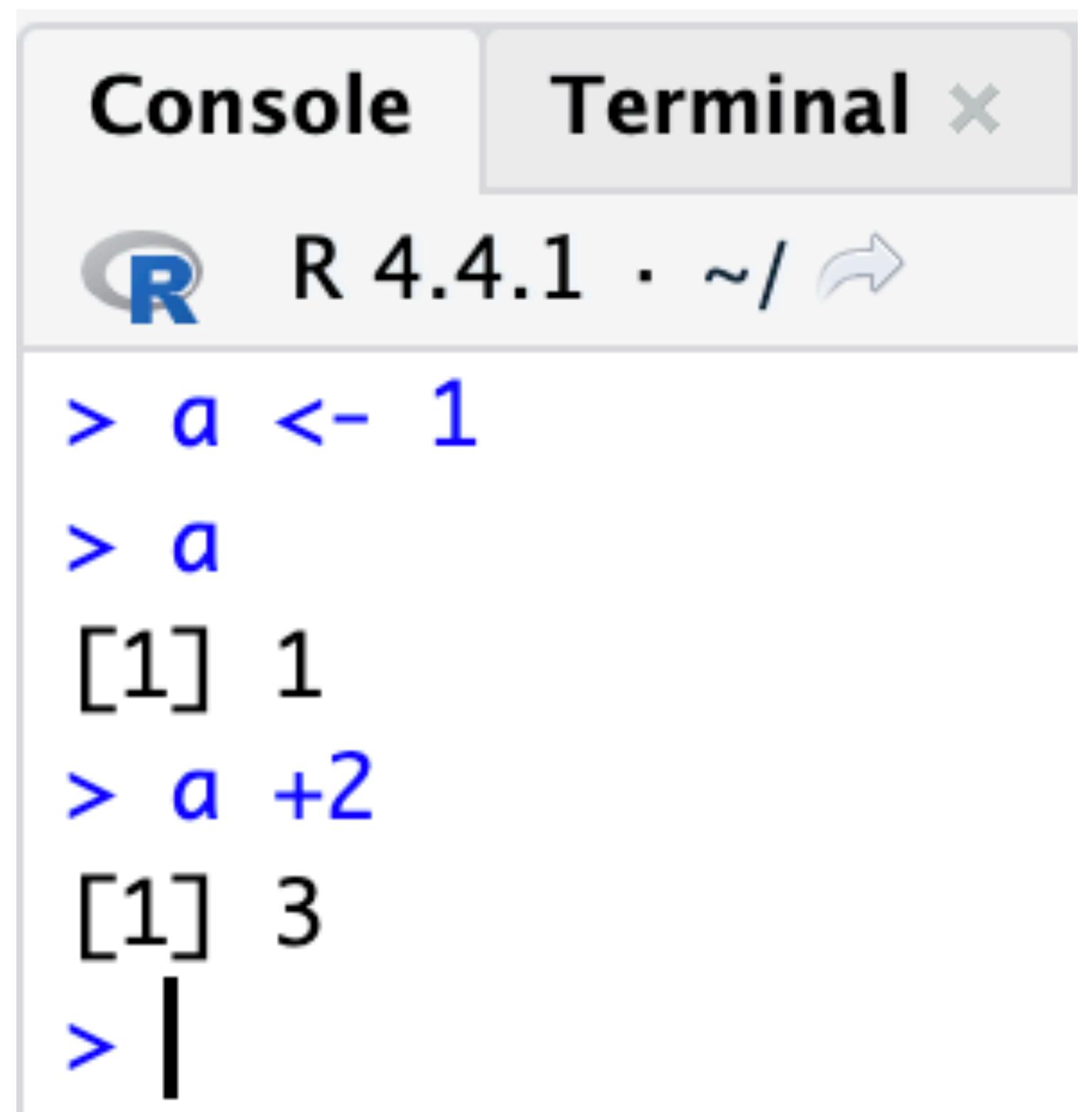
```
> a <- 1
> a
[1] 1
> a +2
[1] 3
> |
```



Basics

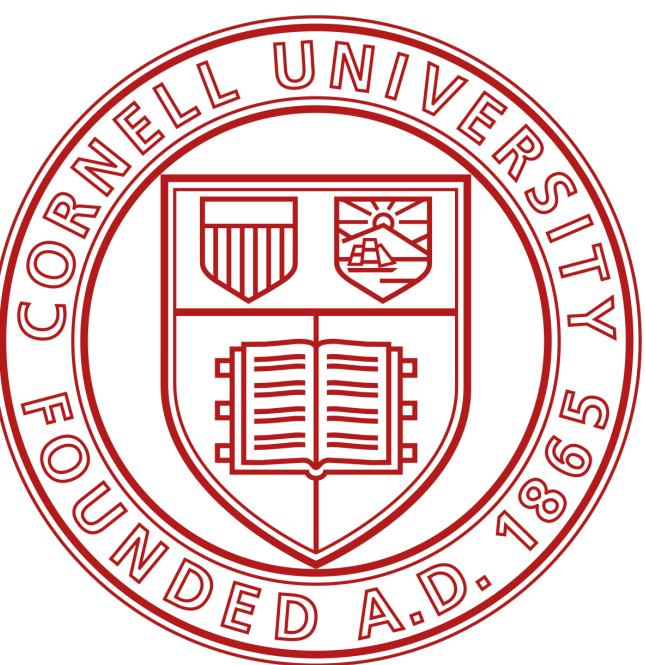
Storing data

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



A screenshot of an R console window. The title bar says "Console Terminal x". The R logo icon is on the left, followed by "R 4.4.1 · ~/". The console area contains the following R session:

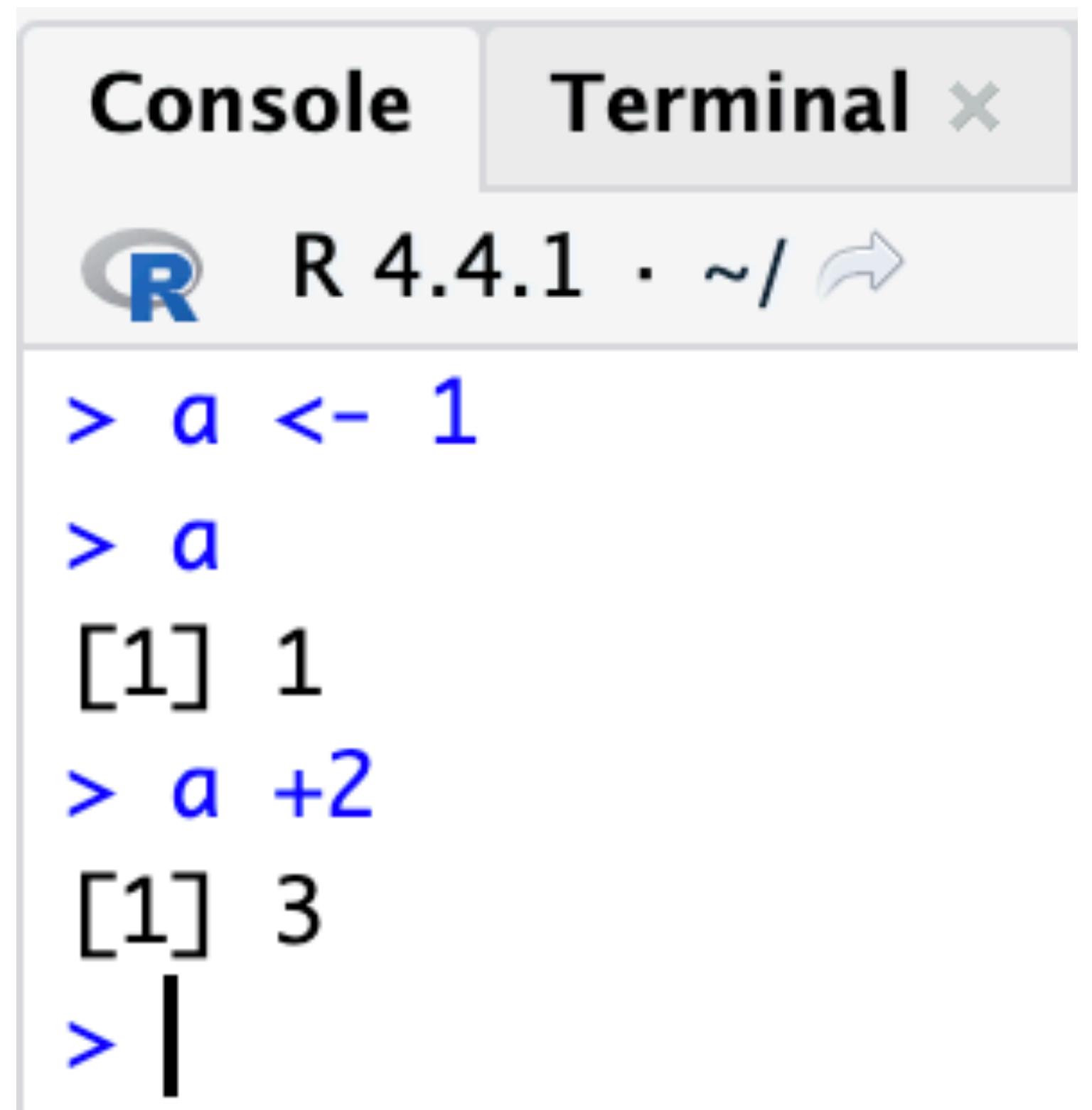
```
> a <- 1
> a
[1] 1
> a +2
[1] 3
> |
```



Basics

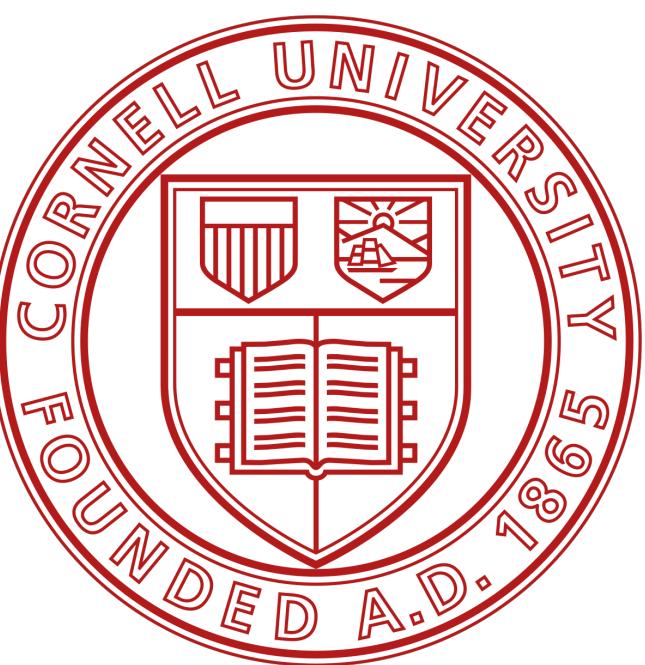
Storing data

- 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.



A screenshot of an R console window. The title bar says "Console Terminal x". The R logo icon is on the left, followed by "R 4.4.1 · ~/". The console area contains the following R session:

```
> a <- 1
> a
[1] 1
> a +2
[1] 3
> |
```



Basics

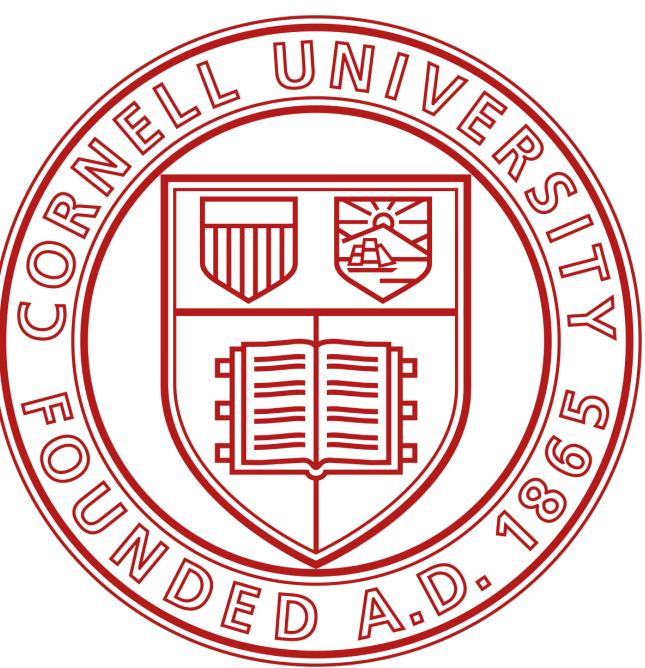
Storing data

- 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.

A screenshot of an R console window. The title bar says "Console Terminal x". The R logo icon is on the left. The text area shows the following session:

```
R 4.4.1 · ~/ ↗
> a <- 1
> a
[1] 1
> a +2
[1] 3
> |
```

The text is in a monospaced font, with command inputs in blue and outputs in black. The cursor is at the end of the last line.

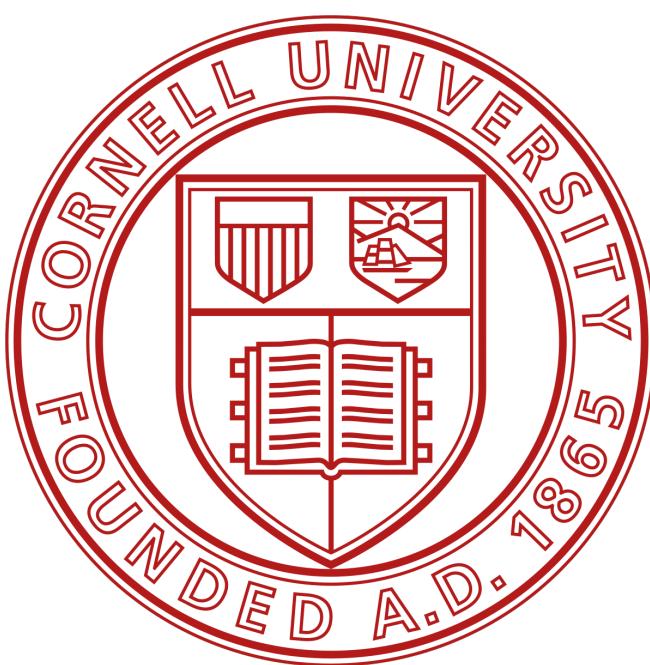


Running R code

Create an object

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

```
Console Terminal ×  
R 4.4.1 · ~/ ↗  
> die <- 1:6  
> die  
[1] 1 2 3 4 5 6  
> |
```



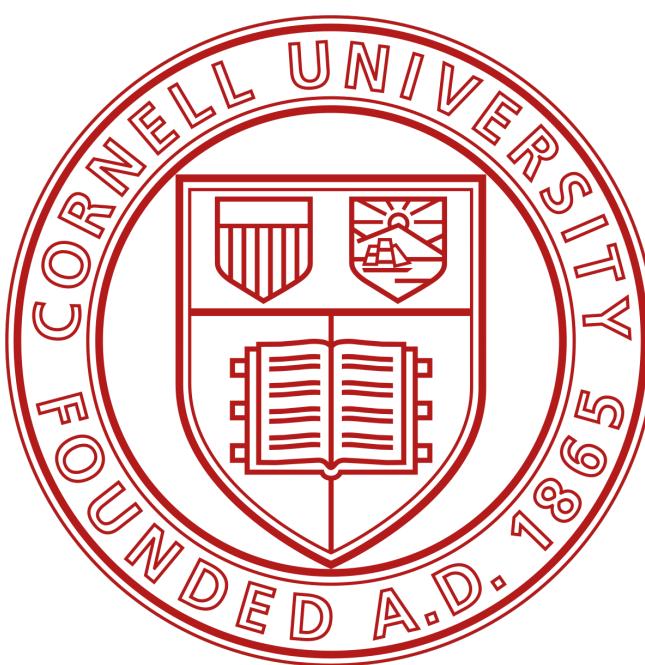
Running R code

Environment pane

A screenshot of the RStudio interface showing the Environment pane. The tab bar at the top has three tabs: "Environment" (which is selected), "History", and "Connections". Below the tabs are several icons: a folder with a green arrow, a blue document icon, a grid icon with a green arrow, a dropdown menu labeled "Import Dataset", a memory usage indicator showing "252 MiB", and a broom icon. Underneath these are dropdown menus for "R" and "Global Environment", both currently set to "Global Environment". The main area is titled "Values" and shows a single entry: "die" of type "int [1:6]" containing the values 1, 2, 3, 4, 5, 6.

A screenshot of the RStudio interface showing the Console pane. The tab bar at the top has two tabs: "Console" (selected) and "Terminal". Below the tabs is the R logo and the text "R 4.4.1 · ~/". The console window displays the following R session:

```
> ls()
[1] "die"          "my_number"
>
```



Running R code

Environment pane

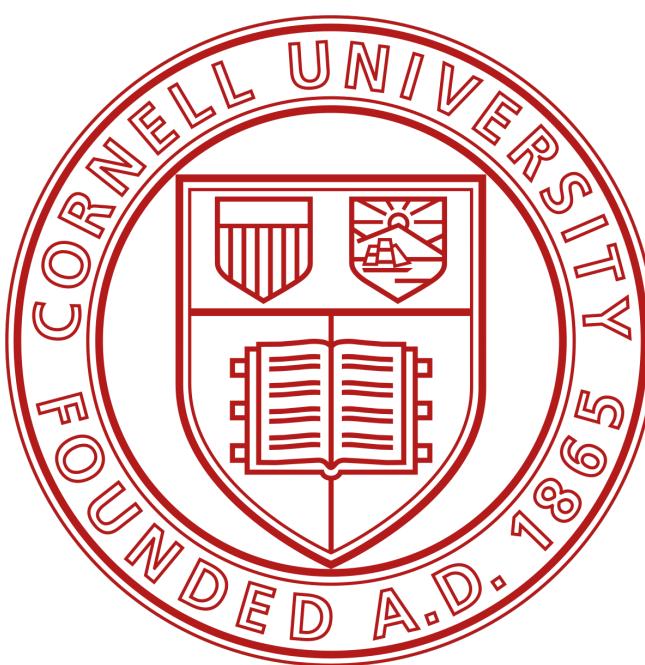
- When you create an object, the object will appear in the environment pane of RStudio.

A screenshot of the RStudio interface focusing on the Environment pane. The pane has tabs for Environment, History, and Connections, with Environment selected. It shows a file icon, a global environment icon, and a memory usage indicator of 252 MiB. Below these are sections for 'Values' and 'Global Environment'. In the 'Values' section, there is a row for 'die' which is defined as an integer vector from 1 to 6.

die	int [1:6] 1 2 3 4 5 6
-----	-----------------------

A screenshot of the RStudio interface focusing on the Console pane. The pane has tabs for Console and Terminal, with Console selected. It shows the R logo and the text "R 4.4.1 · ~/". Below this, the command "> ls()" is entered, followed by the output "[1] "die" "my_number"".

```
R 4.4.1 · ~/  
> ls()  
[1] "die"           "my_number"  
>
```



Running R code

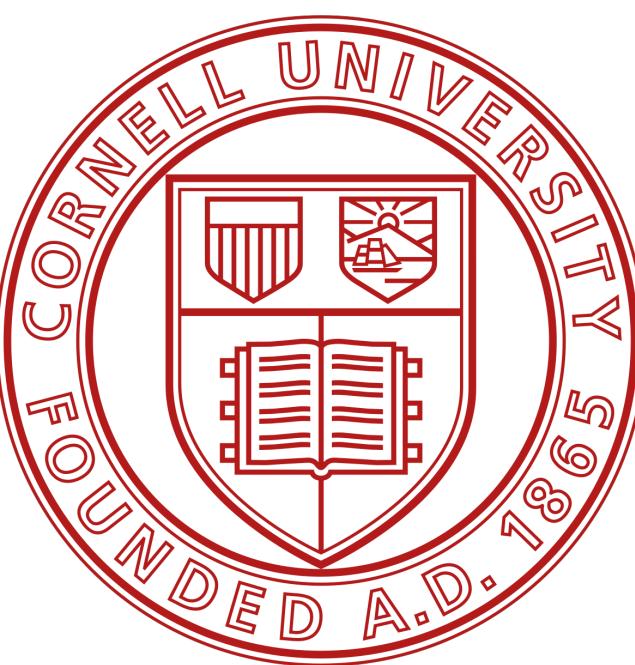
Environment pane

- 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.

The screenshot shows the RStudio interface. The top navigation bar has tabs for "Environment", "History", and "Connections". The "Environment" tab is active. Below it, there are icons for file operations like "Import Dataset" and "252 MiB". The "Global Environment" dropdown is set to "R". The "Values" section lists an object named "die" which is an integer vector from 1 to 6. The bottom part of the image shows the "Console" tab, which displays the R command "ls()", its output "[1] "die"" and "[1] "my_number"" in blue, and the prompt ">".

```
Environment History Connections
Import Dataset 252 MiB
Global Environment
Values
die int [1:6] 1 2 3 4 5 6

Console Terminal ×
R 4.4.1 · ~/ ↗
> ls()
[1] "die"           "my_number"
>
```



Running R code

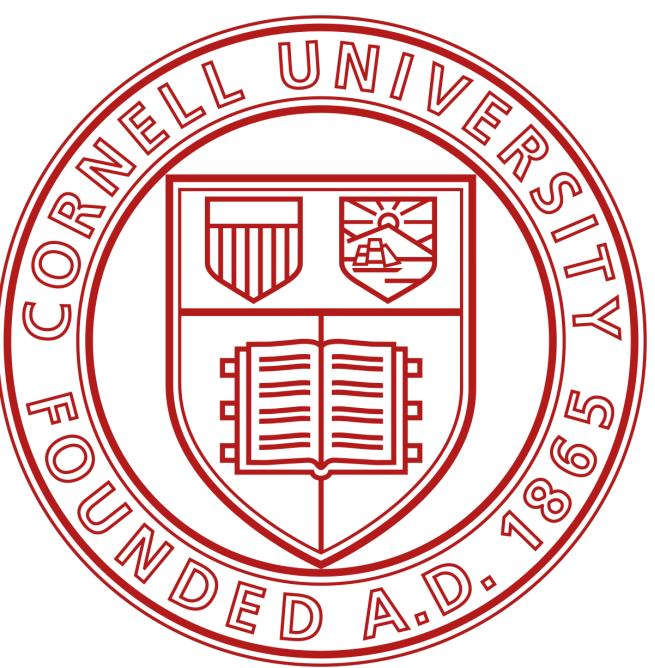
Environment pane

- 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`

The screenshot shows the RStudio interface. The top navigation bar has tabs for "Environment", "History", and "Connections". The "Environment" tab is active. Below it, there are buttons for "File", "Import Dataset", "Memory Usage" (252 MiB), and "Global Environment". The "Global Environment" dropdown is open, showing the object "die". The "Values" section below shows the object "die" with the value "int [1:6] 1 2 3 4 5 6".

