Intro to R/RStudio

2020-07-14

Programming To-Dos:

1. Install both R and RStudio
2. Open RStudio. Familiarize yourself with the layout
3. Create a new R project in your respective folder
4. Open a new R script file
5. Install your first package!

What is R?

A Language for Statistical Computing and Data Science

[Home]

Download

CRAN

R Project

About R
Logo
Contributors

The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.

If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

Wide variety of statistical and graphical techniques

Install R

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

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
- Download R for (Mac) OS X
- 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!

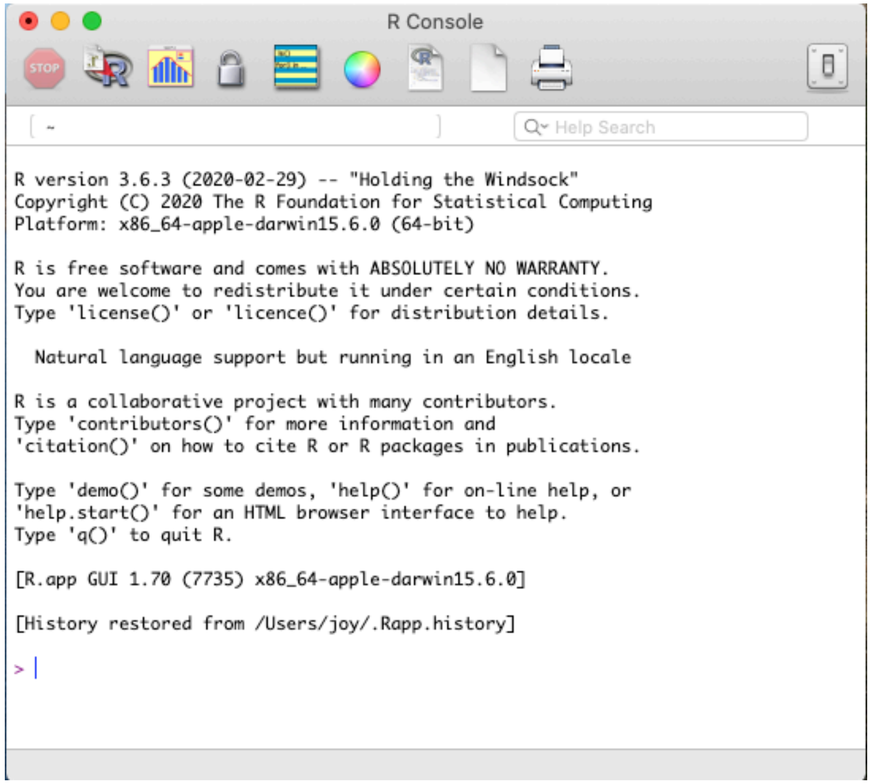
- The latest release (2020-06-22, Taking Off Again) R-4.0.2.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. 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

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

R


JUL
13
Joy_worm_imag


R Console
STOP
~
Help Search
R version 3.6.3 (2020-02-29) -- "Holding the Windsock"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)
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 R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[R.app GUI 1.70 (7735) x86_64-apple-darwin15.6.0]
[History restored from /Users/joy/.Rapp.history]
>

Install RStudio

Download RStudio

Choose Your Version

RStudio is a set of integrated tools designed to help you be more productive with R. It includes a console, syntax-highlighting editor that supports direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace.