The bottom part of the image shows the "Console" tab, which displays the R session:

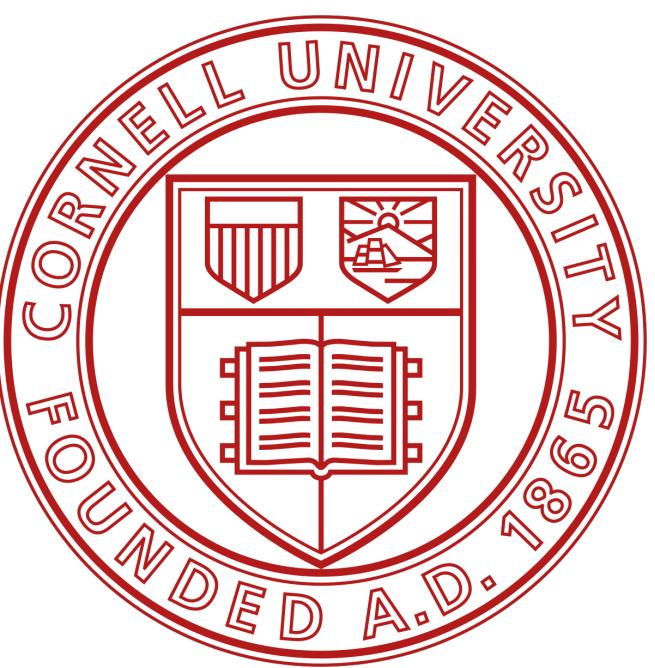
```
R 4.4.1 · ~/ 
> ls()
[1] "die"           "my_number"
```



Running R code

Name an object

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



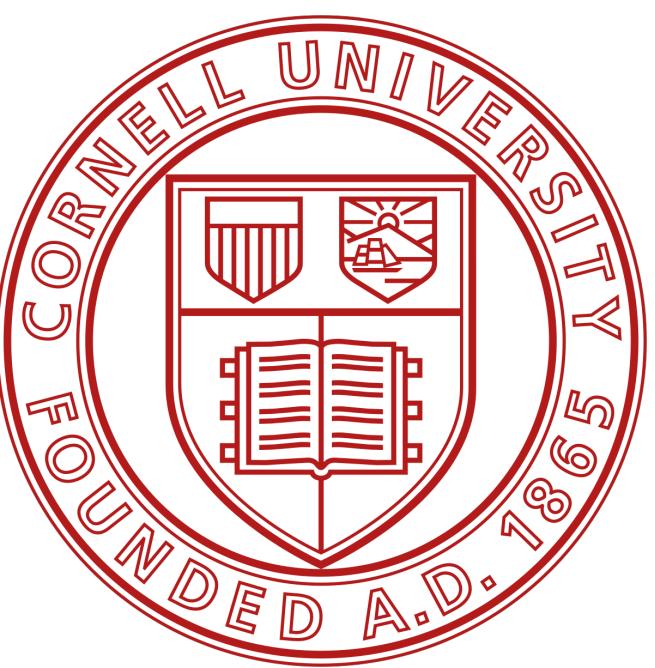
Running R code

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



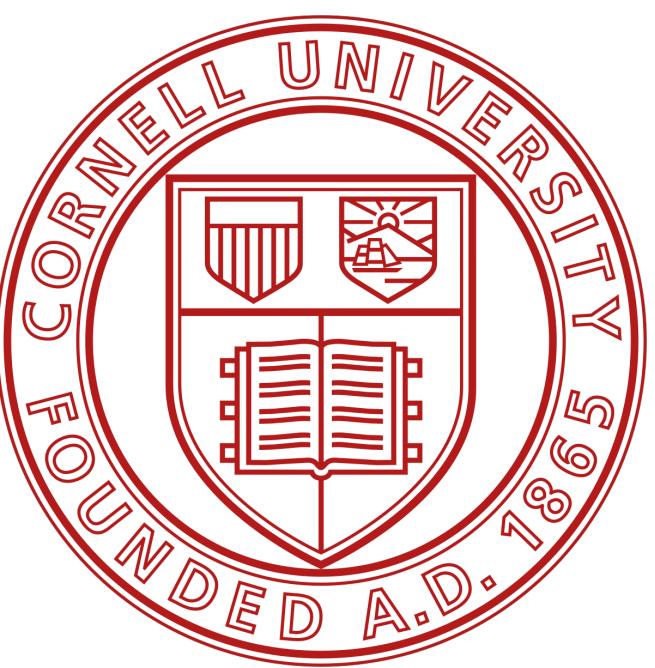
Running R code

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

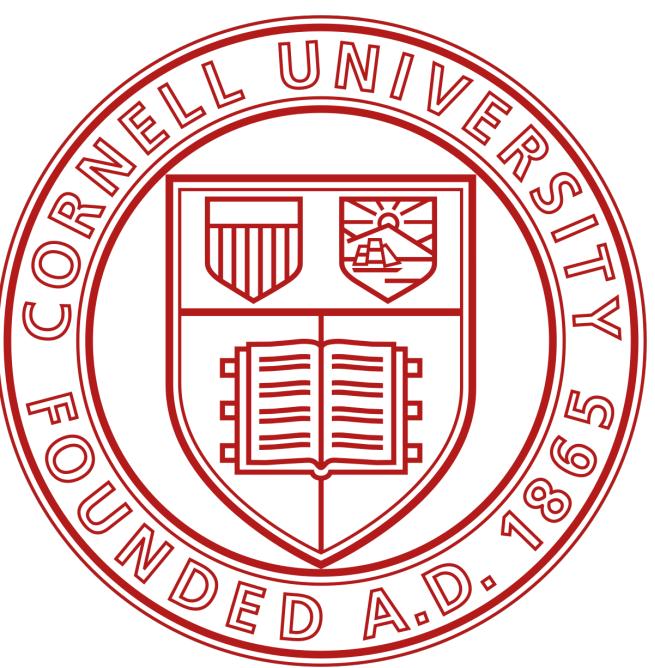


Running R code

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

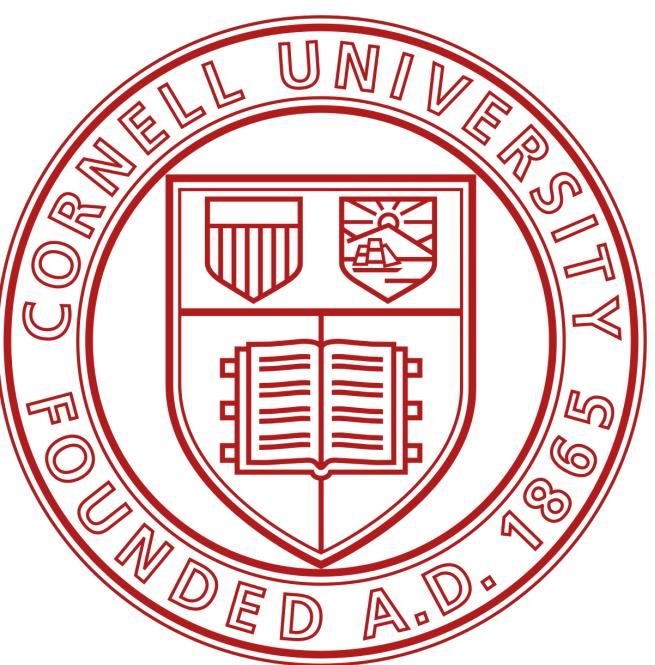


Running R code

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

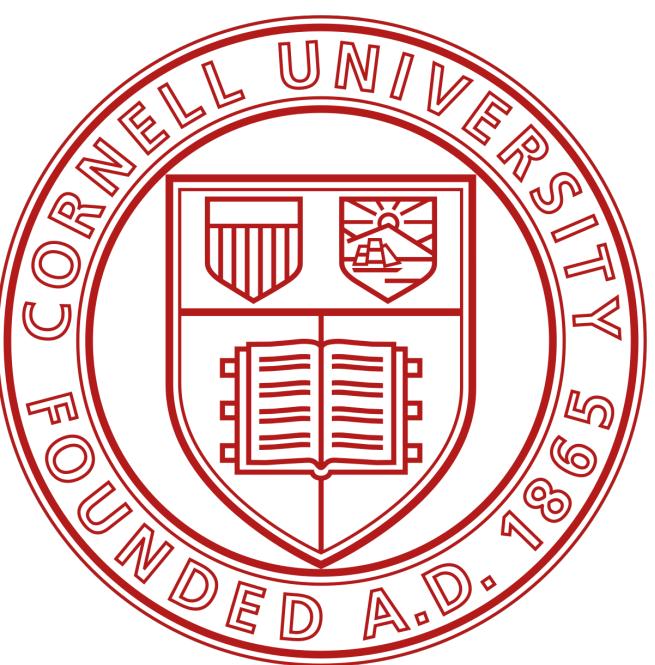
Good names	Names that cause errors
a	1trial
b	\$
FOO	^mean
my_var	2nd
.day	!bad



Running R code

Name overwriting

```
Console Terminal ×
R 4.4.1 · ~/ ↗
> my_number <- 1
> my_number
[1] 1
> my_number <- 999
> my_number
[1] 999
> |
```

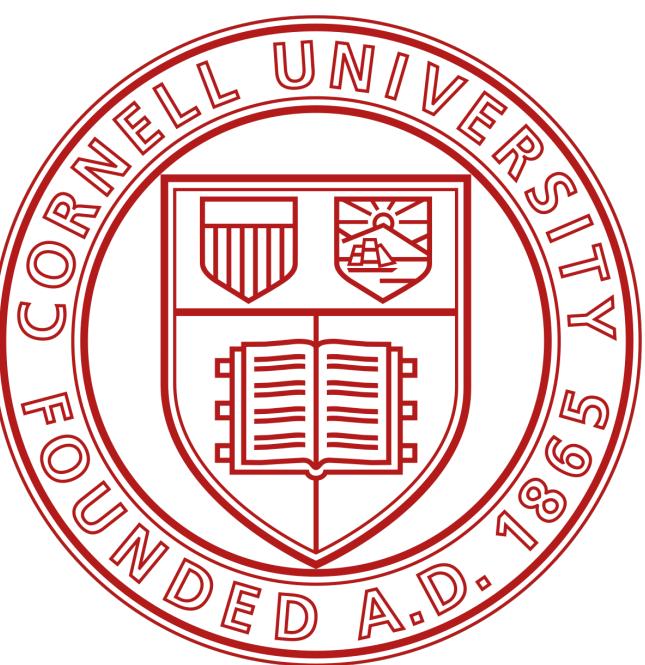


Running R code

Name overwriting

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

```
Console Terminal ×
R 4.4.1 · ~/ ↗
> my_number <- 1
> my_number
[1] 1
> my_number <- 999
> my_number
[1] 999
> |
```

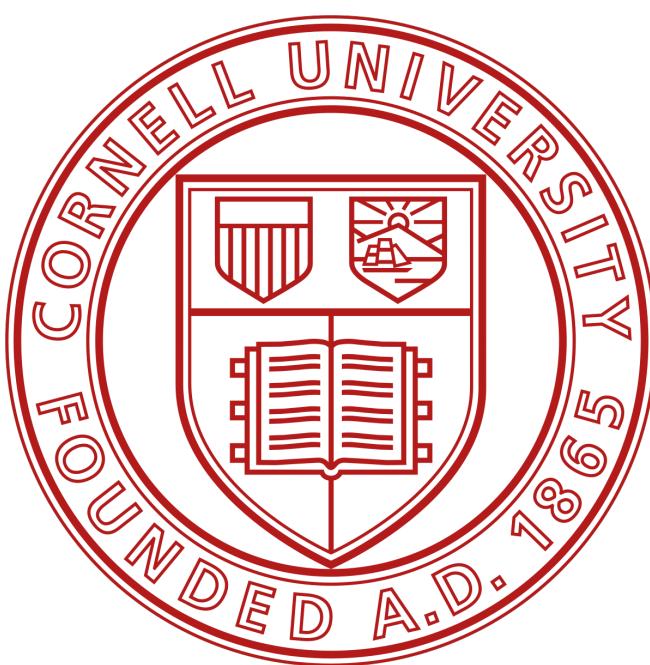


Running R code

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.

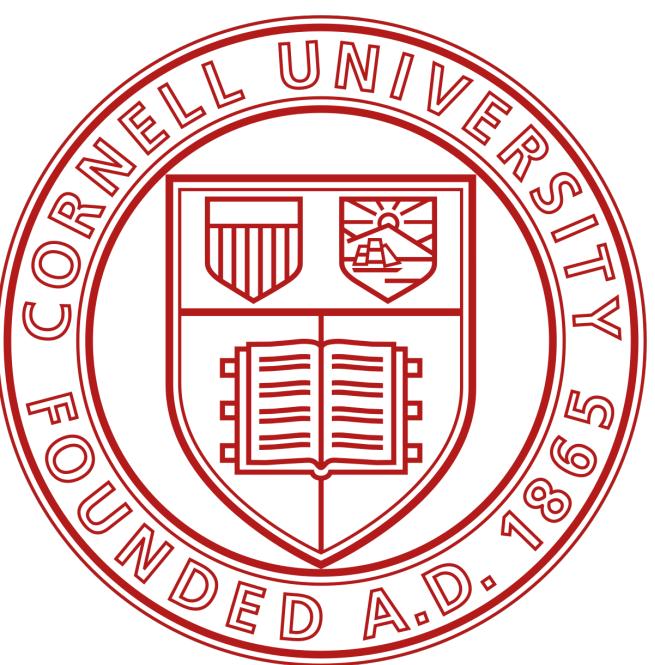
```
Console Terminal ×
R 4.4.1 · ~/ ↗
> my_number <- 1
> my_number
[1] 1
> my_number <- 999
> my_number
[1] 999
> |
```



Running R code

Element-wise execution

```
Console Terminal ×
R 4.4.1 · ~/ ↗
> die
[1] 1 2 3 4 5 6
> die - 1
[1] 0 1 2 3 4 5
> die / 2
[1] 0.5 1.0 1.5 2.0 2.5 3.0
> die * die
[1] 1 4 9 16 25 36
>
```

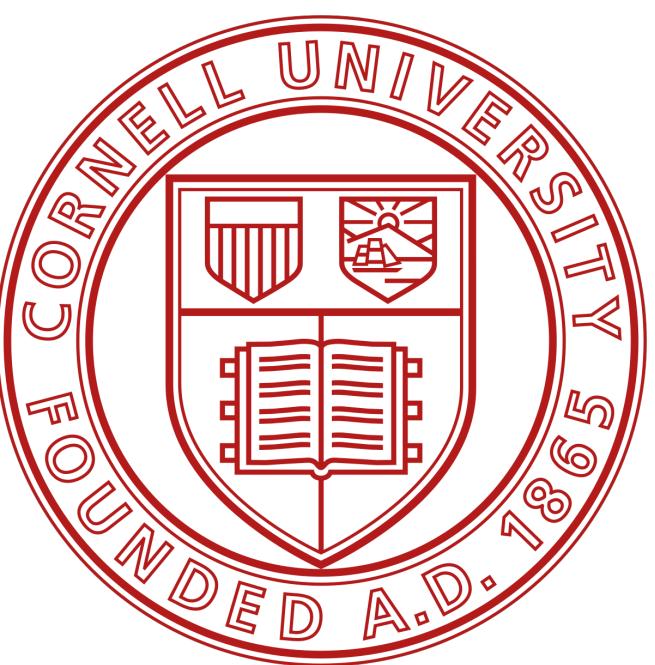


Running R code

Element-wise execution

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

```
Console Terminal ×
R 4.4.1 · ~/ ↗
> die
[1] 1 2 3 4 5 6
> die - 1
[1] 0 1 2 3 4 5
> die / 2
[1] 0.5 1.0 1.5 2.0 2.5 3.0
> die * die
[1] 1 4 9 16 25 36
>
```

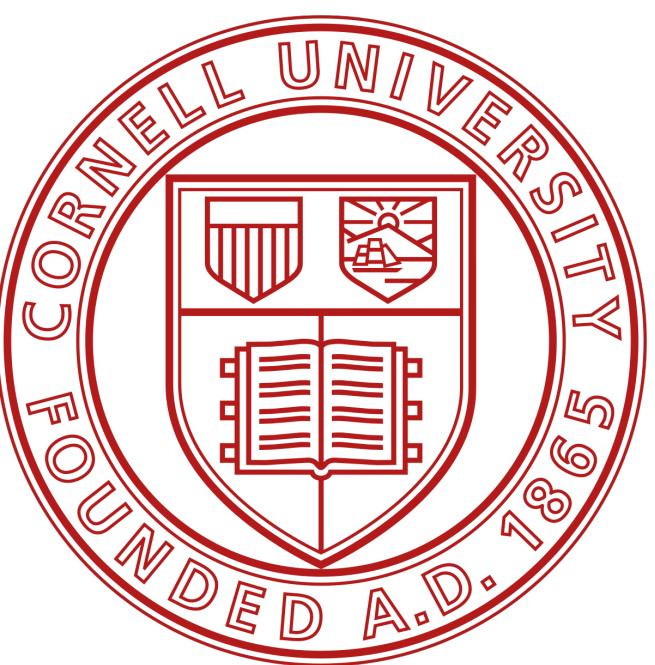


Running R code

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.

```
Console Terminal ×
R 4.4.1 · ~/ ↗
> die
[1] 1 2 3 4 5 6
> die - 1
[1] 0 1 2 3 4 5
> die / 2
[1] 0.5 1.0 1.5 2.0 2.5 3.0
> die * die
[1] 1 4 9 16 25 36
>
```

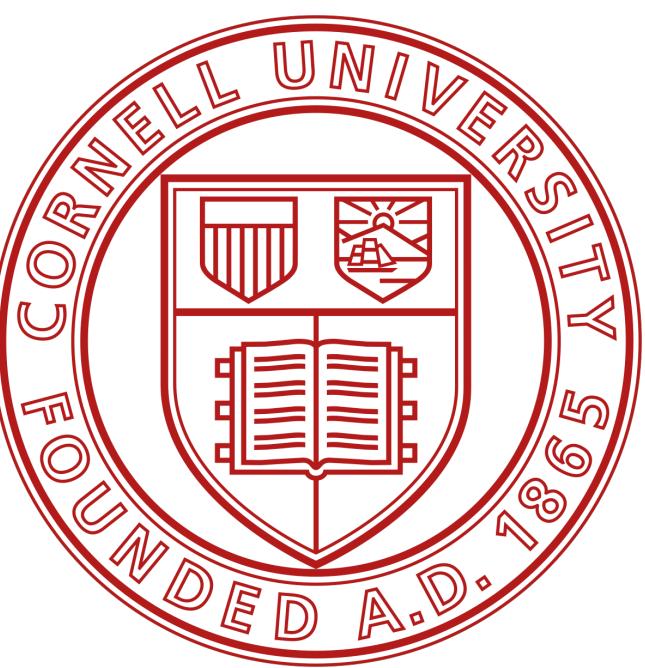


Running R code

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.

```
Console Terminal ×
R 4.4.1 · ~/ ↗
> die
[1] 1 2 3 4 5 6
> die - 1
[1] 0 1 2 3 4 5
> die / 2
[1] 0.5 1.0 1.5 2.0 2.5 3.0
> die * die
[1] 1 4 9 16 25 36
>
```

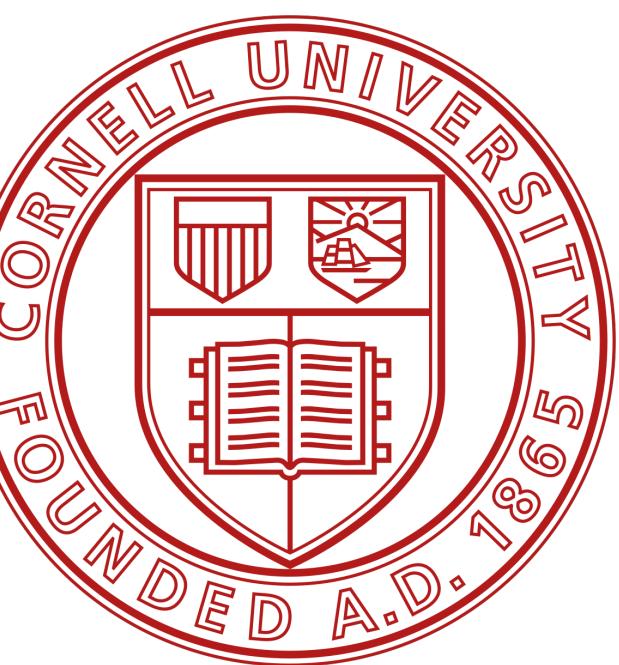


Running R code

Element-wise execution

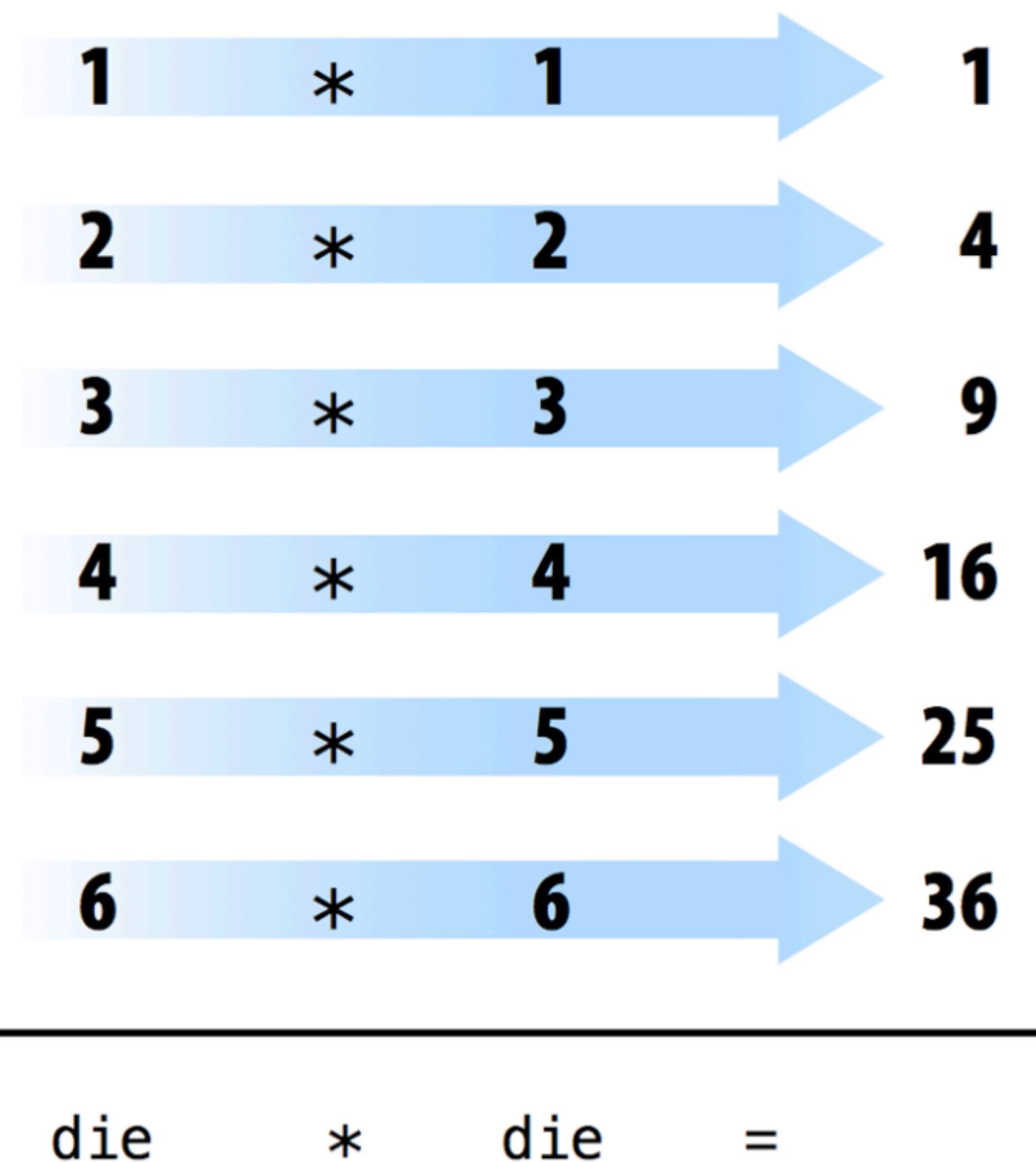
- You now have a virtual die that is stored in your computer's memory.
- You can do all sorts of math with the die.
- R does not always follow the rules of matrix multiplication.
- R uses element-wise execution.