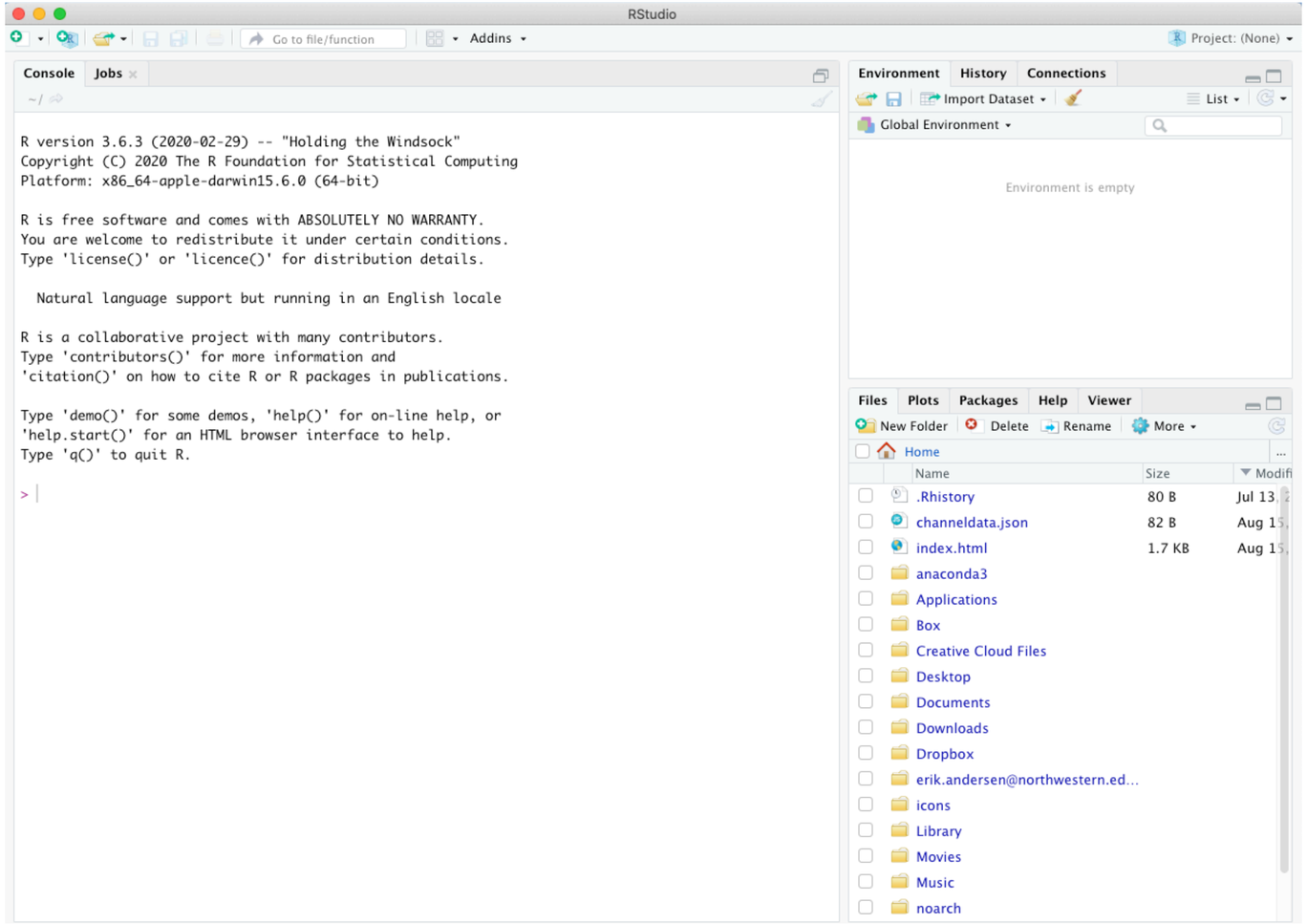
LEARN MORE ABOUT RSTUDIO FEATURES

RStudio's new solution for every professional data science team. RStudio Team includes RStudio Server Pro, RStudio Connect and RStudio Package Manager.

LEARN MORE

RStudio Desktop	RStudio Desktop	RStudio Server	RStudio Server Pro
Open Source License	Commercial License	Open Source License	Commercial License
Free	$995	Free	$4,975
	/year		/year