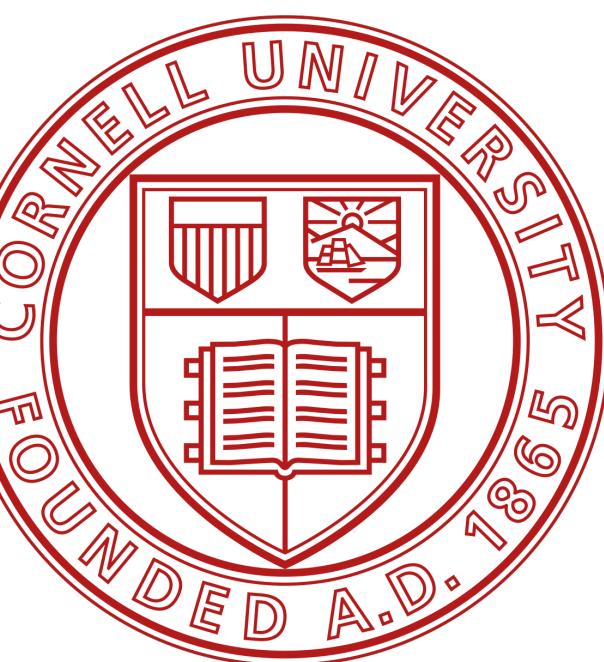
```
Console Terminal ×
R 4.4.1 · ~/ ↗
> die
[1] 1 2 3 4 5 6
> die - 1
[1] 0 1 2 3 4 5
> die / 2
[1] 0.5 1.0 1.5 2.0 2.5 3.0
> die * die
[1] 1 4 9 16 25 36
>
```



Running R code

Element-wise execution

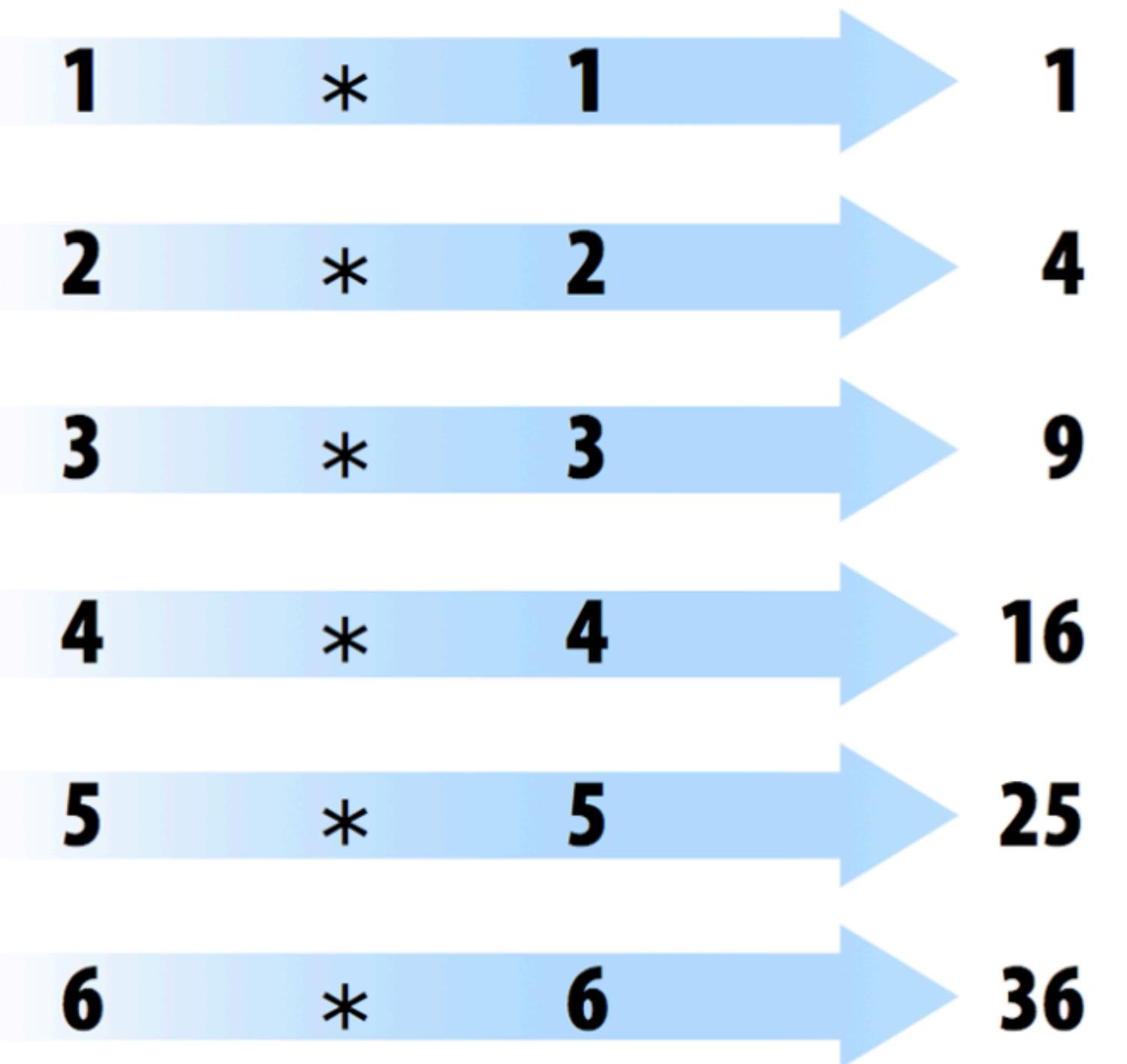




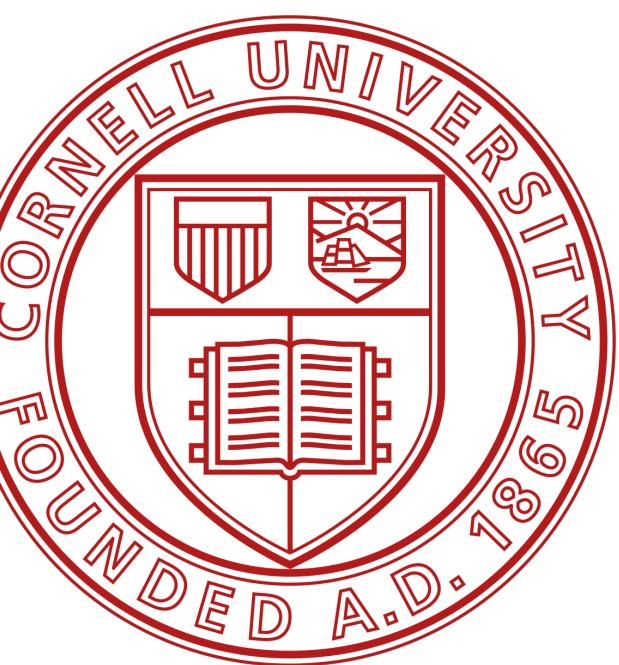
Running R code

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.



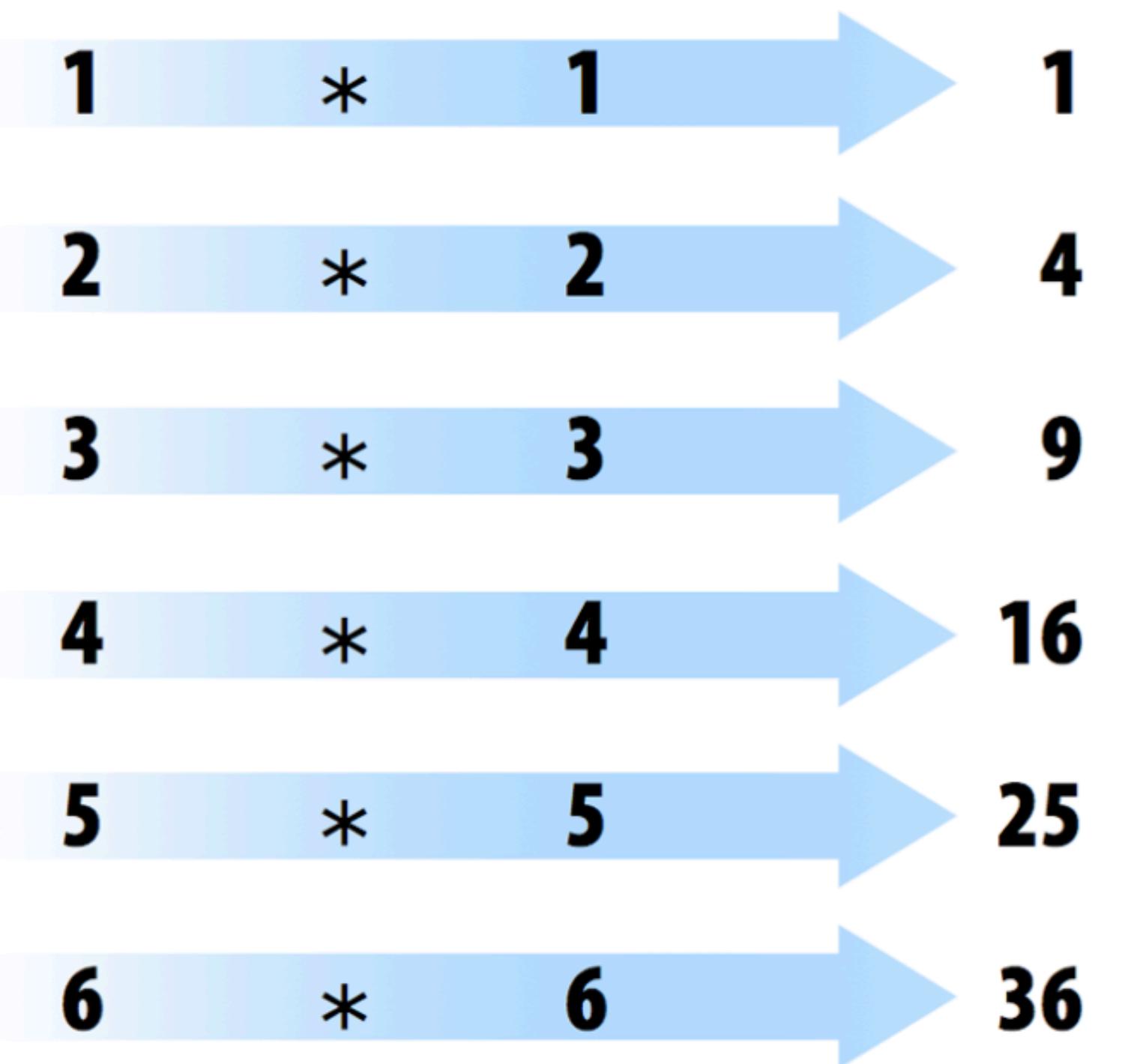
die * die =



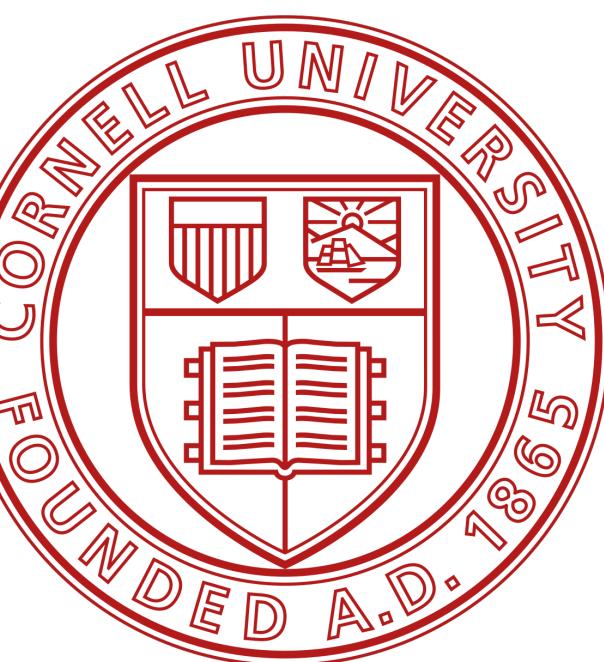
Running R code

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.
- 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.



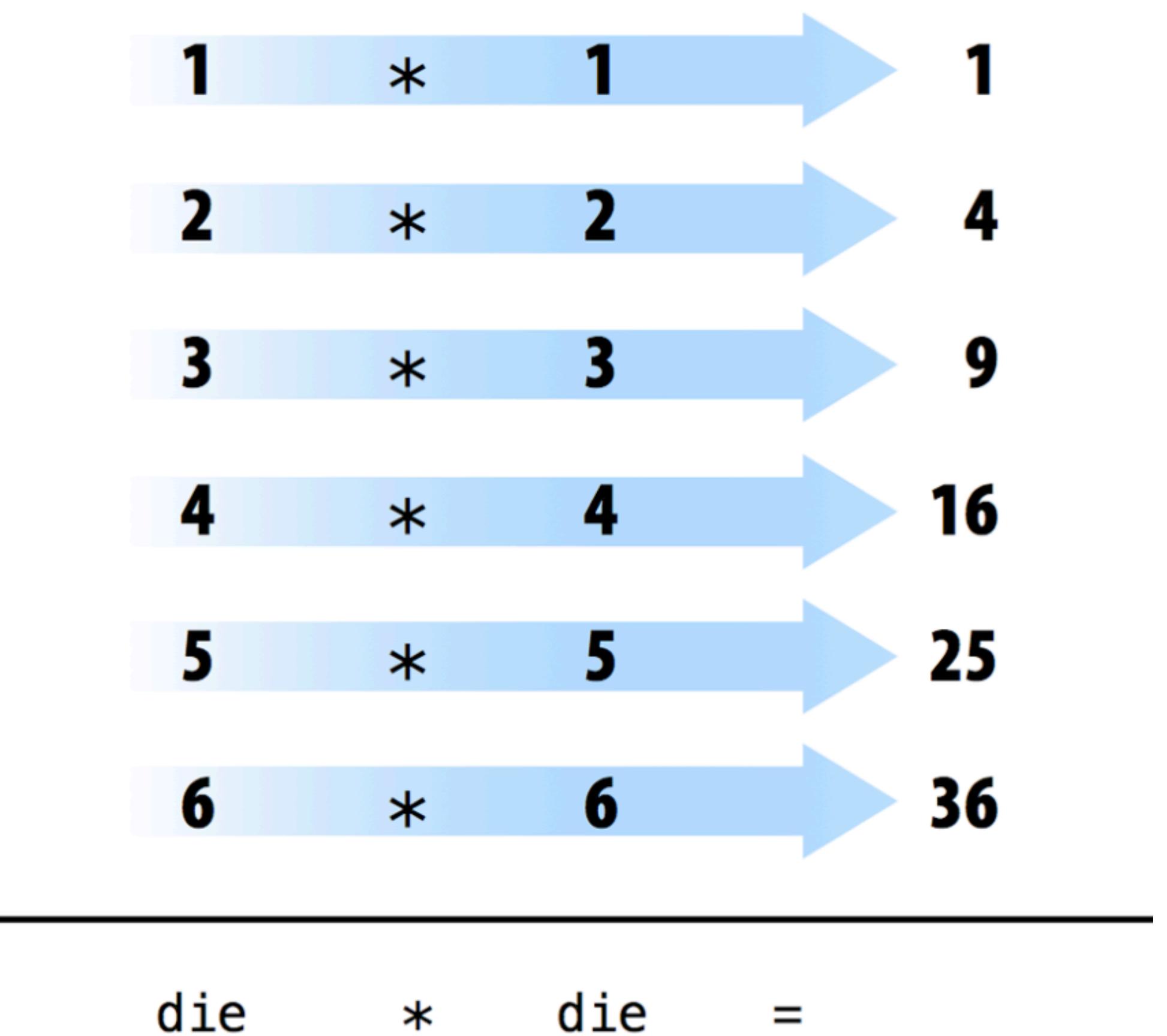
die * die =

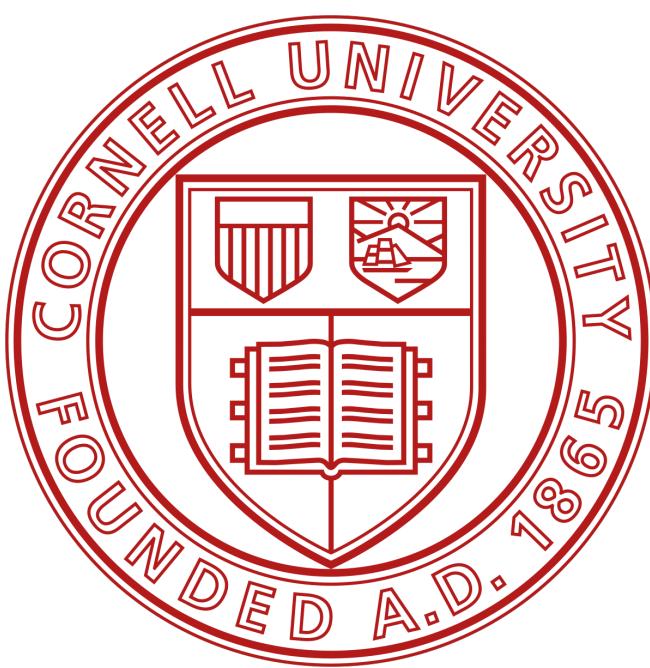


Running R code

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.
- 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.

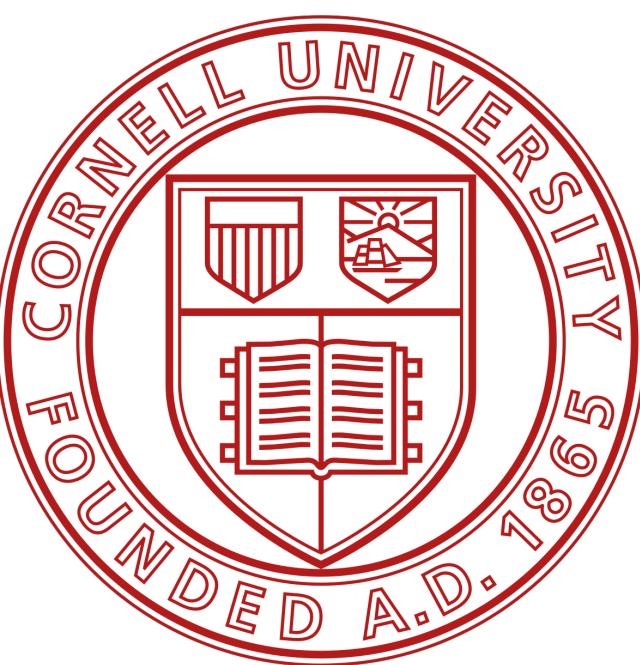




Running R code

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
```

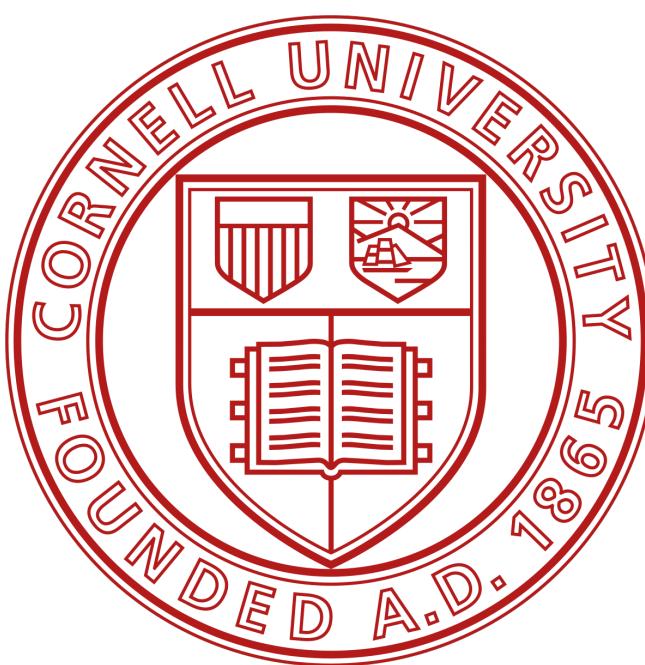


Running R code

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)

```
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
```

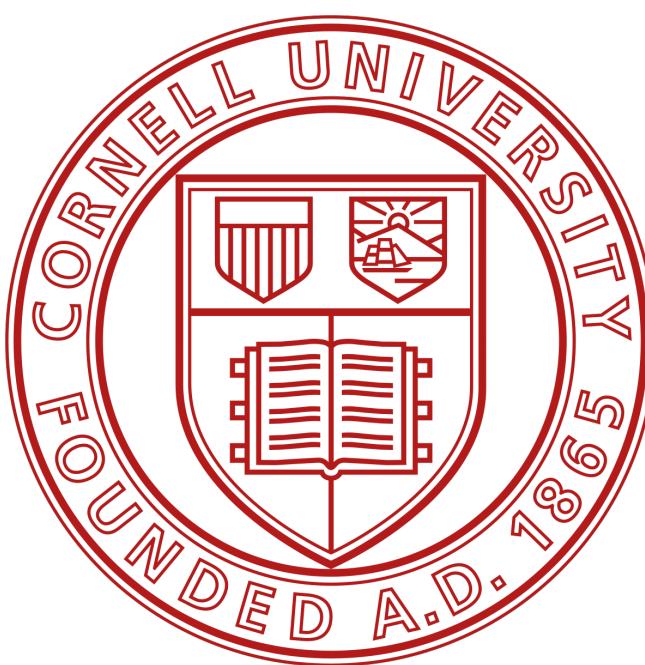


Running R code

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*)
- This isn't a permanent change, the shorter vector will be its original size after R does the math.

```
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
```

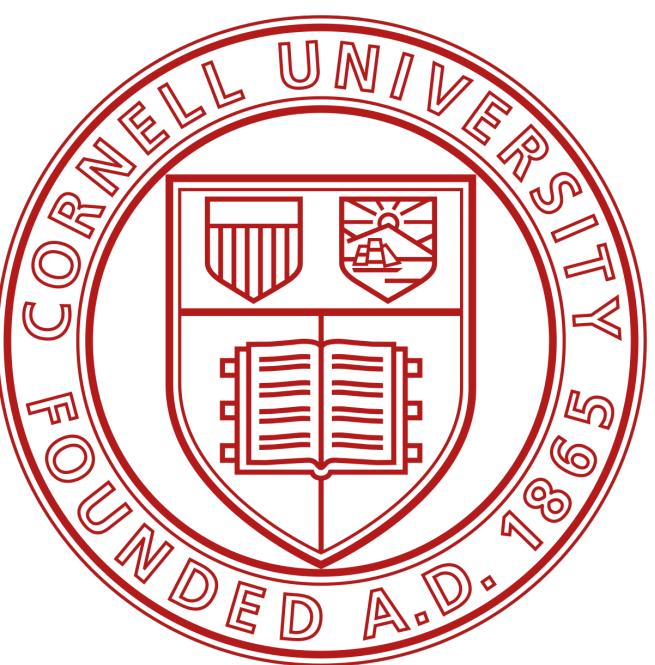


Running R code

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*)
- 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.

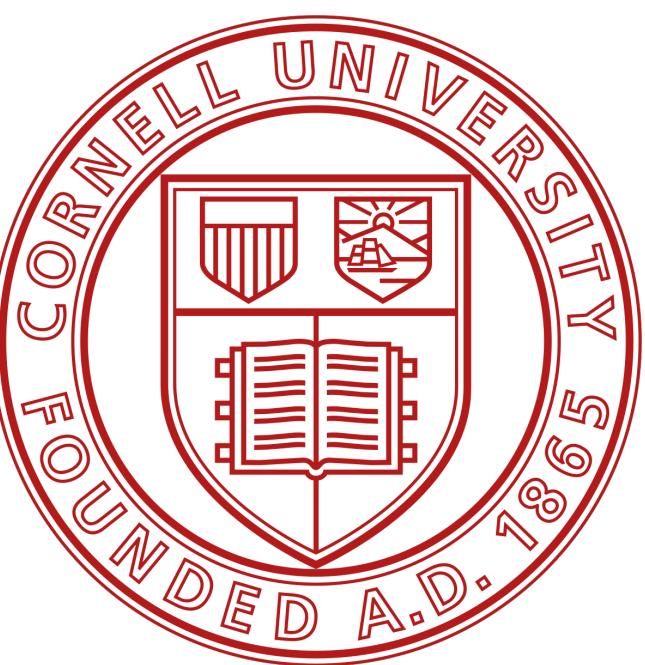
```
Console Terminal ×
R 4.4.1 · ~/ ↗
> 1:2
[1] 1 2
> 1:4
[1] 1 2 3 4
> die
[1] 1 2 3 4 5 6
> die + 1:2
[1] 2 4 4 6 6 8
> die + 1:4
[1] 2 4 6 8 6 8
Warning message:
In die + 1:4 :
  longer object length is not a multiple of shorter object length
```



Running R code

Linear algebra operations

```
Console Terminal ×
R 4.4.1 · ~/ ↗
> die %*% die
     [,1]
[1,]  91
> die %*% 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
```

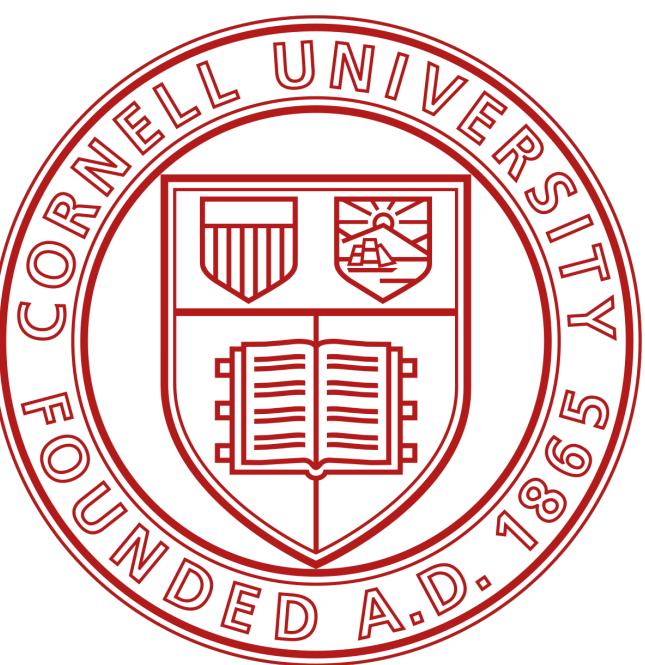


Running R code

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.