RStudio

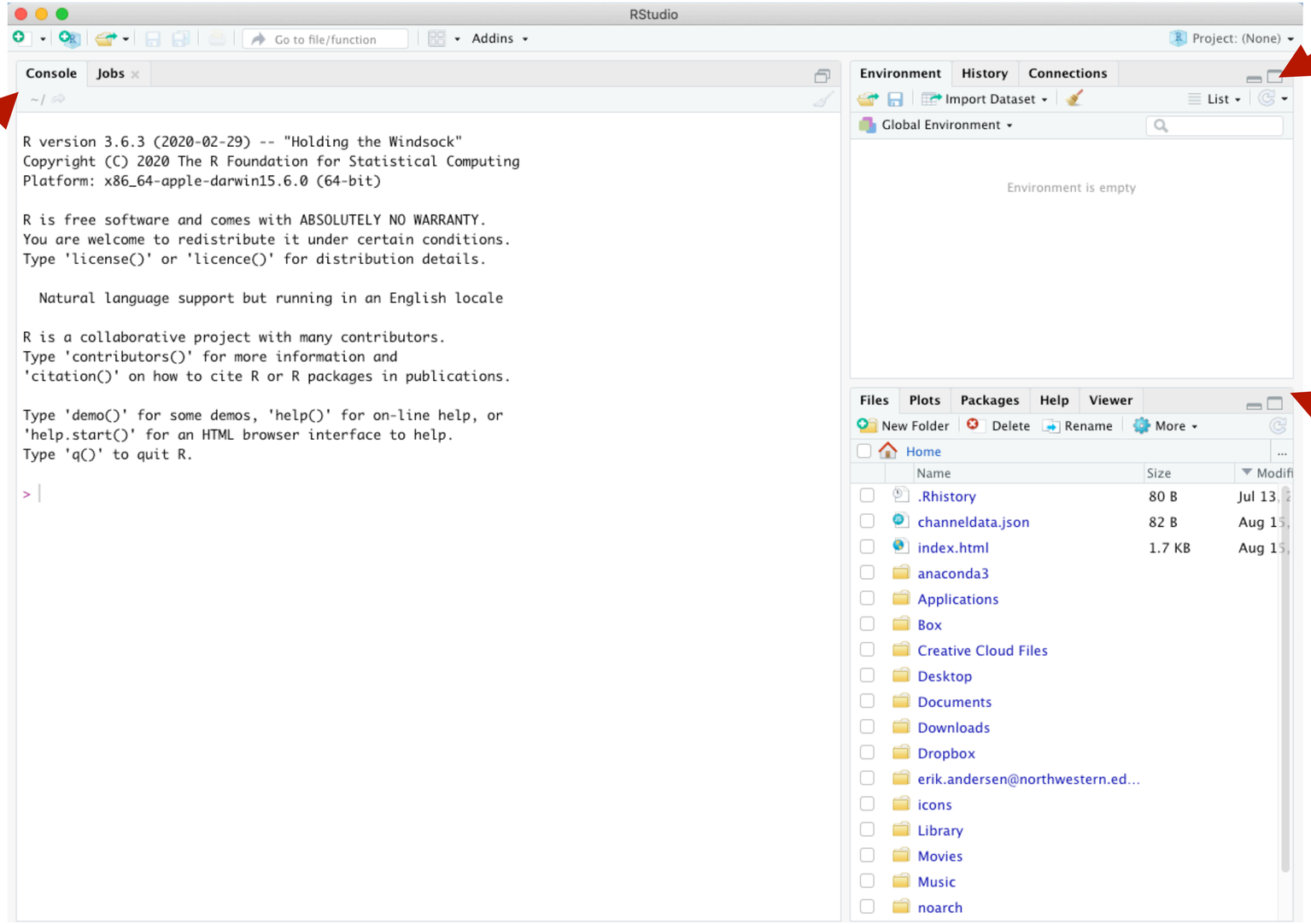

RStudio
Go to file/function
Addins
Project: (None)
Console
Jobs
R version 3.6.3 (2020-02-29) -- "Holding the Windsock"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to 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 R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
Name
Size
.Rhistory
80 B
channeldata.json
82 B
index.html
1.7 KB
anaconda3
Applications
Box
Creative Cloud Files
Desktop
Documents
Downloads
Dropbox
erik.andersen@northwestern.ed...
icons
Library
Movies
Music
noarch

Programming To-Dos:

1. Install both R and RStudio
2. Open RStudio. Familiarize yourself with the layout
3. Create a new R project in your respective folder
4. Open a new R script file
5. Install your first package!

RStudio

Console - shows commands that have been run.



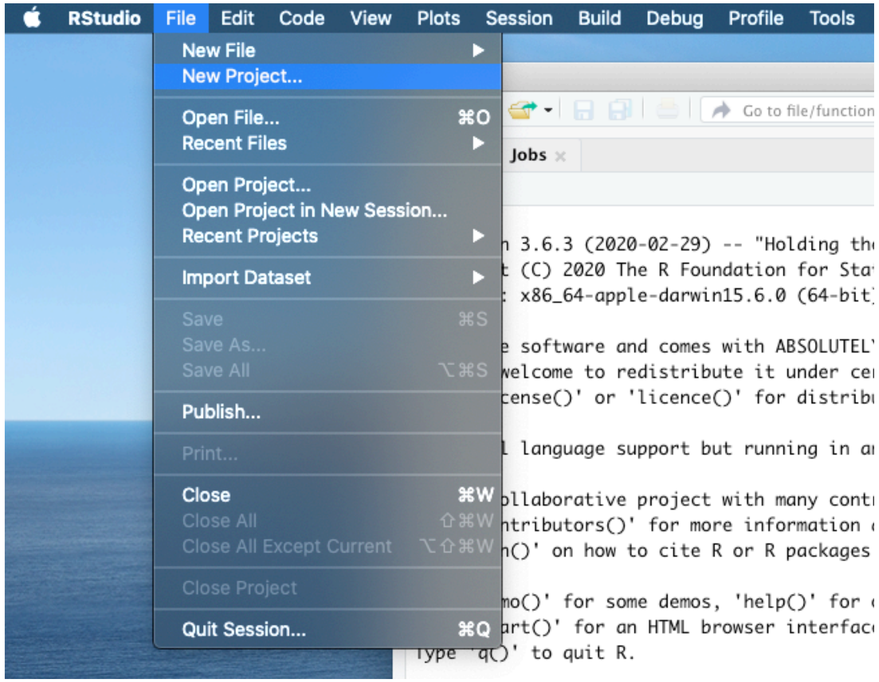
Workspace - shows items in your environment

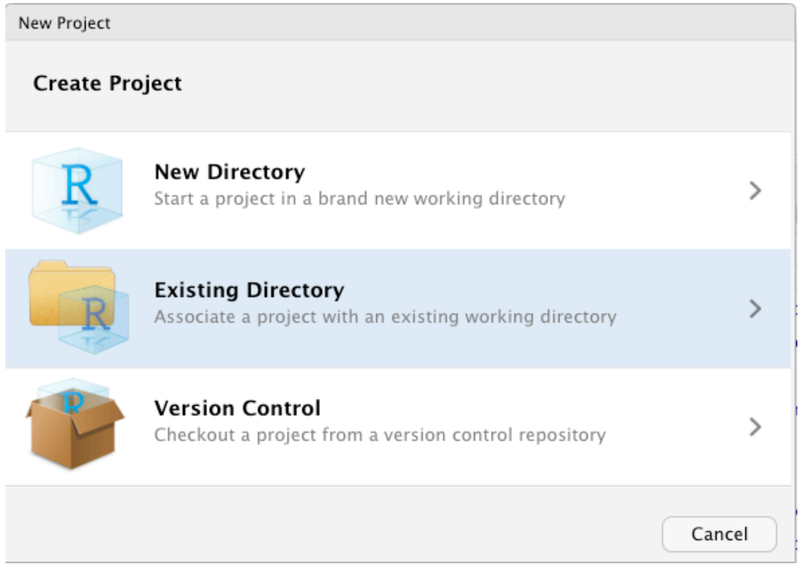
Access to files, view plots, see help documents