```
Console Terminal ×
R 4.4.1 · ~/ ↗
> die %*% die
[,1]
[1,] 91
> die %*% 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
```

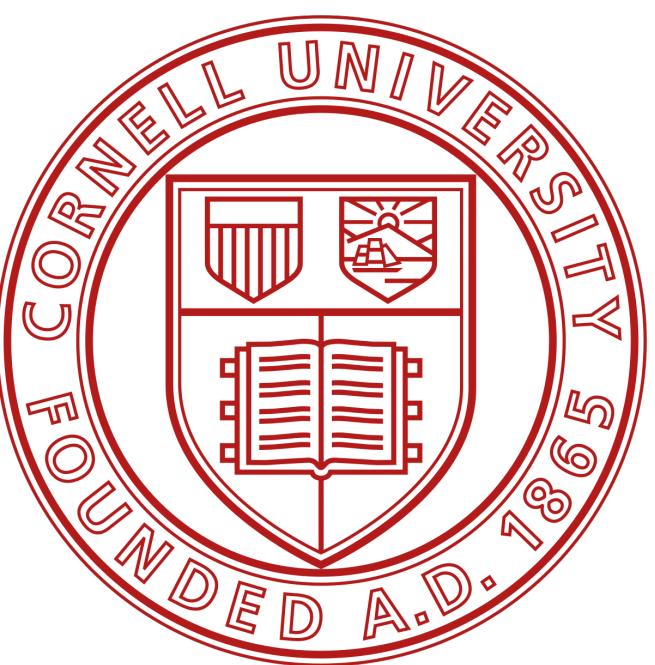


Running R code

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.
- Don't think that R has given up on traditional matrix multiplication.

```
Console Terminal ×
R 4.4.1 · ~/ ↗
> die %*% die
[,1]
[1,] 91
> die %*% 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
```

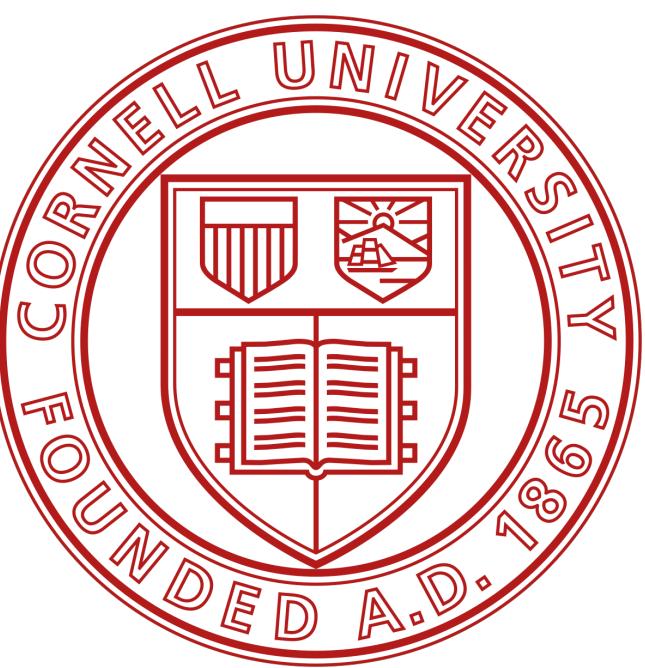


Running R code

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.
- 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.

```
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
```

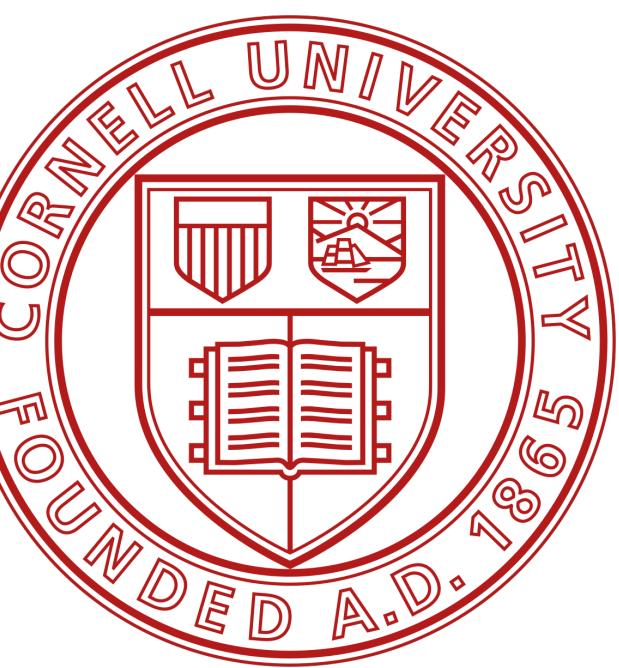


Running R code

Vector inner product

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

$\langle \cdot, \cdot \rangle$



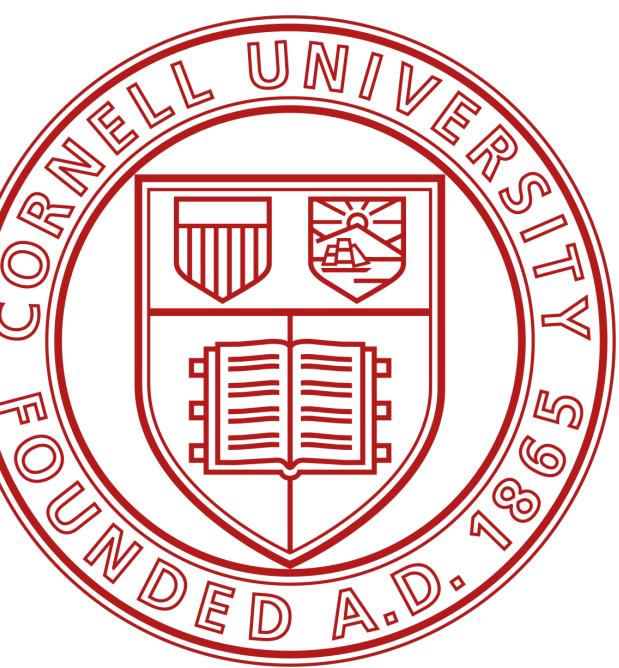
Running R code

Vector inner product

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

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

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

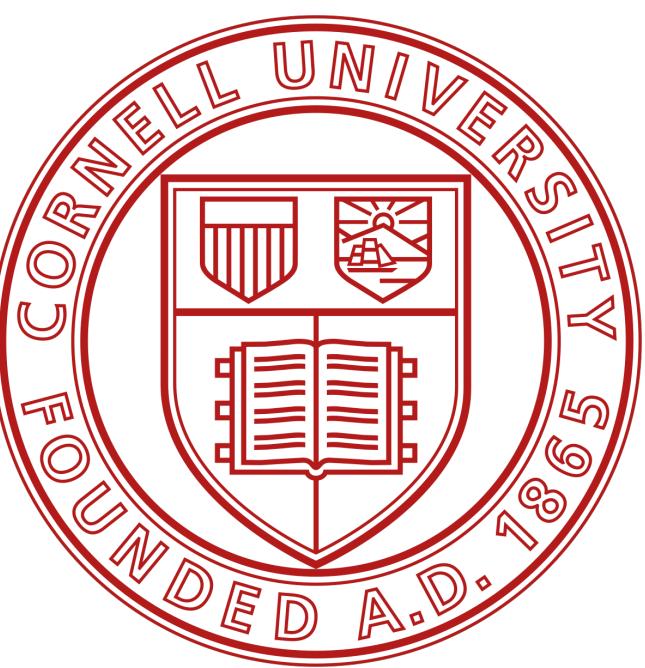


Running R code

Vector inner product

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

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



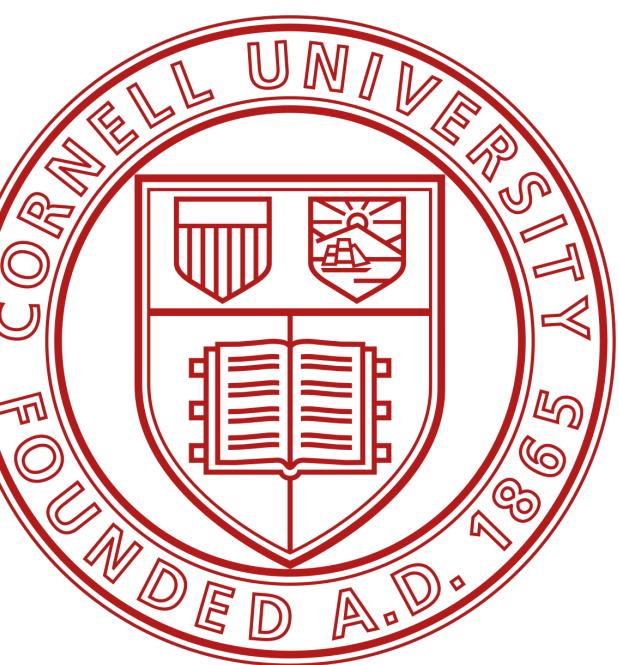
Running R code

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

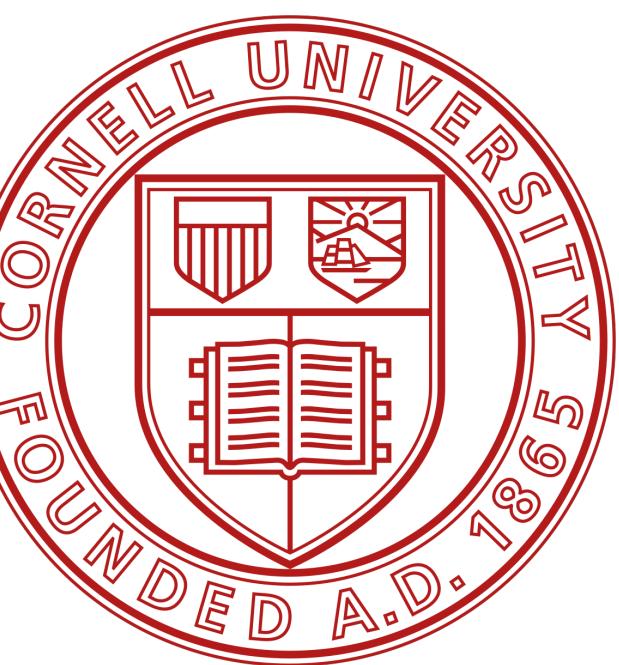
Vector outer product

- 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

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

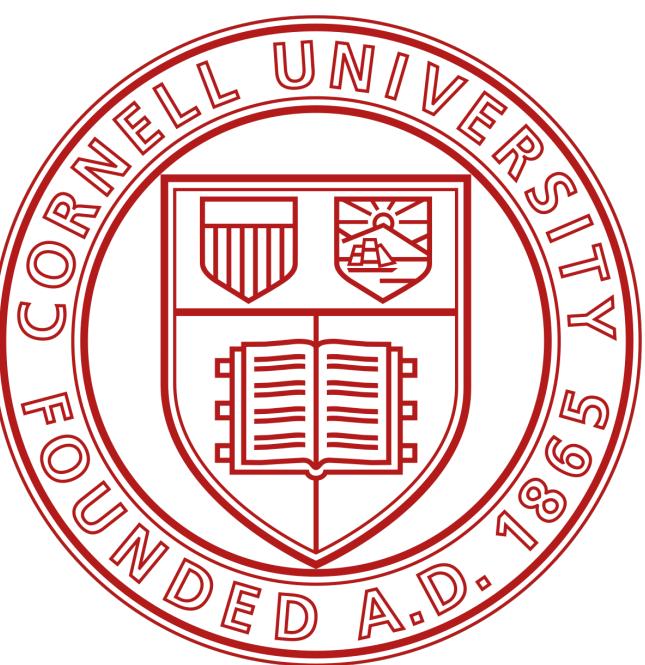
- The outer product of two coordinate vectors is the matrix whose entries are all products of an element in the first vector with an element in the second vector.
- If the two coordinate vectors have dimensions n and m , then their outer product is an $n \times m$ matrix.

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

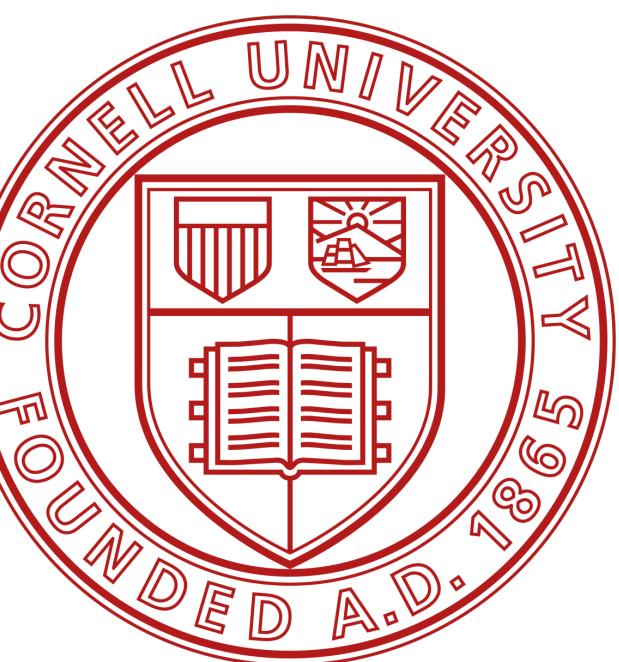
$$\mathbf{u} \otimes \mathbf{v} = \mathbf{A} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & \dots & u_1 v_n \\ u_2 v_1 & u_2 v_2 & \dots & u_2 v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_m v_1 & u_m v_2 & \dots & u_m v_n \end{bmatrix}$$

Running R code

Functions



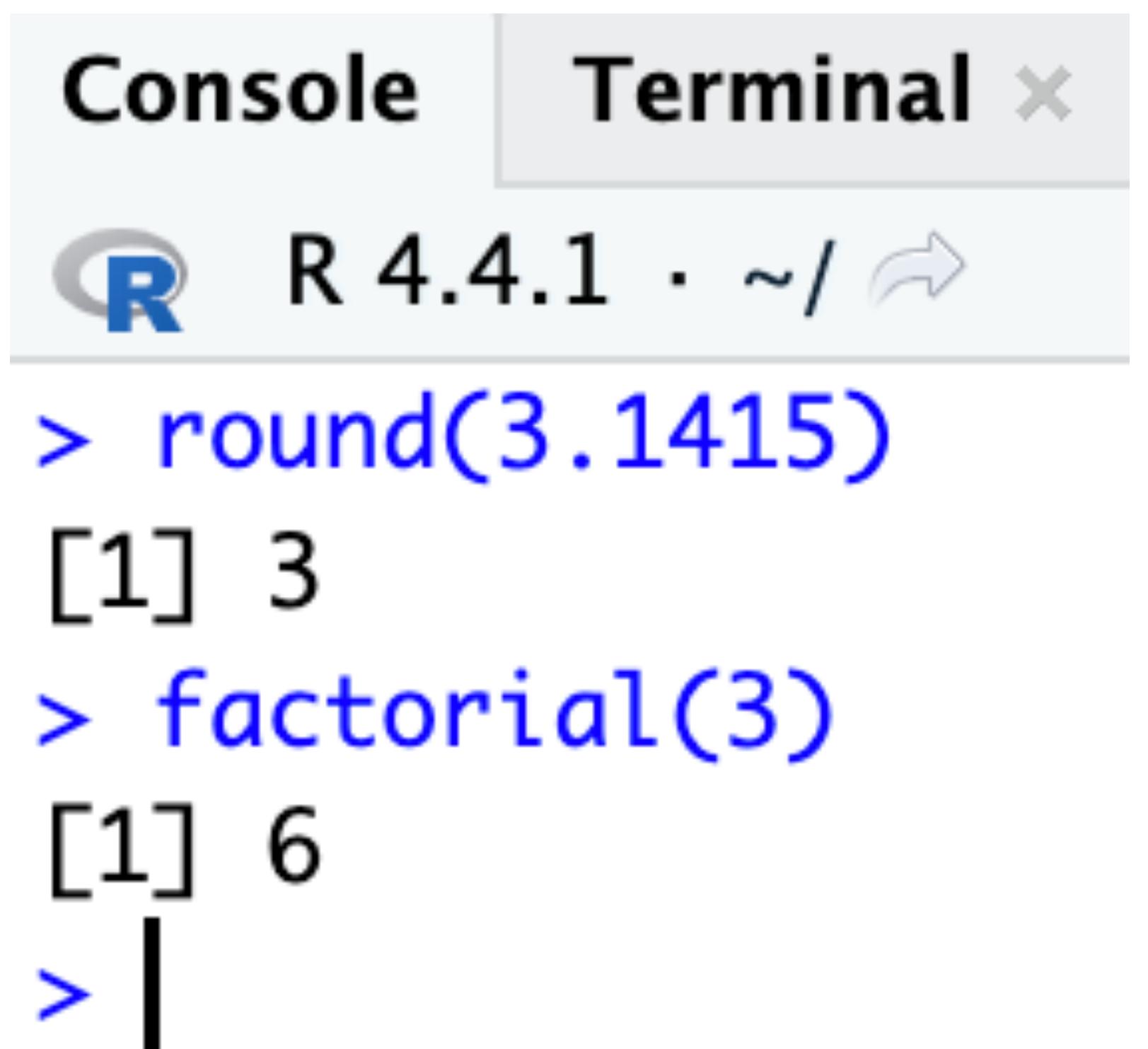
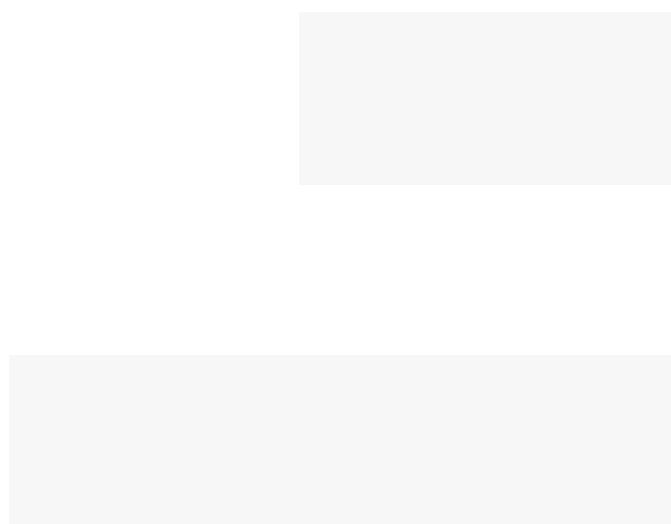
```
Console Terminal ×
R 4.4.1 · ~/ ↗
> round(3.1415)
[1] 3
> factorial(3)
[1] 6
> |
```



Running R code

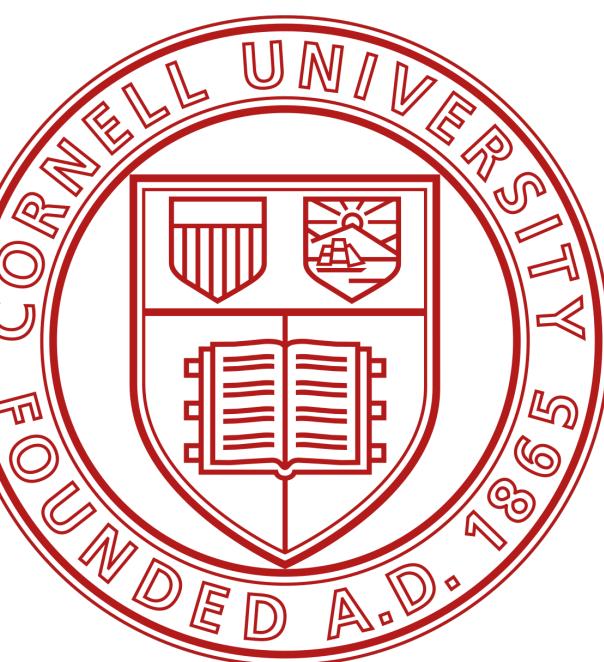
Functions

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



A screenshot of an R console window. The title bar says "Console Terminal x". The R logo icon is on the left. The text area shows the following R session:

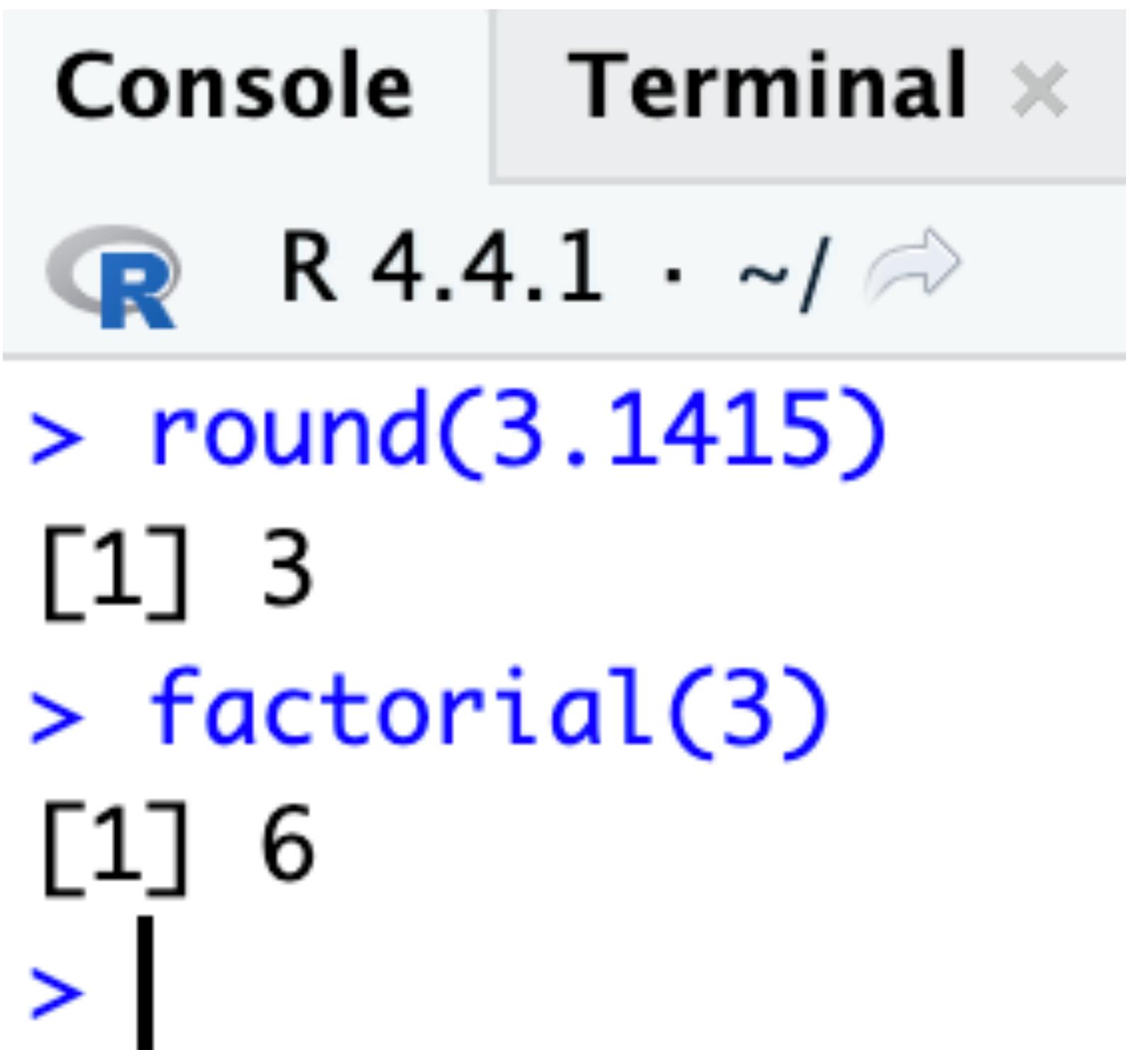
```
R 4.4.1 · ~/ ↗
> round(3.1415)
[1] 3
> factorial(3)
[1] 6
> |
```



Running R code

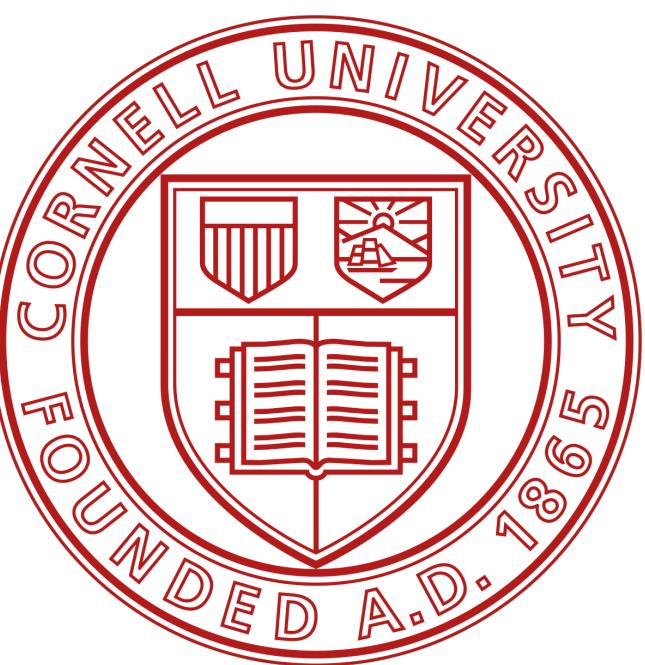
Functions

- 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.

A screenshot of an R console window. The title bar shows "Console" and "Terminal". The R logo icon and "R 4.4.1 · ~/..." are visible. The console area displays the following R session:

```
> round(3.1415)
[1] 3
> factorial(3)
[1] 6
> |
```

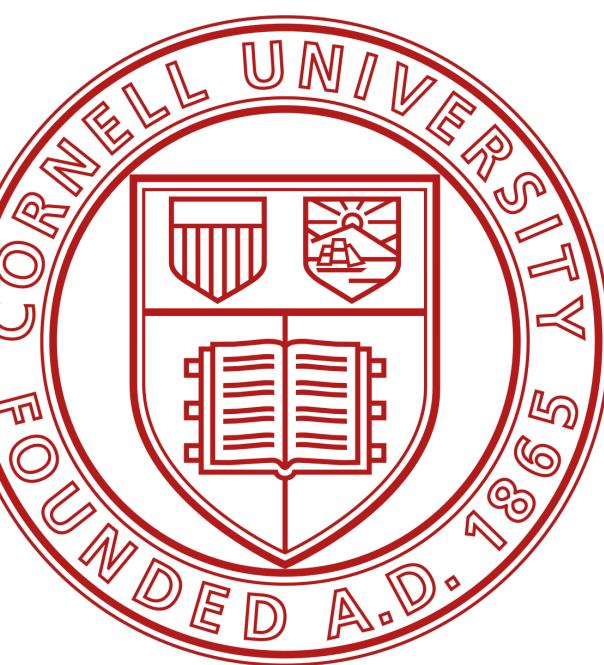
The cursor is shown as a vertical line after the final greater than sign.



Running R code

Functions

```
Console Terminal ×
R 4.4.1 · ~/ ↗
> mean(1:6)
[1] 3.5
> mean(die)
[1] 3.5
> round(mean(die))
[1] 4
> |
```

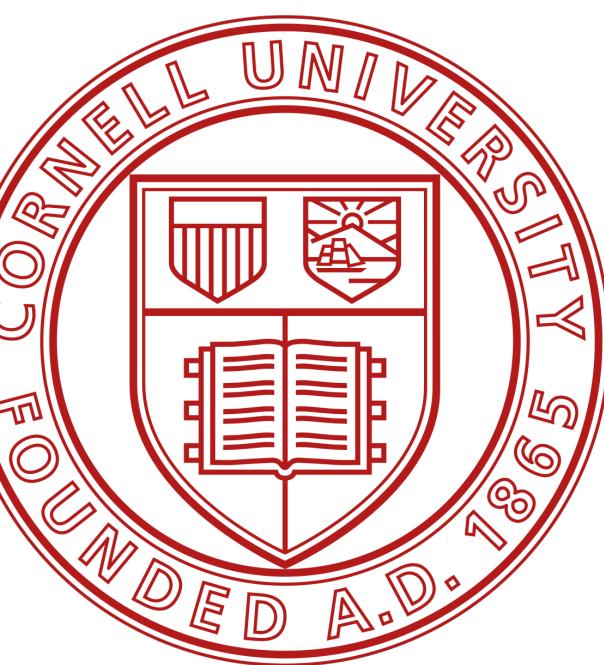


Running R code

Functions

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

```
Console Terminal ×
R 4.4.1 · ~/ ↗
> mean(1:6)
[1] 3.5
> mean(die)
[1] 3.5
> round(mean(die))
[1] 4
> |
```

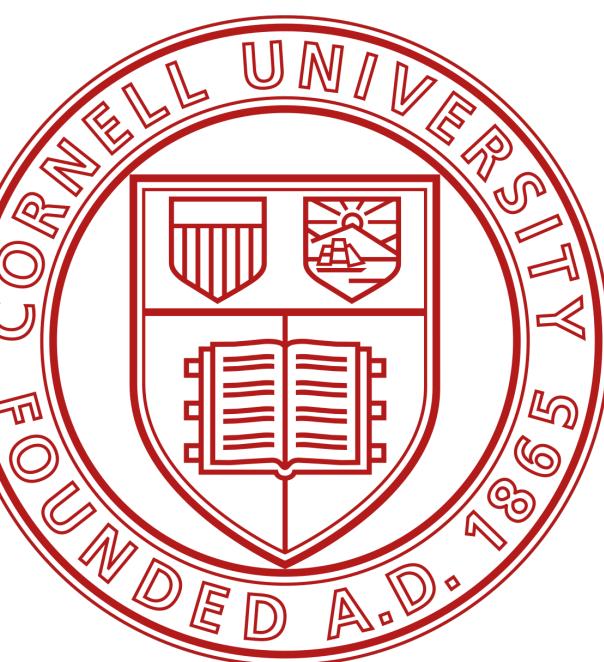


Running R code

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.

```
Console Terminal ×
R 4.4.1 · ~/ ↗
> mean(1:6)
[1] 3.5
> mean(die)
[1] 3.5
> round(mean(die))
[1] 4
> |
```

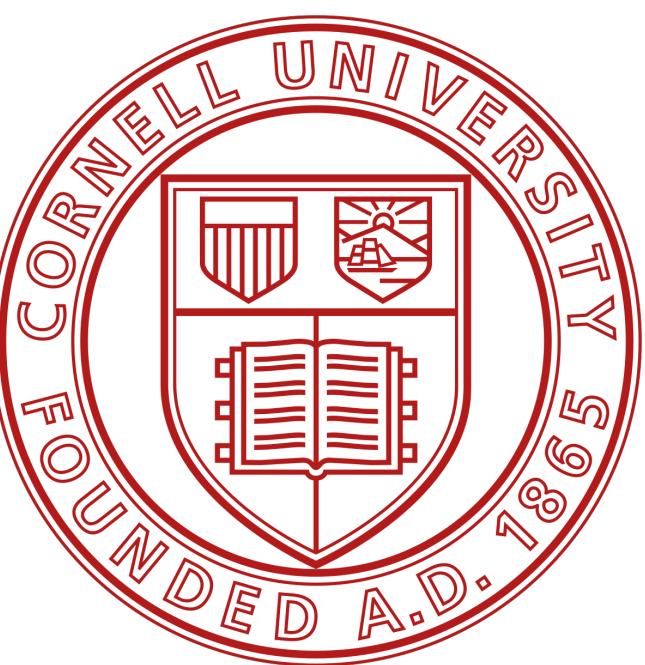


Running R code

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

```
Console Terminal ×
R 4.4.1 · ~/ ↗
> mean(1:6)
[1] 3.5
> mean(die)
[1] 3.5
> round(mean(die))
[1] 4
> |
```



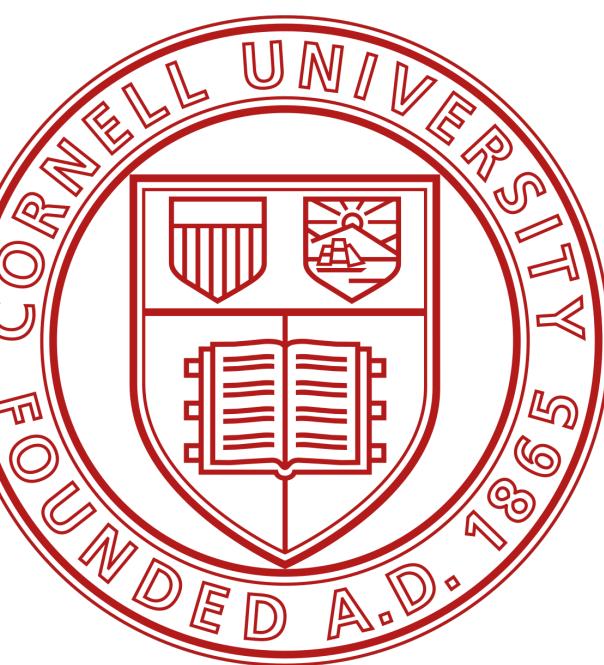
Running R code

Functions

Console Terminal ×

R R 4.4.1 · ~/ ↗

```
> mean(1:6)
[1] 3.5
> mean(die)
[1] 3.5
> round(mean(die))
[1] 4
> |
```

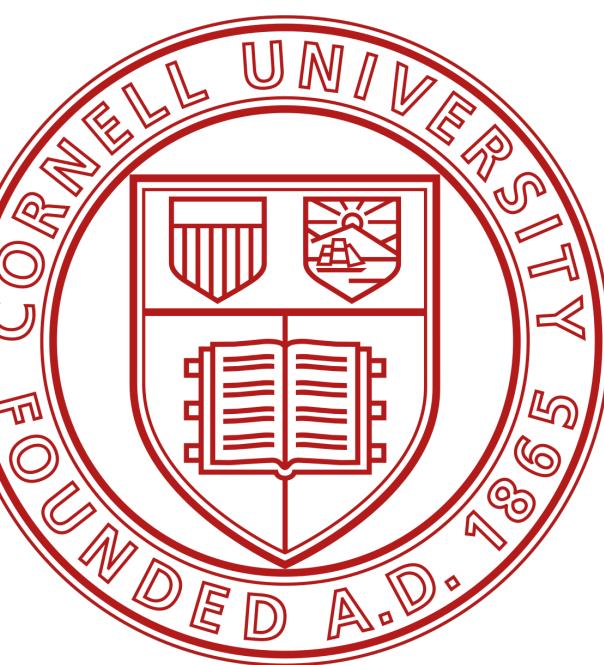


Running R code

Functions

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

```
Console Terminal ×
R 4.4.1 · ~/ ↗
> mean(1:6)
[1] 3.5
> mean(die)
[1] 3.5
> round(mean(die))
[1] 4
> |
```

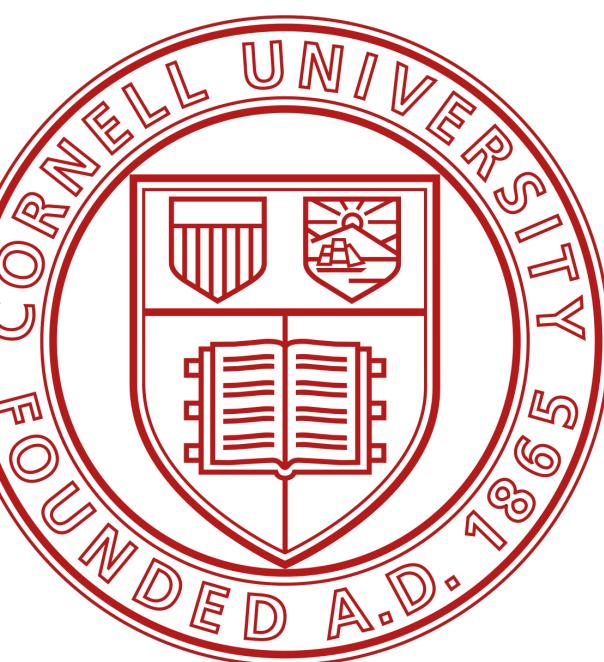


Running R code

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.

```
Console Terminal ×
R 4.4.1 · ~/ ↗
> mean(1:6)
[1] 3.5
> mean(die)
[1] 3.5
> round(mean(die))
[1] 4
> |
```

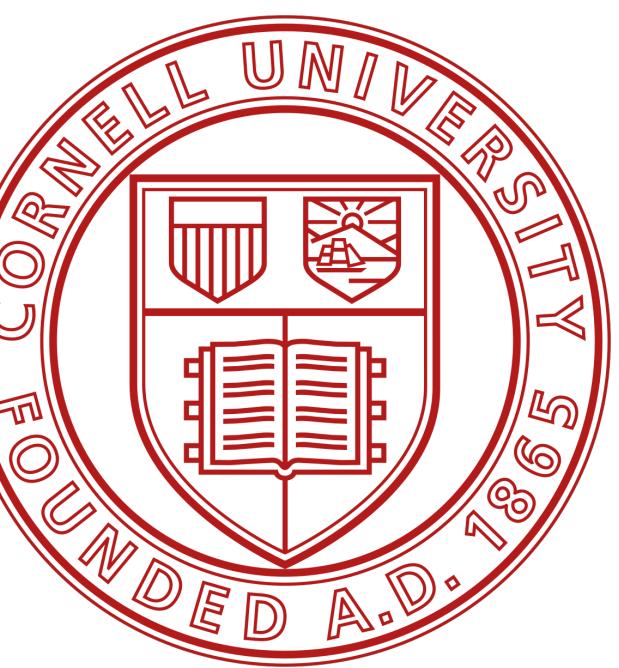


Running R code

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

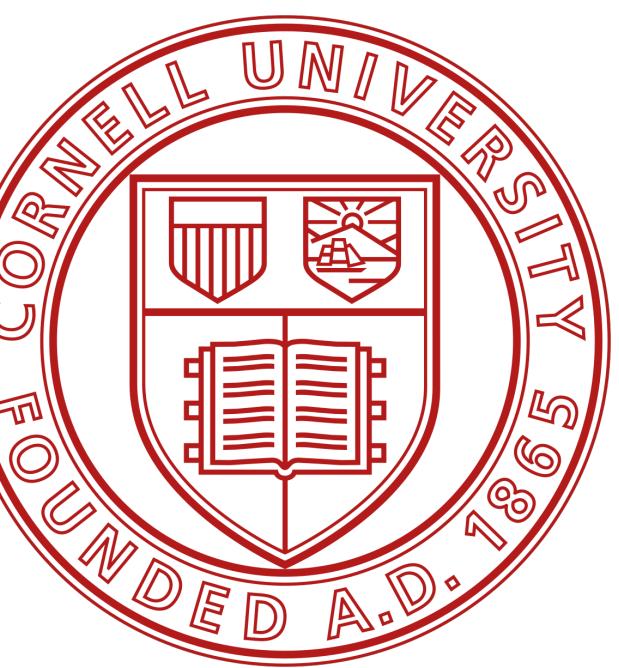
```
Console Terminal ×
R 4.4.1 · ~/ ↗
> mean(1:6)
[1] 3.5
> mean(die)
[1] 3.5
> round(mean(die))
[1] 4
> |
```



Basics

Roll a dice

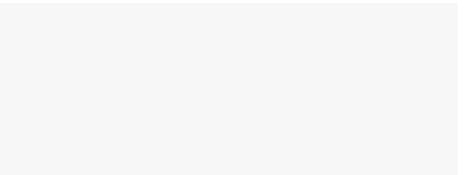


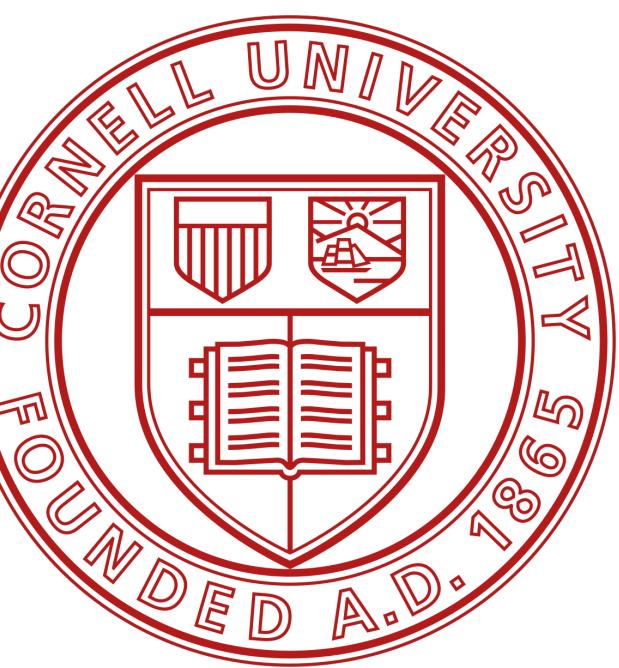


Basics

Roll a dice

- Let's roll the virtual die



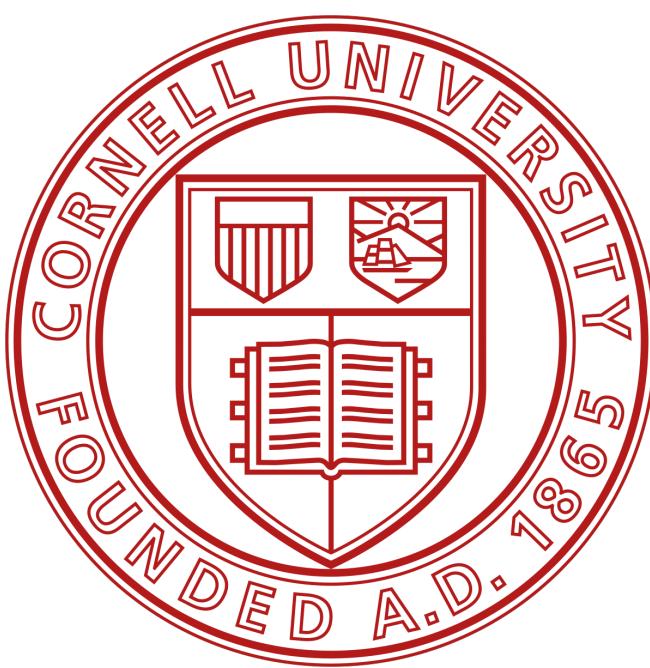


Basics

Roll a dice

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





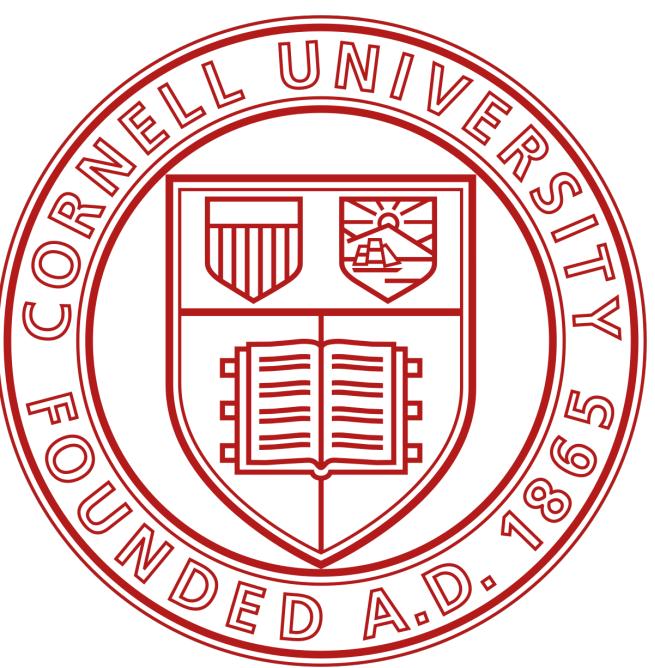
Basics

sample function

Console Terminal ×

R 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
> |
```

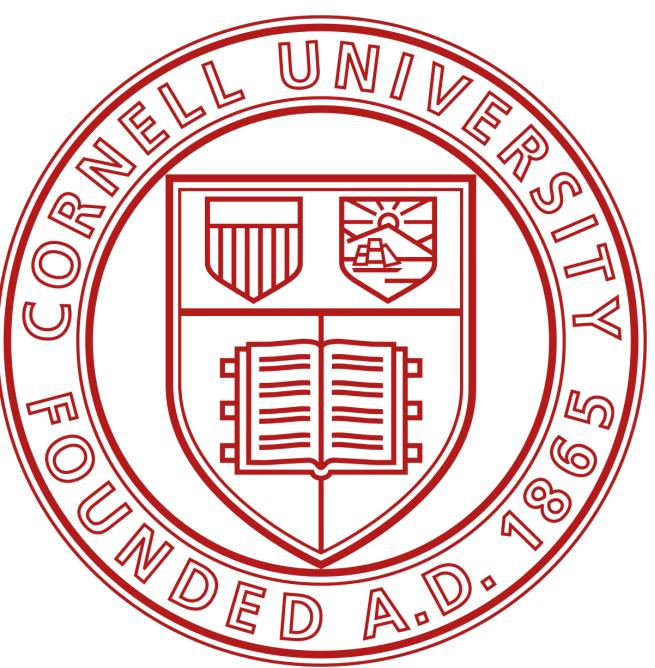


Basics

sample function

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

```
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
> |
```

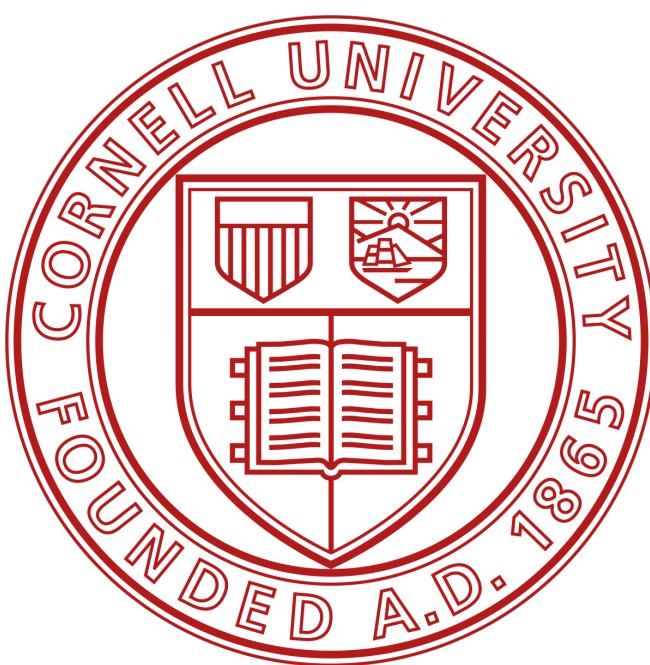


Basics

sample function

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

```
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
> |
```



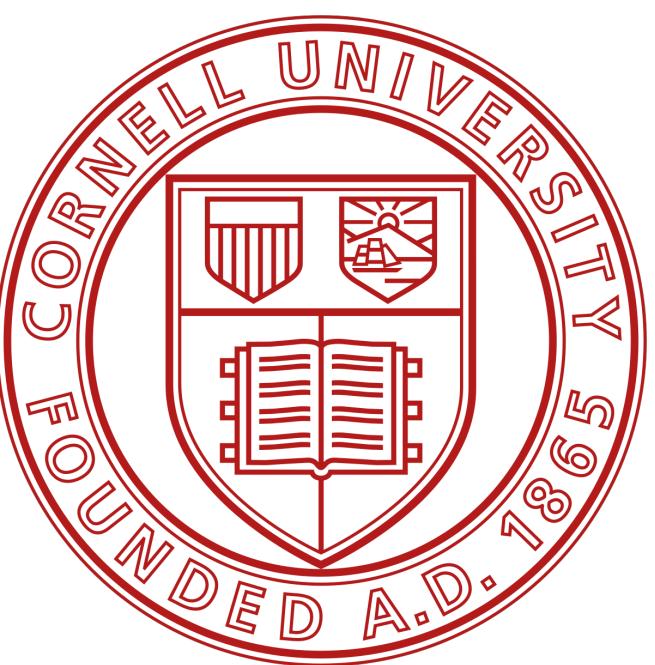
Basics

Arguments

Console Terminal ×

R 4.4.1 · ~/ ↗

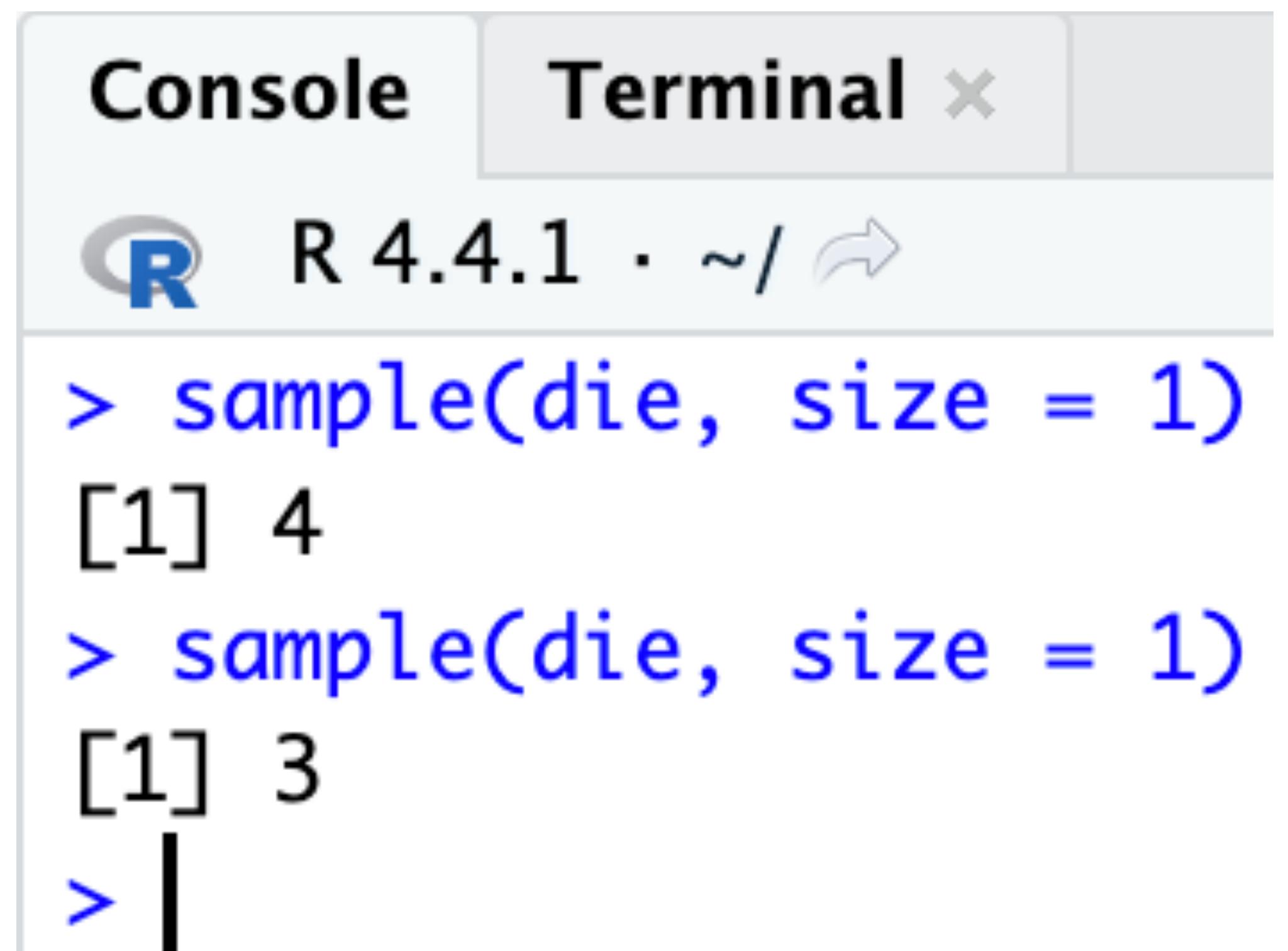
```
> sample(die, size = 1)
[1] 4
> sample(die, size = 1)
[1] 3
> |
```



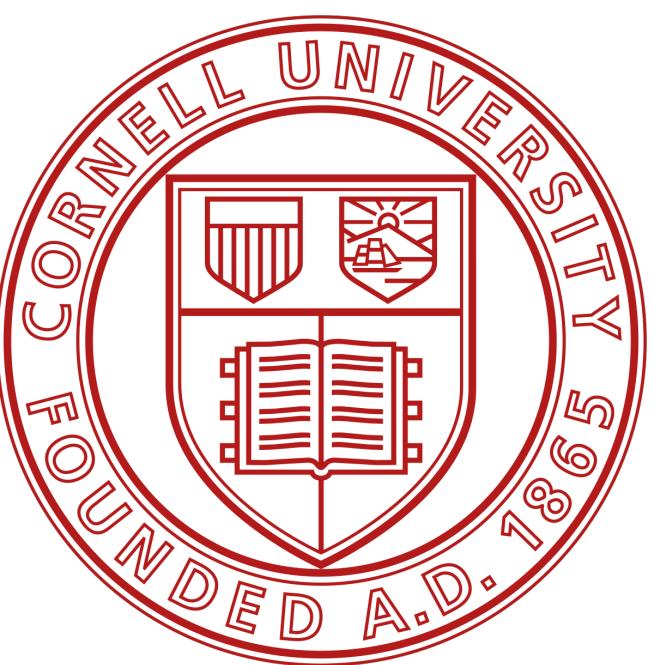
Basics

Arguments

- Every argument in every R function has a name.

A screenshot of the RStudio interface showing a single R console window. The window title is "Console". The R logo icon is visible next to the version number "R 4.4.1 · ~/". The console displays two identical R commands: "sample(die, size = 1)". Both executions resulted in the output "[1] 4". A cursor is shown at the end of the second command line, indicating it is ready for input.

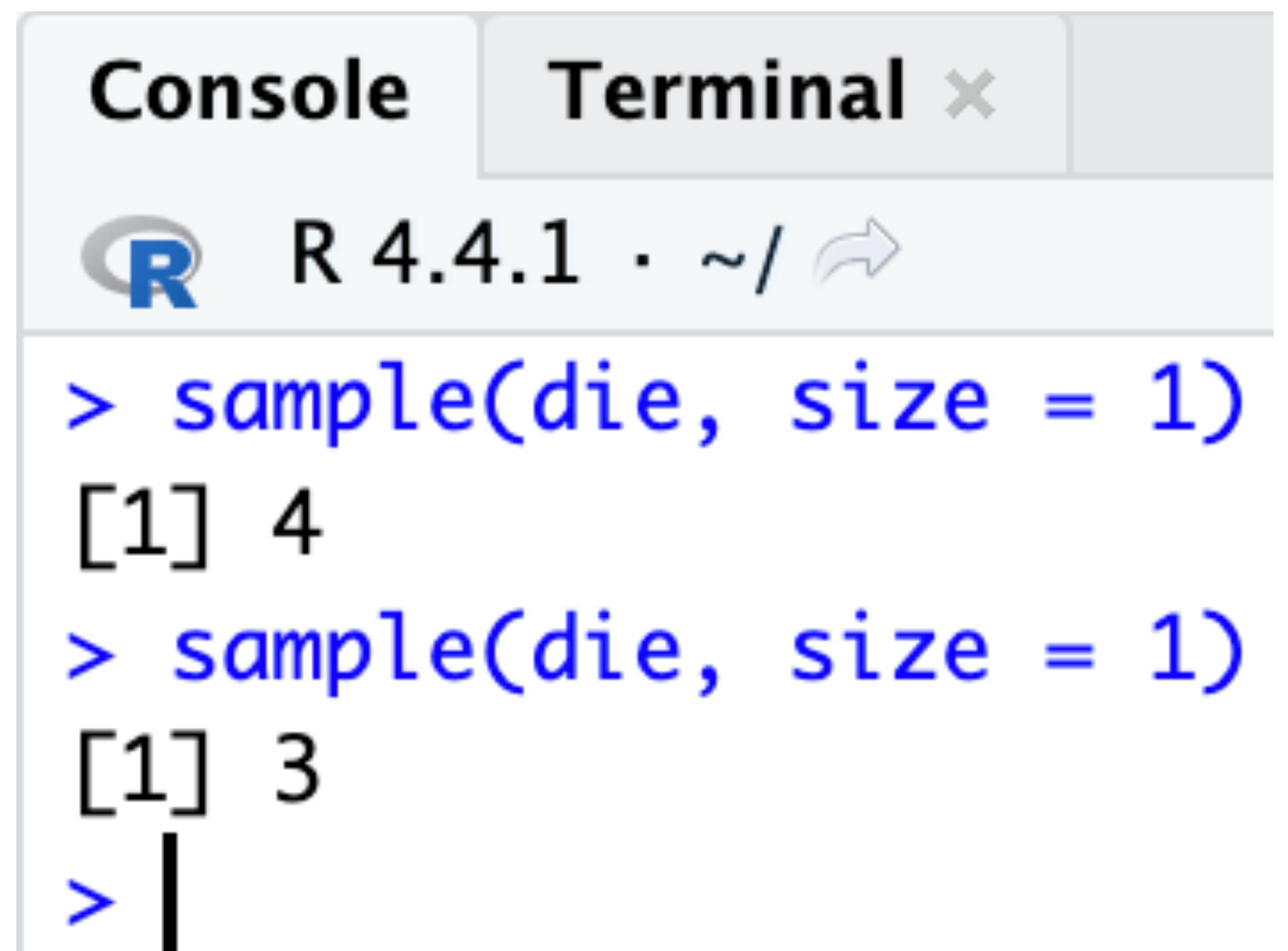
```
R 4.4.1 · ~/ 
> sample(die, size = 1)
[1] 4
> sample(die, size = 1)
[1] 3
>
```



Basics

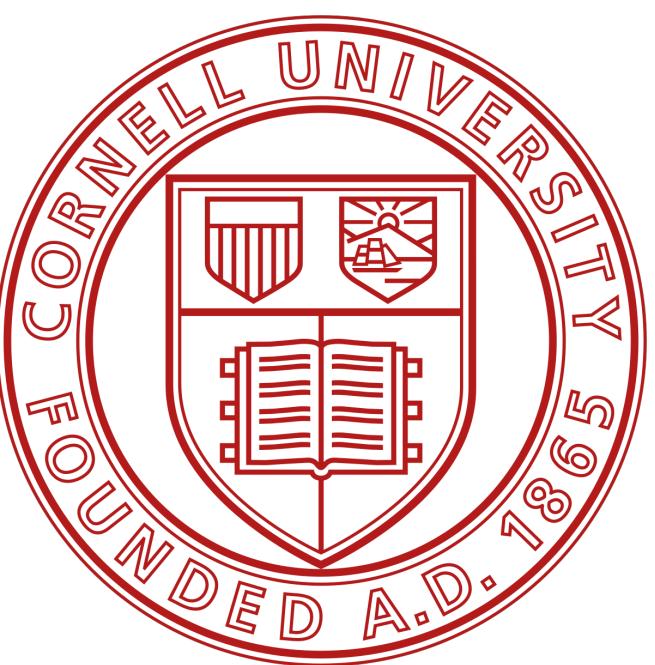
Arguments

- 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.



A screenshot of an R console window. The title bar says "Console Terminal x". The R logo is in the top-left corner. The text area shows the following R session:

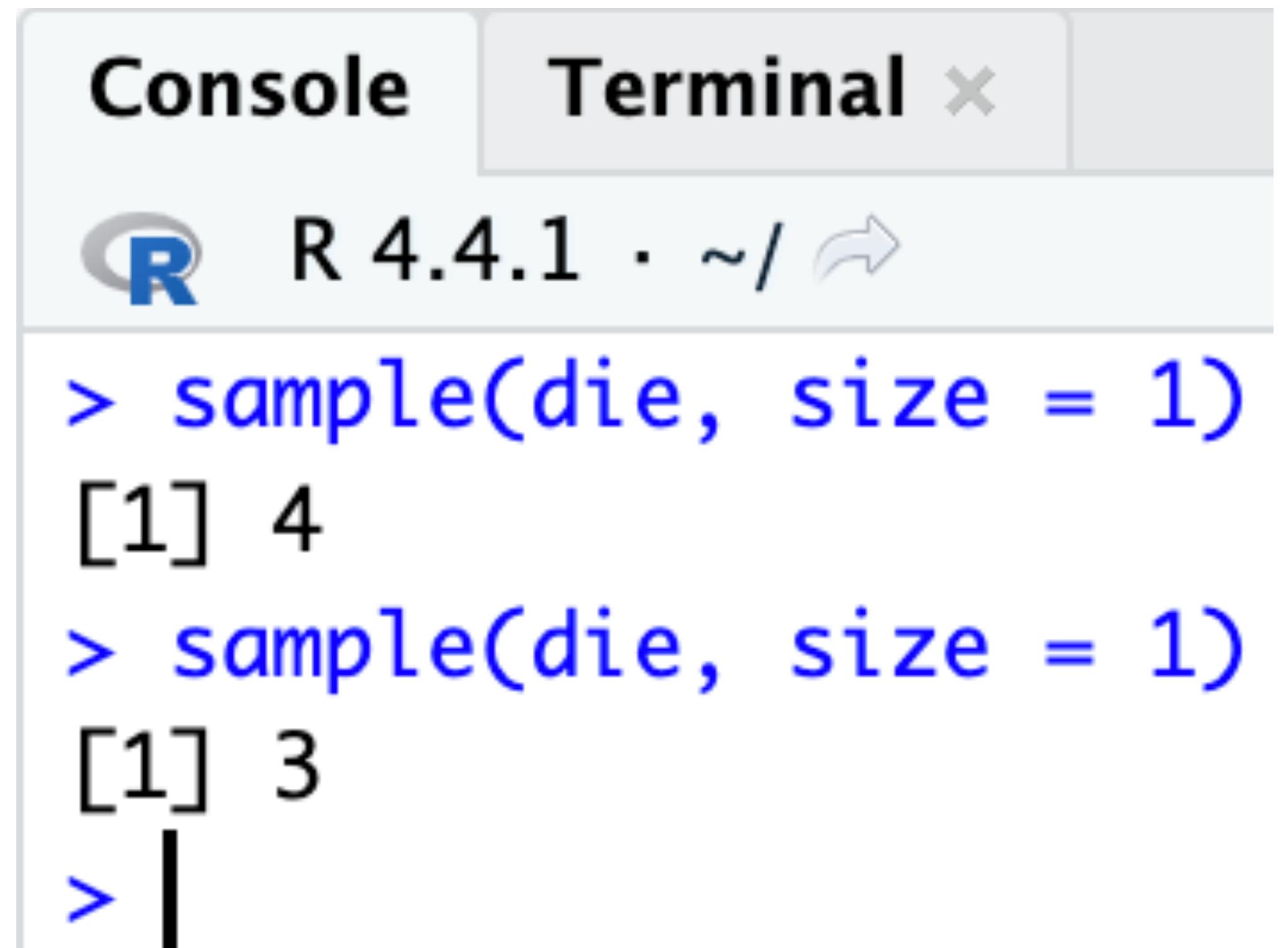
```
R 4.4.1 · ~/ ↗
> sample(die, size = 1)
[1] 4
> sample(die, size = 1)
[1] 3
> |
```



Basics

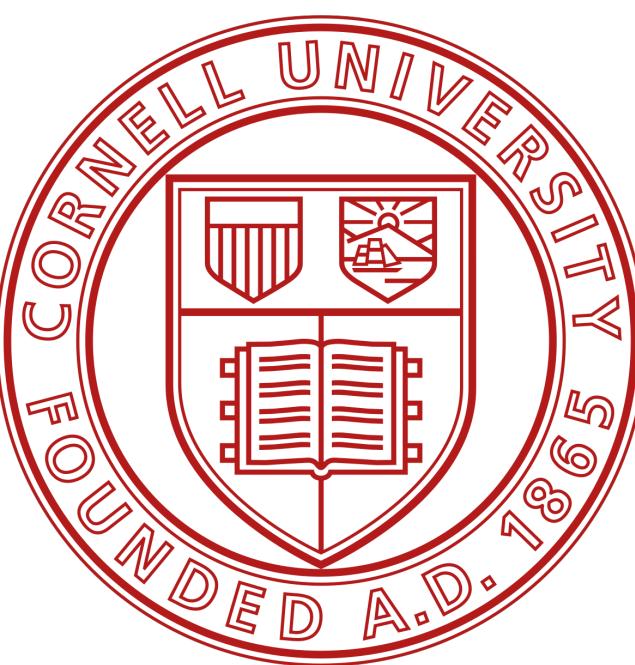
Arguments

- 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.



A screenshot of an R console window. The title bar says "Console Terminal x". The R logo is on the left. The text area shows R code and its output:

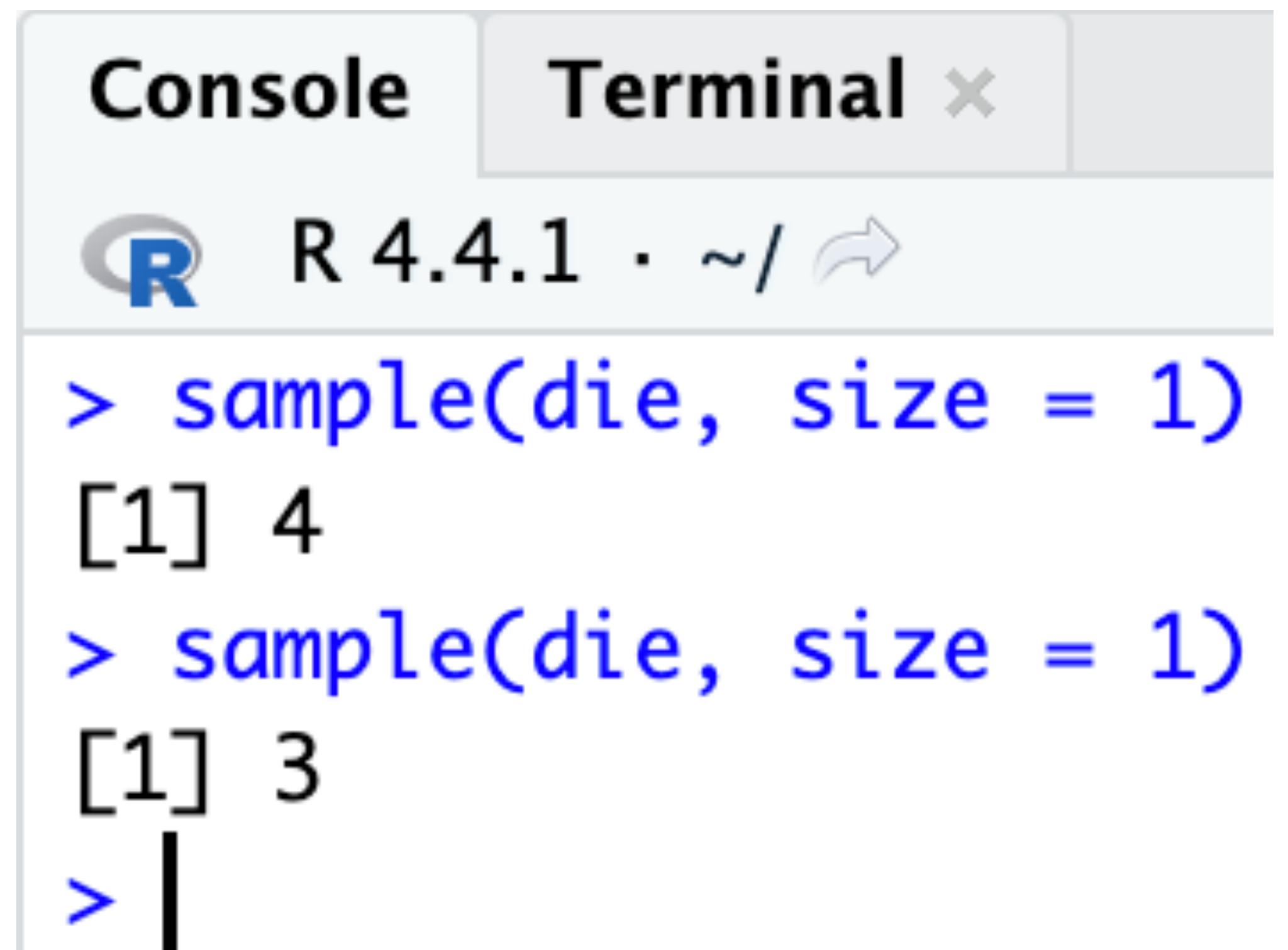
```
R 4.4.1 · ~/ ↗
> sample(die, size = 1)
[1] 4
> sample(die, size = 1)
[1] 3
> |
```



Basics

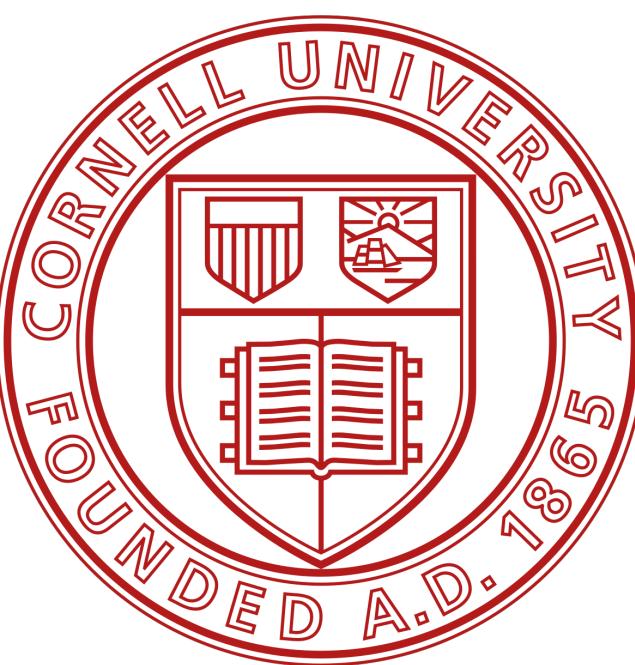
Arguments

- 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.