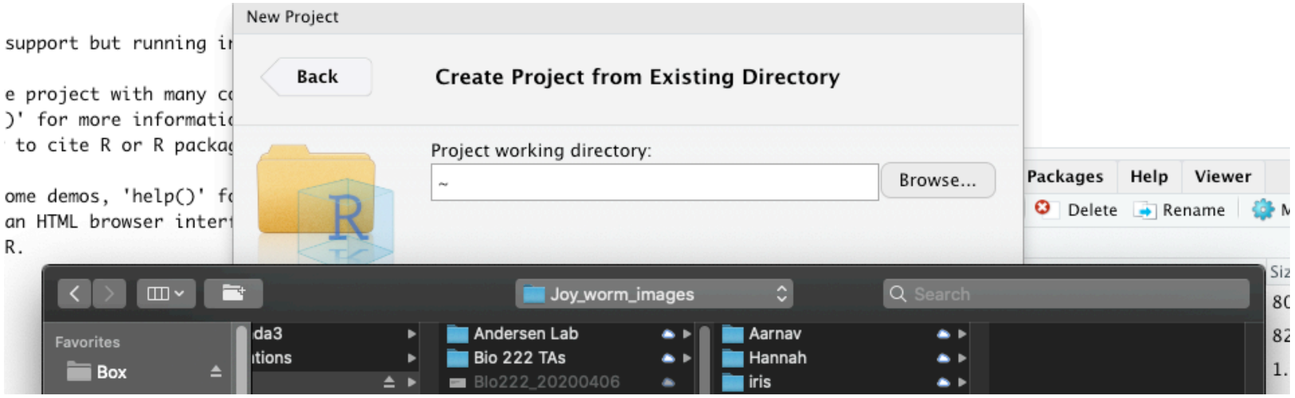
Programming To-Dos:

1. Install both R and RStudio
2. Open RStudio. Familiarize yourself with the layout
3. Create a new R project in your respective folder
4. Open a new R script file
5. Install your first package!