A screenshot of an R console window. The title bar says "Console Terminal x". The R logo is on the left. The text area shows:

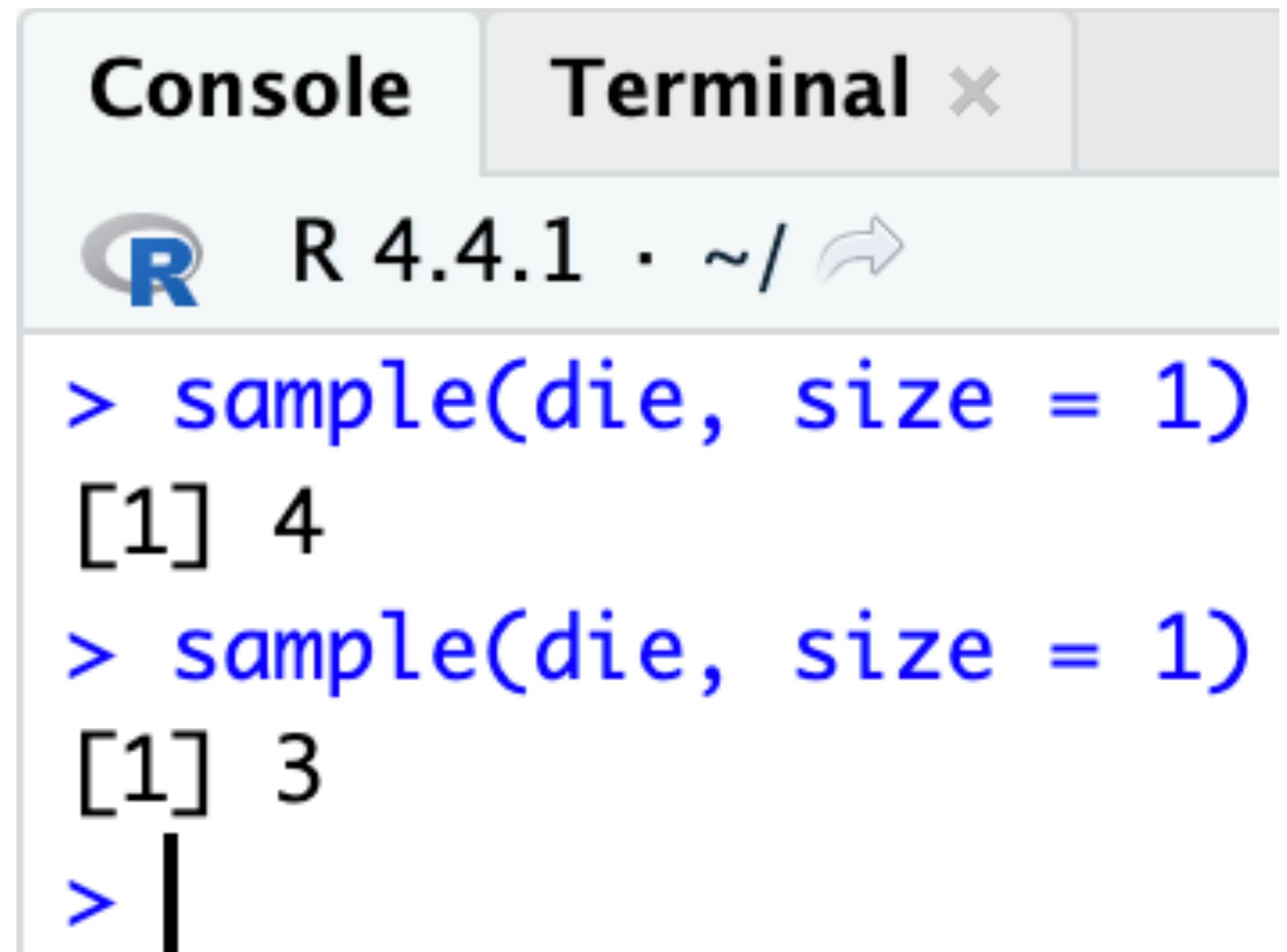
```
R 4.4.1 · ~/ ↗
> sample(die, size = 1)
[1] 4
> sample(die, size = 1)
[1] 3
> |
```



Basics

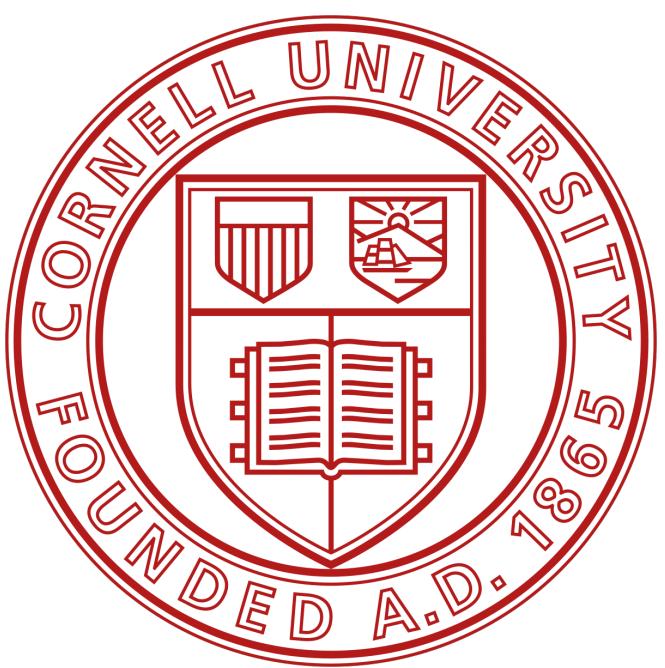
Arguments

- 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



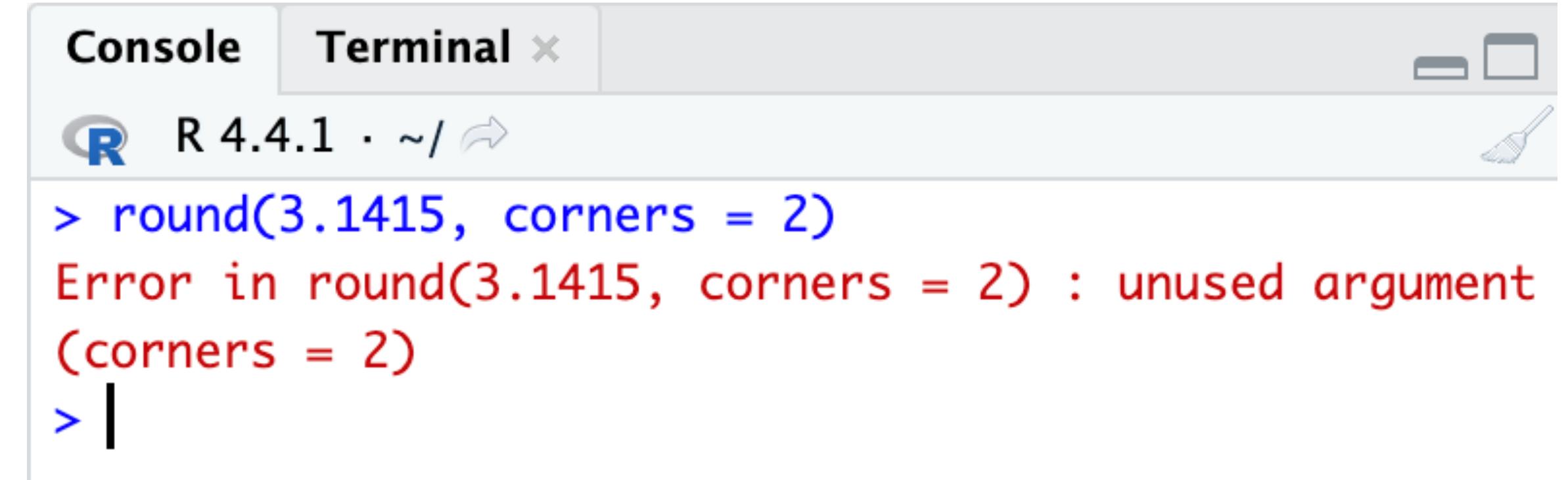
A screenshot of an R console window. The title bar says "Console Terminal x". The R logo is on the left. The text area shows:

```
R 4.4.1 · ~/ ↗
> sample(die, size = 1)
[1] 4
> sample(die, size = 1)
[1] 3
> |
```



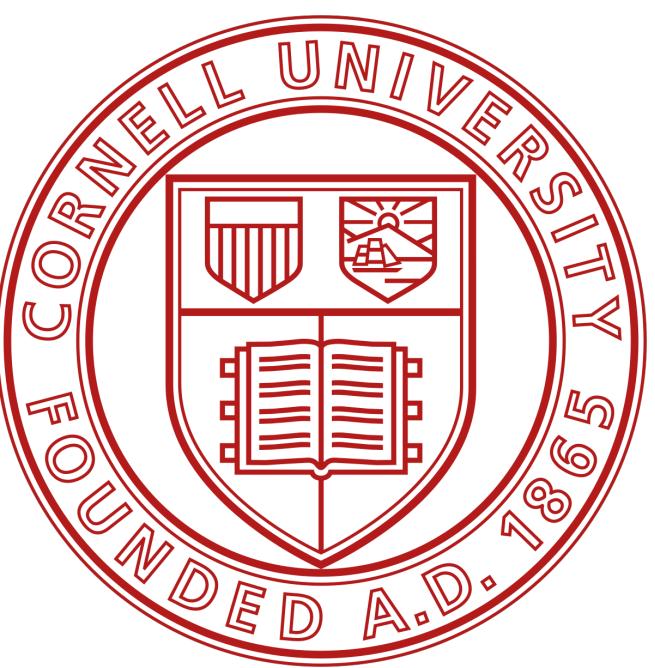
Basics

Arguments



A screenshot of the RStudio interface showing the Console tab selected. The R logo icon is visible next to the text "R 4.4.1 · ~/". The console output shows:

```
R 4.4.1 · ~/ 
> round(3.1415, corners = 2)
Error in round(3.1415, corners = 2) : unused argument
(corners = 2)
> |
```



Basics

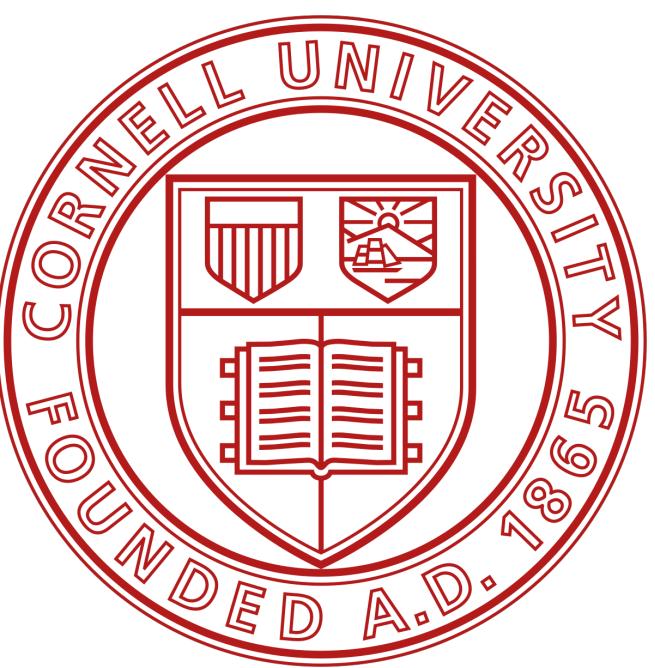
Arguments

- How do you know which argument names to use?

A screenshot of the RStudio interface showing the Console tab. The console window title is "Console Terminal x". The R logo icon and the text "R 4.4.1 · ~/ ↗" are visible. The console output shows:

```
> round(3.1415, corners = 2)
Error in round(3.1415, corners = 2) : unused argument
(corners = 2)
> |
```

The word "corners" in the error message is highlighted in blue, indicating it is a misspelled argument name.



Basics

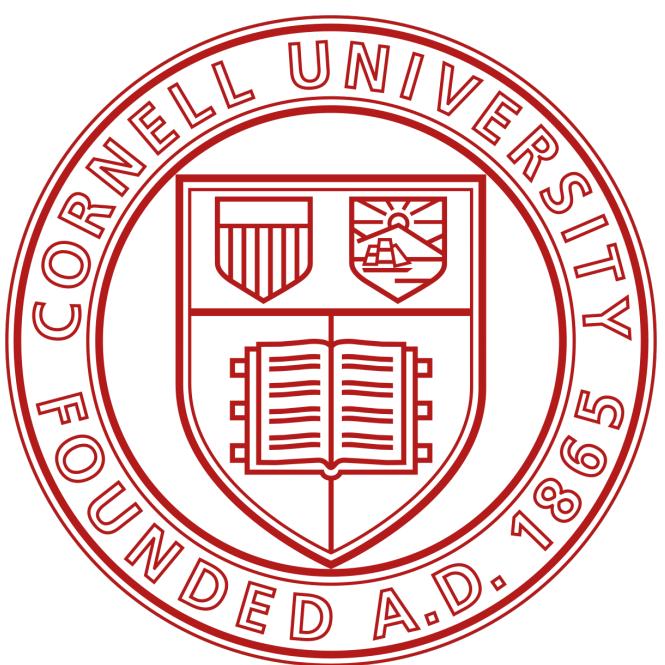
Arguments

- 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.

A screenshot of the RStudio interface showing the Console tab. The R logo icon is visible next to the text "R 4.4.1 · ~/". The console output shows:

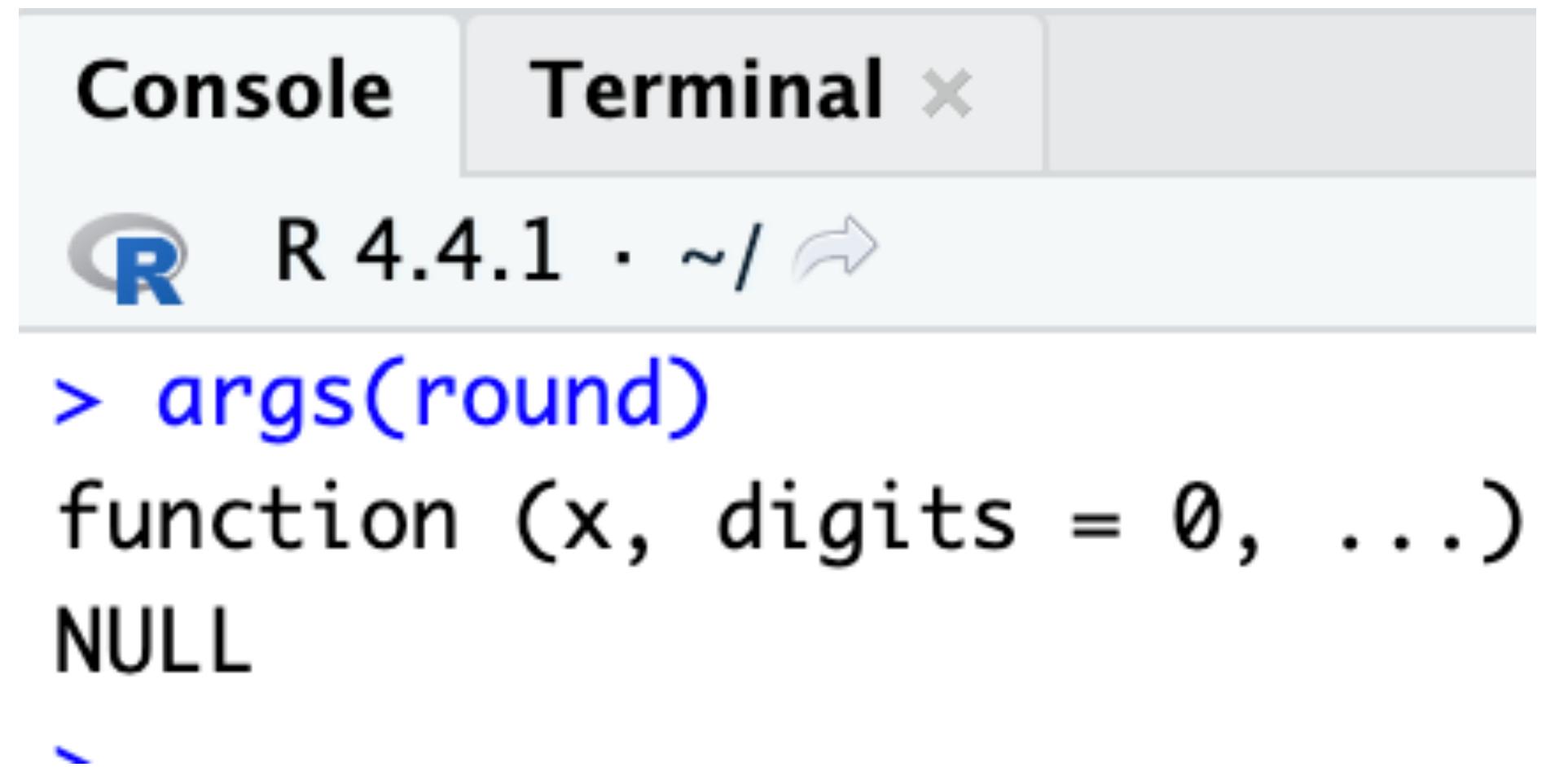
```
Console Terminal ×
R 4.4.1 · ~/ 
> round(3.1415, corners = 2)
Error in round(3.1415, corners = 2) : unused argument
(corners = 2)
> |
```

The "Terminal" tab is shown as inactive.



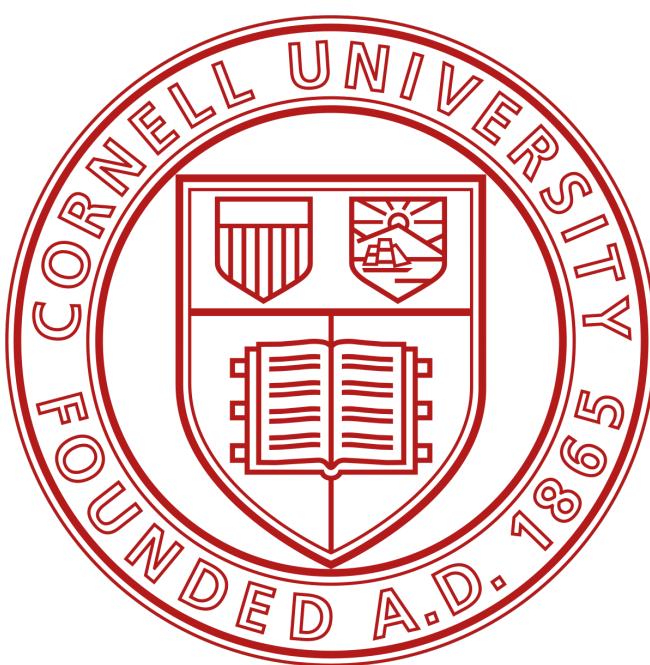
Basics

Arguments



A screenshot of the RStudio interface showing a console window. The title bar includes tabs for 'Console' and 'Terminal'. The terminal tab is active, showing the R logo and the text 'R 4.4.1 · ~/'. The console window displays the following R code and output:

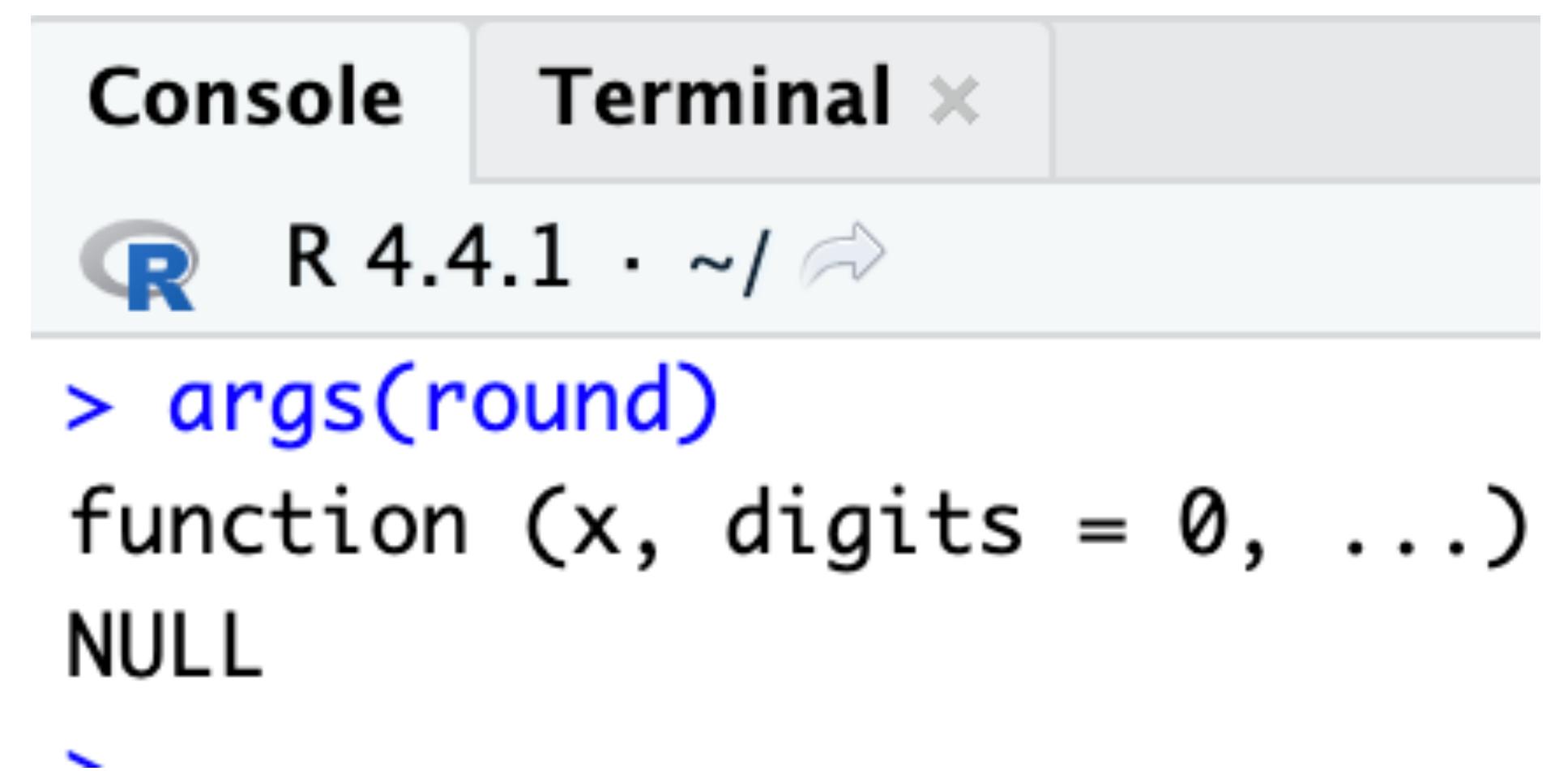
```
> args(round)
function (x, digits = 0, ...)
NULL
`
```



Basics

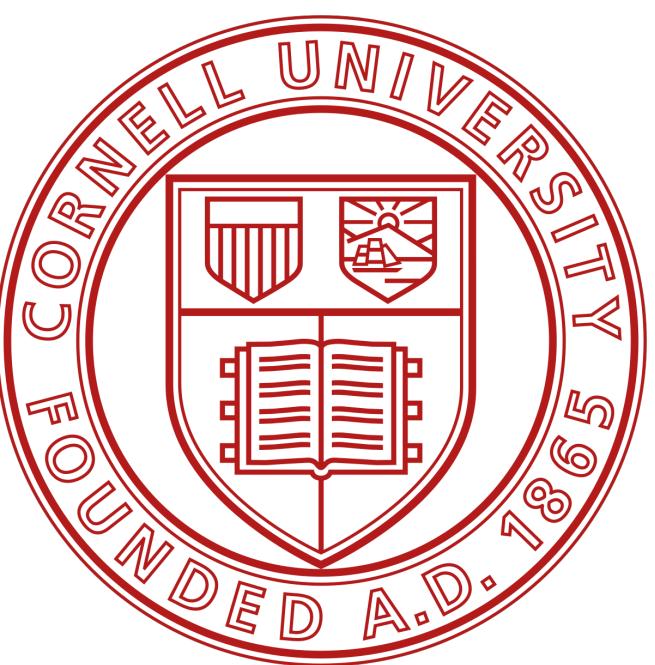
Arguments

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

A screenshot of the RStudio interface showing the Terminal tab. The terminal window displays the following R session:

```
Console Terminal ×
R 4.4.1 · ~/ ↗
> args(round)
function (x, digits = 0, ...)
NULL
`
```