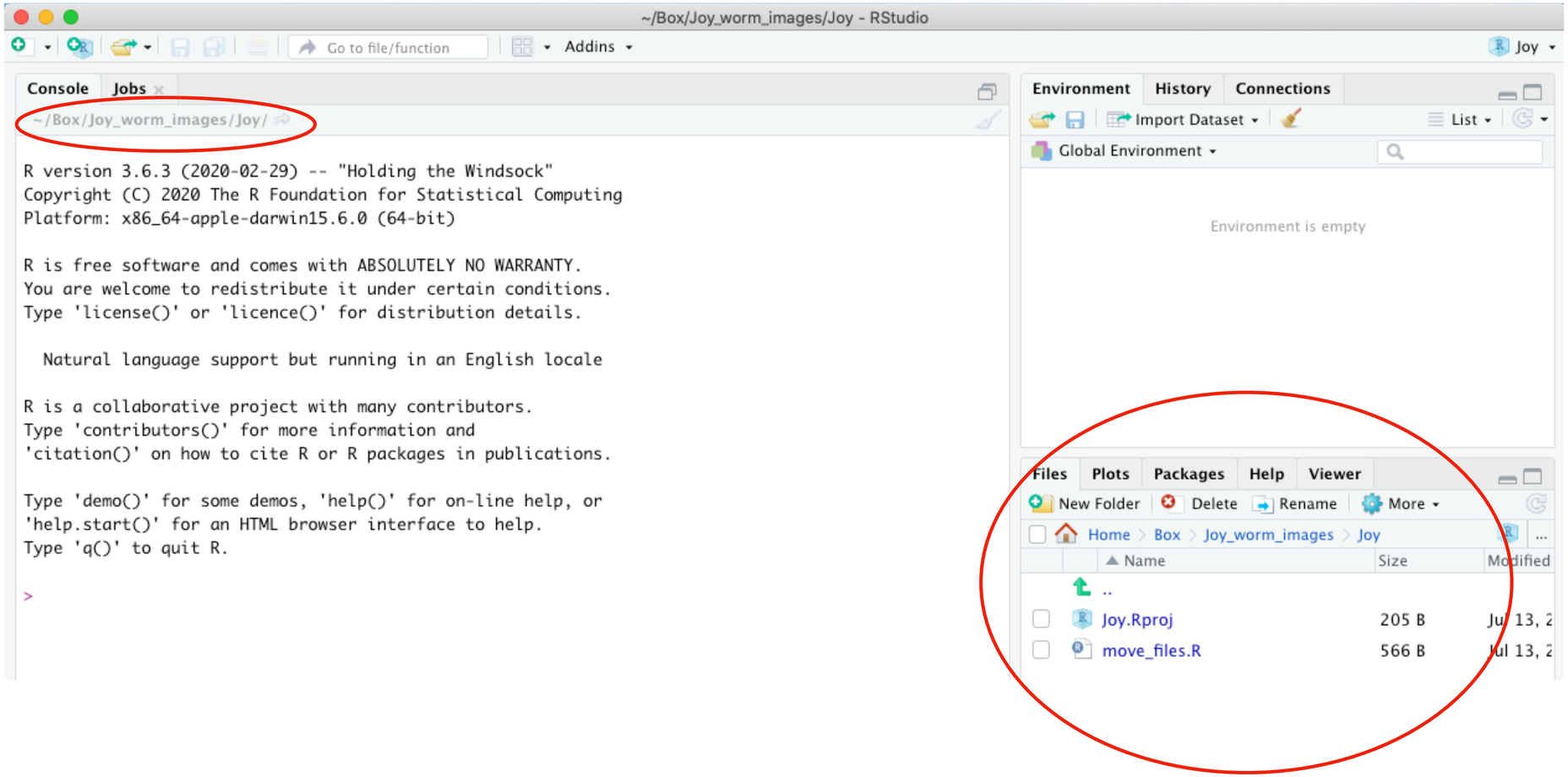
Create a new R project







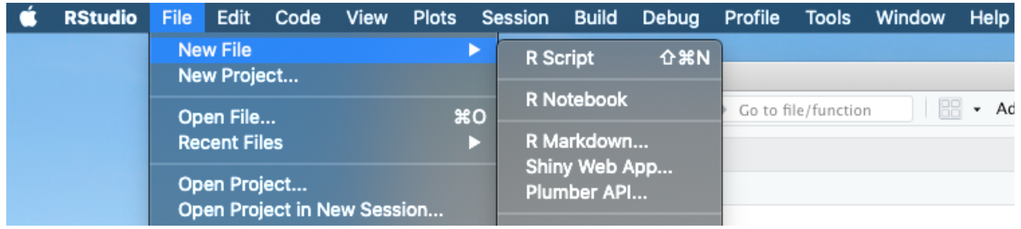
Create a new R project

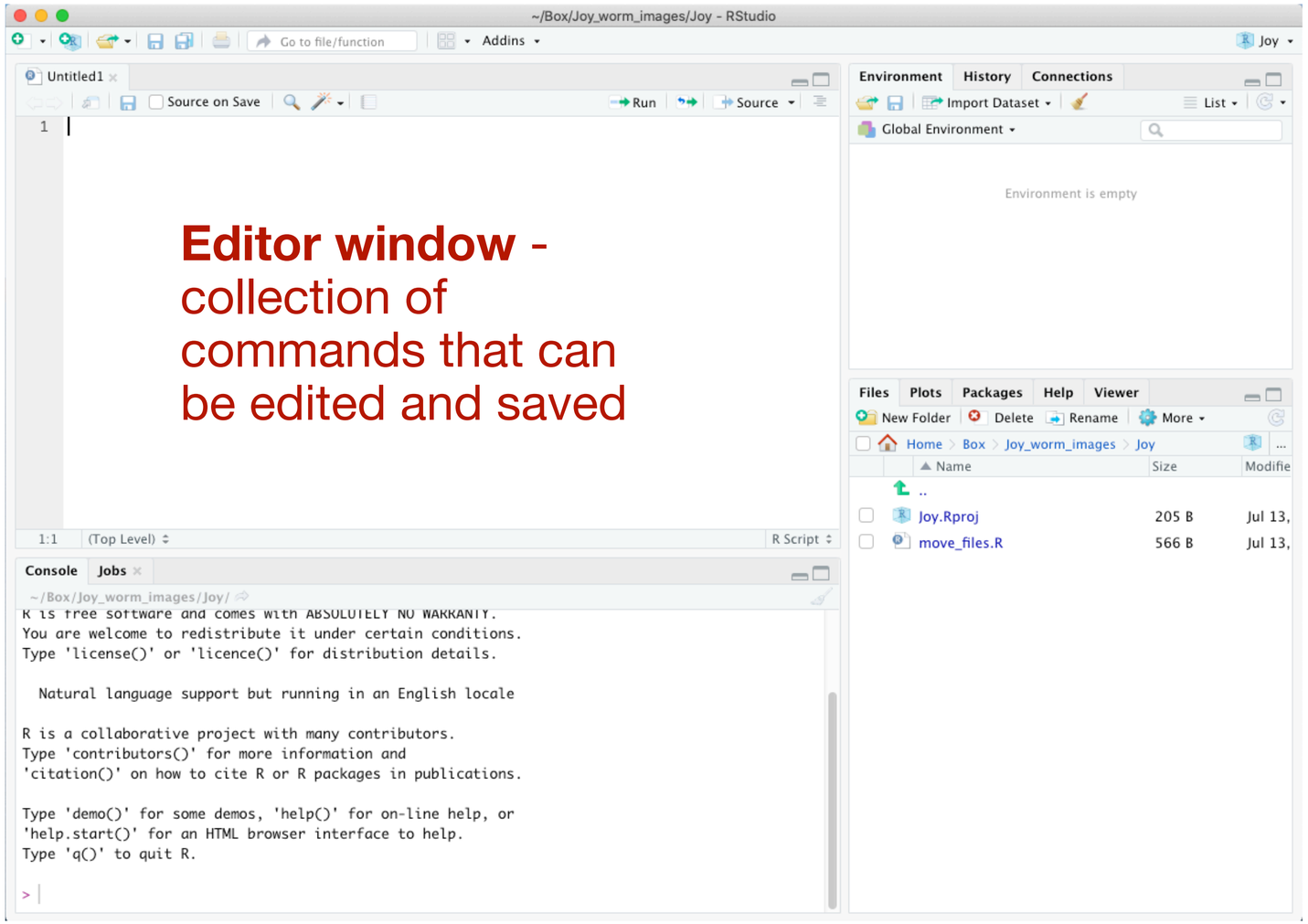


Programming To-Dos:

1. Install both R and RStudio
2. Open RStudio. Familiarize yourself with the layout
3. Create a new R project in your respective folder
4. Open a new R script file
5. Install your first package!

Open a new R script





Editor window - collection of commands that can be edited and saved

Programming To-Dos:

1. Install both R and RStudio
2. Open RStudio. Familiarize yourself with the layout
3. Create a new R project in your respective folder
4. Open a new R script file
5. Install your first package!

Install your first package

What is a package?

Packages are collections of R functions, data, and code in a well-defined format

	A	B	C	D	E
1	Row	Store	Product A	Product B	Product C
2	1	Store A	23	93	48
3	2	Store B	24	95	87
4	3	Store C	67	49	97
5	4	Store D	53	73	50
6	5	Store E	72	5	18
7	6	Store F	30	33	64
8	7	Store G	88	96	15
9	8	Store H	92	84	79
10	9	Store I	4	72	58
11	10	Store J	39	85	79
12	11	Store K	65	69	4
13	12	Store L	61	99	8
14	13	Store M	38	56	21
15	14	Store N	27	4	1
16	15	Store O	44	87	30
17	16	Store P	55	45	7
18	17	Store Q	23	13	11
19	[illegible]	R	[illegible]	8	49

$$\overline{x} = \frac{\sum_{i=1}^{N} x_i}{N}$$

$$\mathrm{SD} = \sqrt{\frac{\sum |x - \bar{x}|^2}{n}}$$

Files Plots Packages Help Viewer

mean

R: Arithmetic Mean Find in Topic

mean {base} R Documentation

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)

## Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)
```

Files Plots Packages Help Viewer

sd

R: Standard Deviation Find in Topic

sd {stats} R Documentation

Standard Deviation

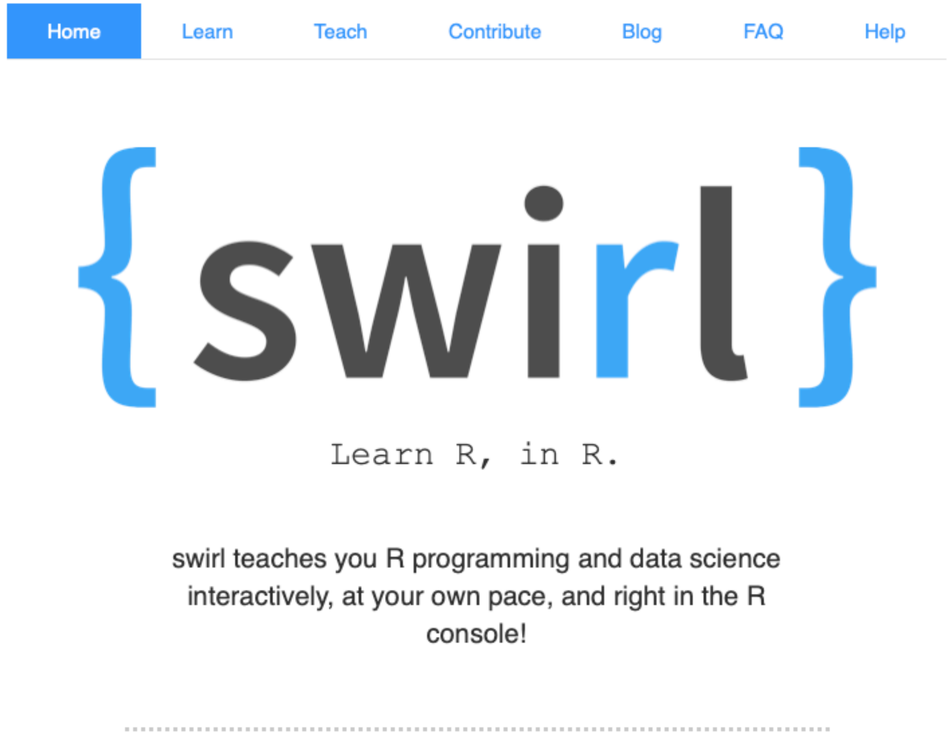
Description

This function computes the standard deviation of the values in x. If na.rm is TRUE then missing values are removed before computation proceeds.

Usage

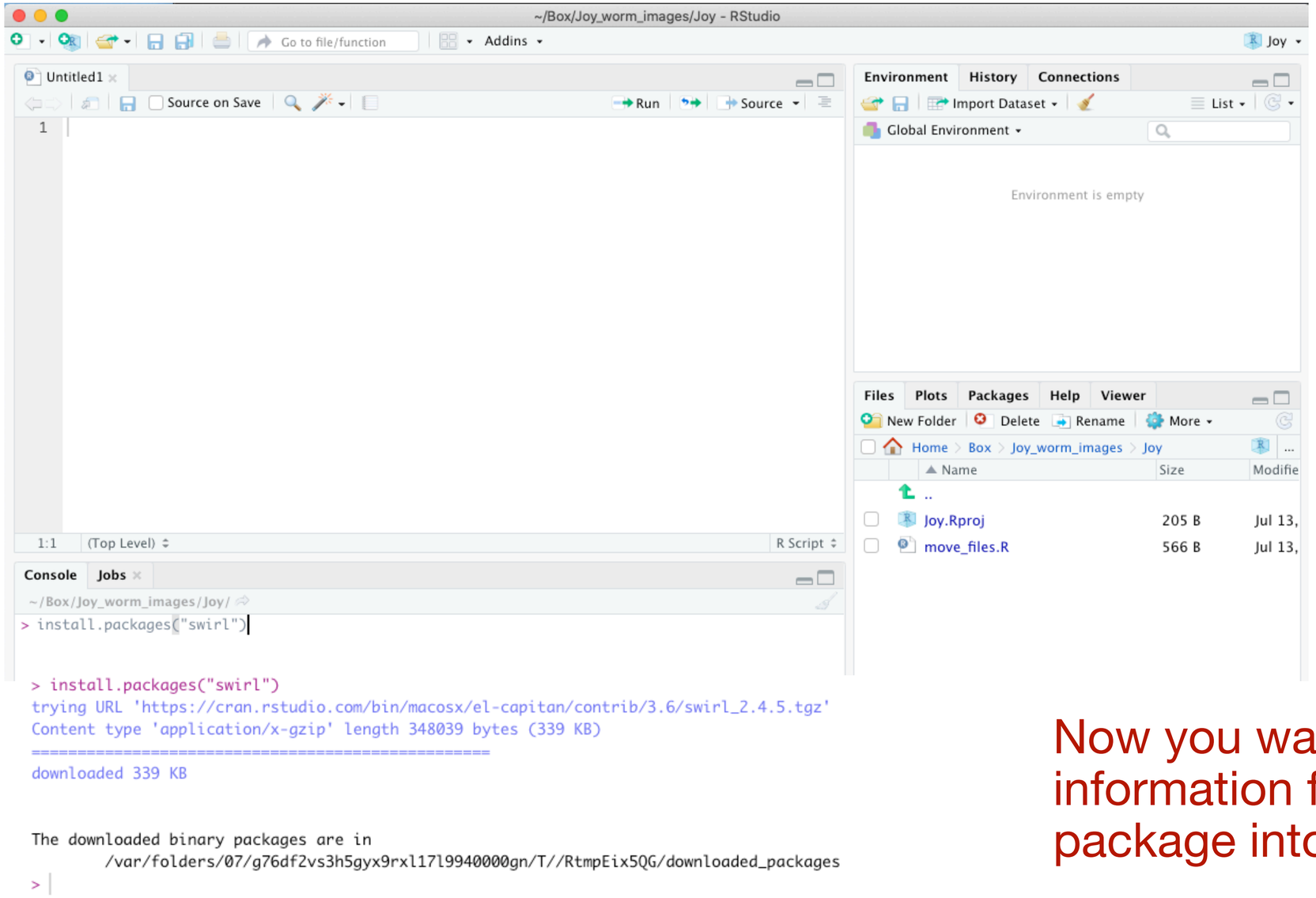
```
sd(x, na.rm = FALSE)