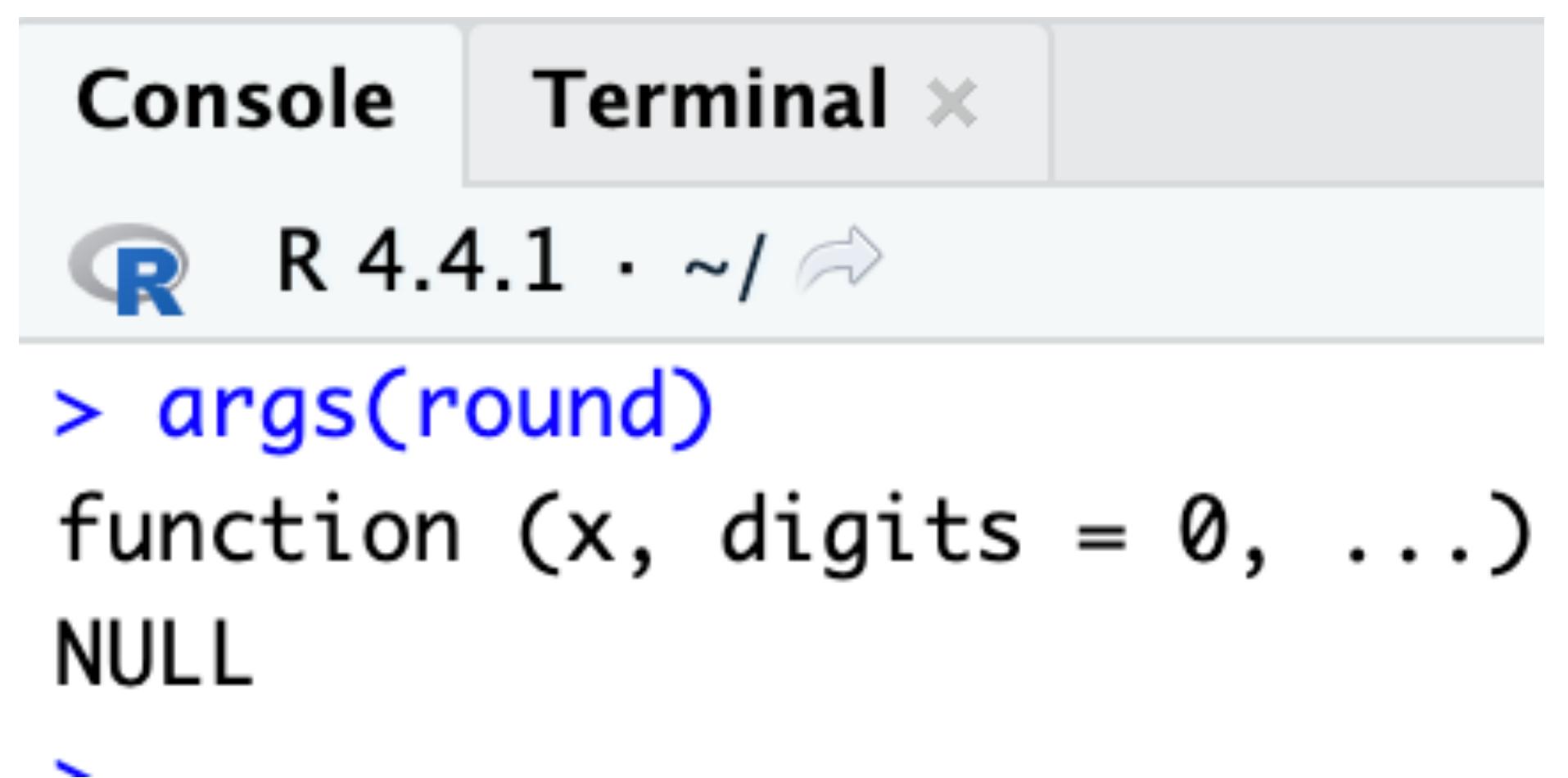
The "Console" tab is also visible above the terminal window.



Basics

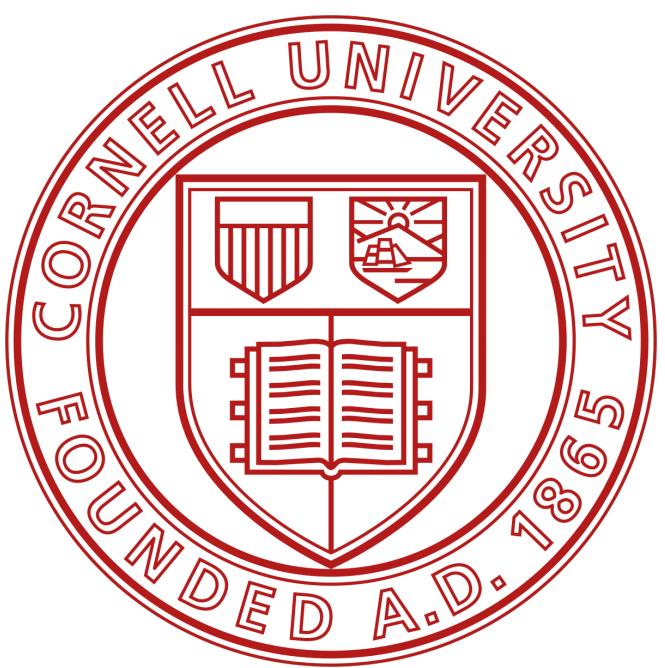
Arguments

- 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`



A screenshot of an R console window. The title bar says "Console Terminal ×". The R logo icon is next to the text "R 4.4.1 · ~/". The console output shows:

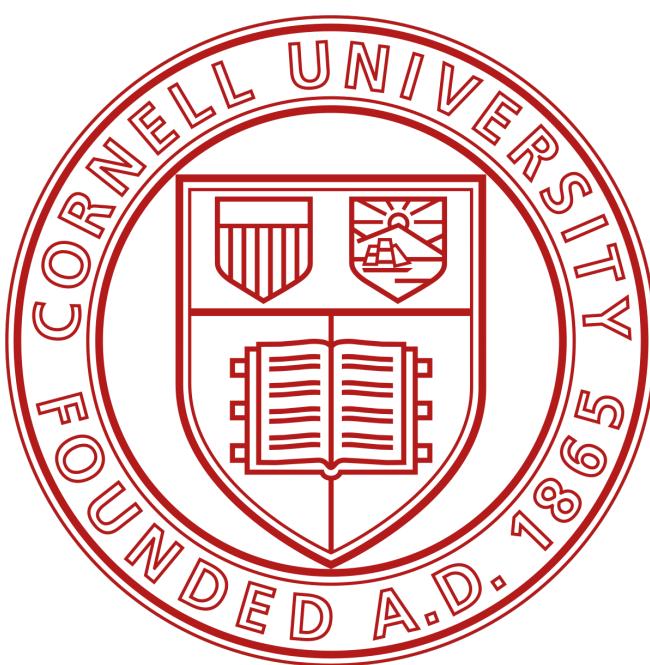
```
> args(round)
function (x, digits = 0, ...)
NULL
`
```



Basics

Arguments

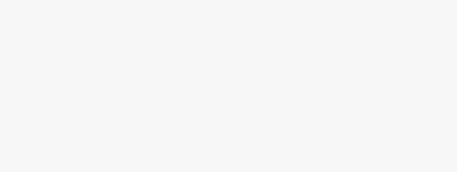
```
Console Terminal ×
R 4.4.1 · ~/ ↗
> round(3.1415)
[1] 3
> round(3.1415, digits = 2)
[1] 3.14
>
```



Basics

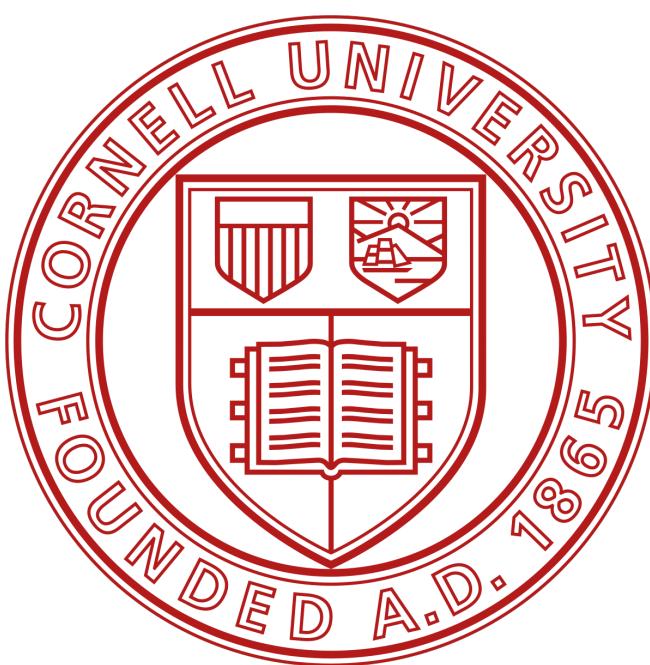
Arguments

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



The screenshot shows the RStudio interface with the 'Console' tab selected. The R logo icon is visible next to the 'R 4.4.1 · ~/`' text. Below it, two lines of R code are shown, each followed by its output:

```
R 4.4.1 · ~/`  
> round(3.1415)  
[1] 3  
> round(3.1415, digits = 2)  
[1] 3.14  
>
```

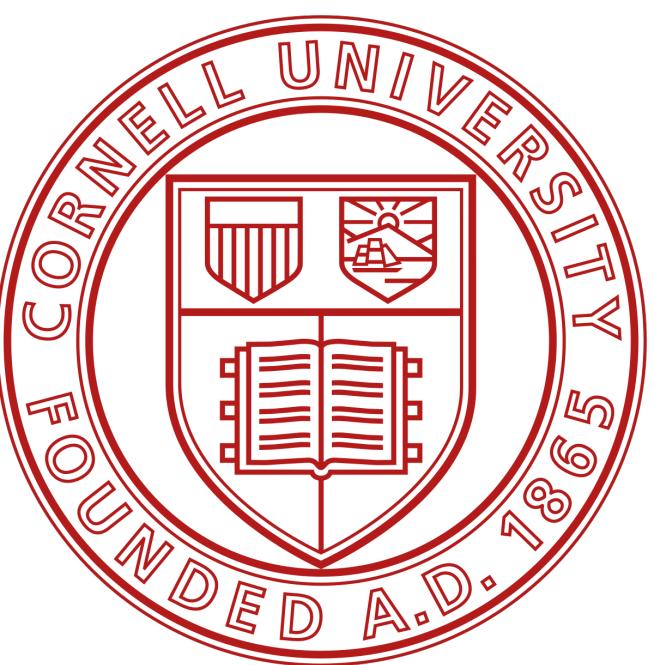


Basics

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`.

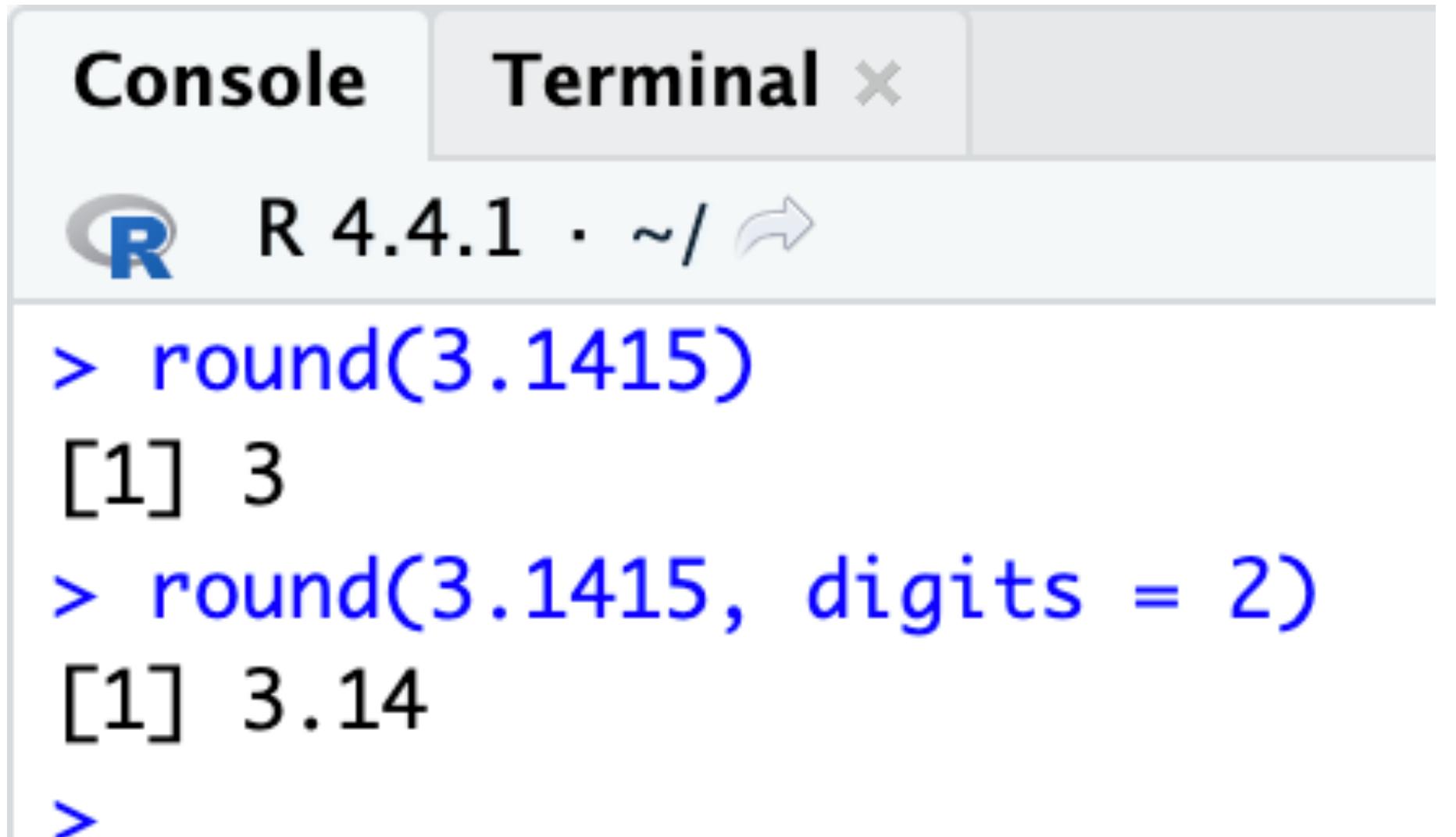
```
Console Terminal ×
R 4.4.1 · ~/ ↗
> round(3.1415)
[1] 3
> round(3.1415, digits = 2)
[1] 3.14
>
```



Basics

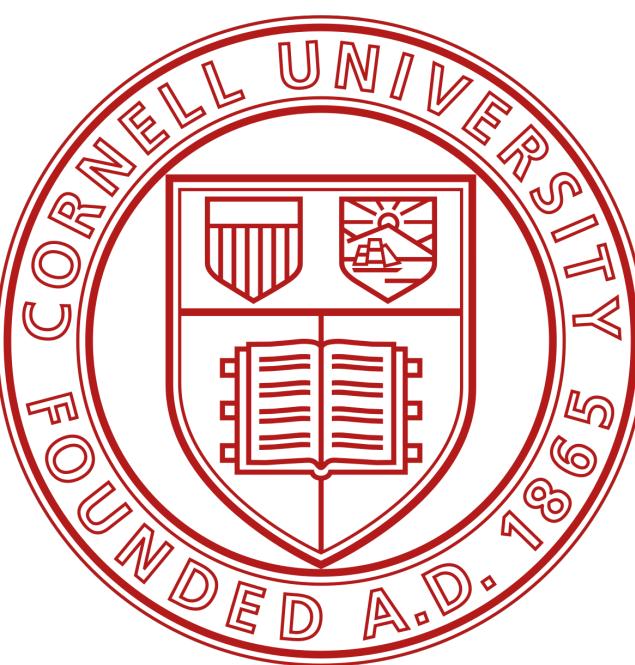
Arguments

- 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.



A screenshot of the RStudio interface showing the Console tab. The R logo icon is visible next to the text "R 4.4.1 · ~/". Below it, two lines of R code are shown, each followed by its output:

```
> round(3.1415)
[1] 3
> round(3.1415, digits = 2)
[1] 3.14
>
```

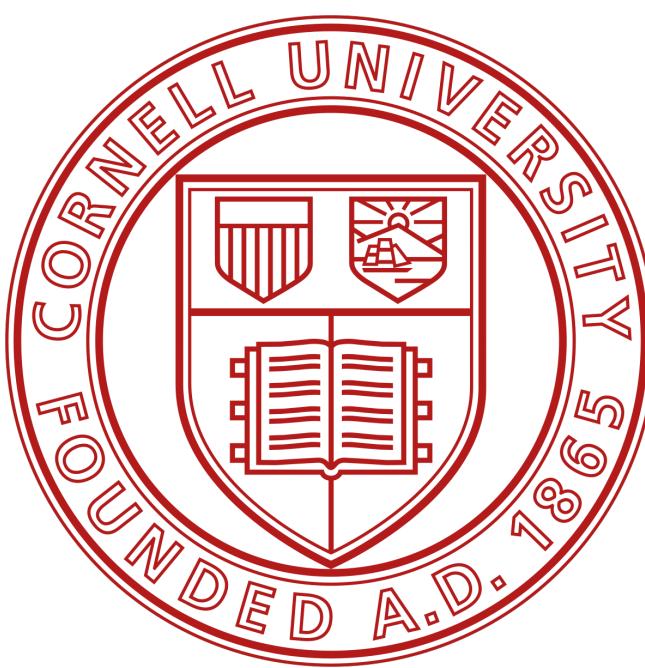


Basics

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.

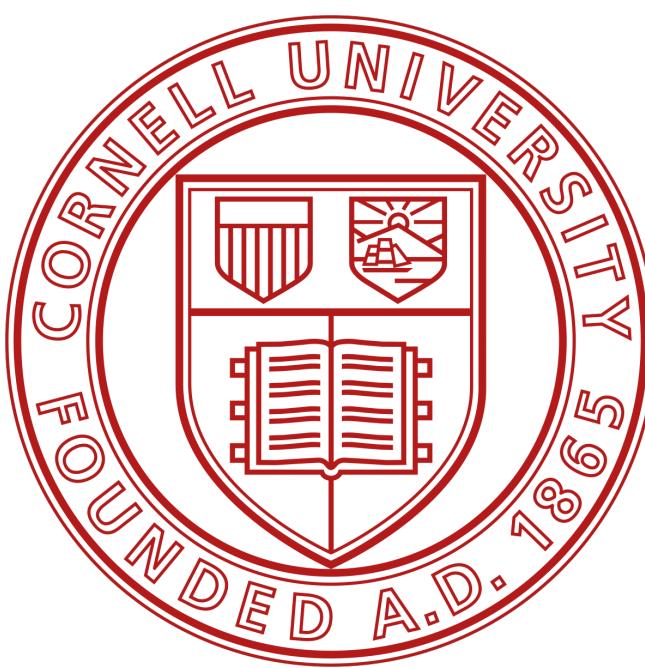
```
Console Terminal ×
R 4.4.1 · ~/ ↗
> round(3.1415)
[1] 3
> round(3.1415, digits = 2)
[1] 3.14
>
```



Basics

Sample with replacement

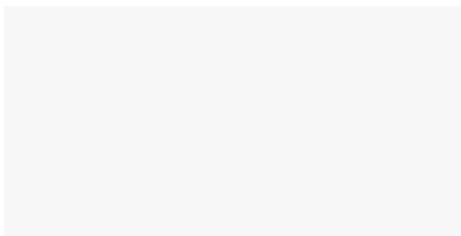
```
Console Terminal ×
R 4.4.1 · ~/ ↗
> sample(die, size = 2)
[1] 1 6
> sample(die, size = 2)
[1] 5 3
> sample(die, size = 2)
[1] 2 1
> sample(die, size = 2)
[1] 4 3
> sample(die, size = 2)
[1] 1 3
> sample(die, size = 2)
[1] 2 1
> sample(die, size = 2)
[1] 4 5
> sample(die, size = 2)
[1] 1 2
> sample(die, 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
> |
```



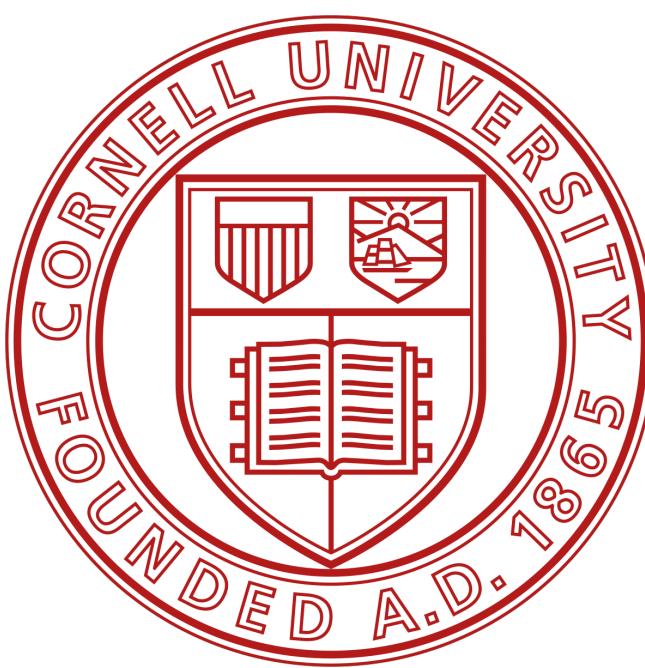
Basics

Sample with replacement

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



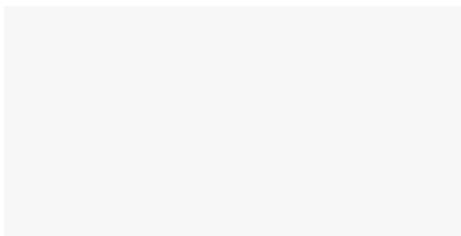
```
Console Terminal ×
R 4.4.1 · ~/ ↗
> sample(die, size = 2)
[1] 1 6
> sample(die, size = 2)
[1] 5 3
> sample(die, size = 2)
[1] 2 1
> sample(die, size = 2)
[1] 4 3
> sample(die, size = 2)
[1] 1 3
> sample(die, size = 2)
[1] 2 1
> sample(die, size = 2)
[1] 4 5
> sample(die, size = 2)
[1] 1 2
> sample(die, 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
> |
```



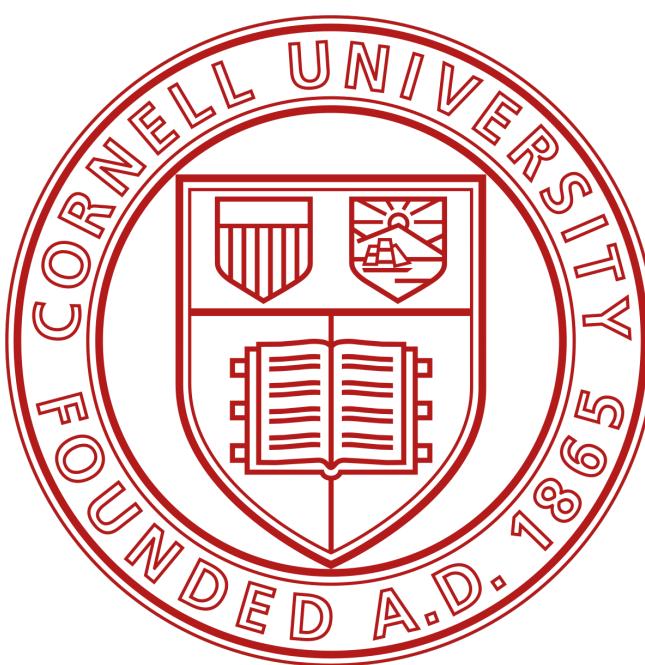
Basics

Sample with replacement

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



```
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.
- “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  
> |
```