```

Install your first package



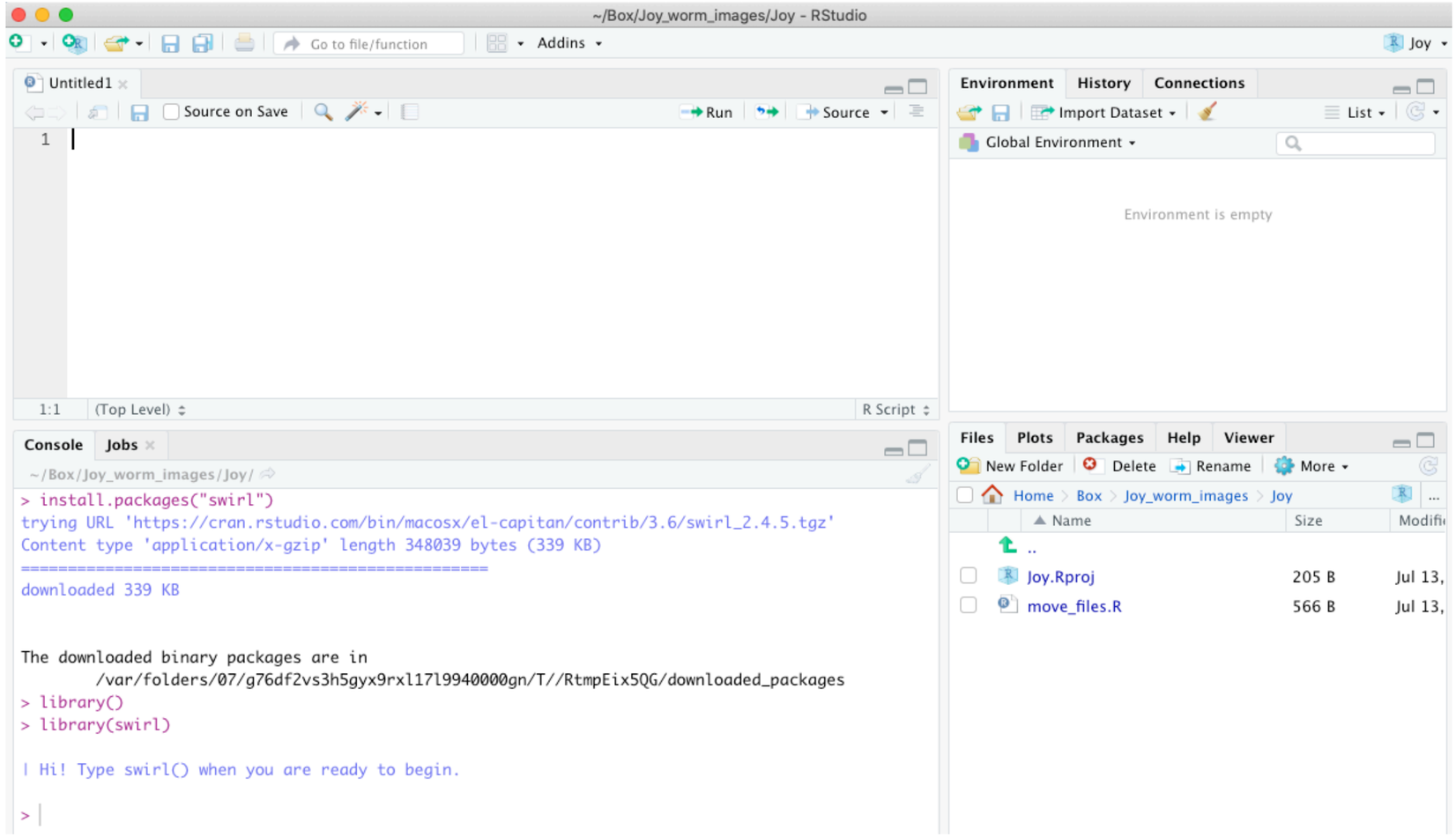
swirl is a software package for R that turns your console into an interactive learning environment.

Install your first package



Now you want to load the information from the package into your library…

Install your first package



Play around with swirl

~/Box/Joy_worm_images/Joy - RStudio

Go to file/function Addins

Source

Console Jobs

~/Box/Joy_worm_images/Joy/

```
| should press Enter when you are done reading and ready to continue.

...  <-- That's your cue to press Enter to continue

| Also, when you see 'ANSWER:', the R prompt (>), or when you are asked to select from
| a list, that means it's your turn to enter a response, then press Enter to continue.

Select 1, 2, or 3 and press Enter

1: Continue.
2: Proceed.
3: Let's get going!

Selection: 1

| You can exit swirl and return to the R prompt (>) at any time by pressing the Esc
| key. If you are already at the prompt, type bye() to exit and save your progress.
| When you exit properly, you'll see a short message letting you know you've done so.

| When you are at the R prompt (>):
| -- Typing skip() allows you to skip the current question.
| -- Typing play() lets you experiment with R on your own; swirl will ignore what you
| do...
| -- UNTIL you type nxt() which will regain swirl's attention.
| -- Typing bye() causes swirl to exit. Your progress will be saved.
| -- Typing main() returns you to swirl's main menu.
| -- Typing info() displays these options again.

| Let's get started!

...

| To begin, you must install a course. I can install a course for you from the internet, or I can send you to a
| web page (https://github.com/swirldev/swirl_courses) which will provide course options and directions for
| installing courses yourself. (If you are not connected to the internet, type 0 to exit.)

1: R Programming: The basics of programming in R
2: Regression Models: The basics of regression modeling in R
3: Statistical Inference: The basics of statistical inference in R
4: Exploratory Data Analysis: The basics of exploring data in R
5: Don't install anything for me. I'll do it myself.
```

Play around with swirl

```
| Please choose a course, or type 0 to exit swirl.

1: R Programming
2: Take me to the swirl course repository!

Selection: 1

| Please choose a lesson, or type 0 to return to course menu.

 1: Basic Building Blocks      2: Workspace and Files        3: Sequences of Numbers
 4: Vectors                    5: Missing Values             6: Subsetting Vectors
 7: Matrices and Data Frames   8: Logic                      9: Functions
10: lapply and sapply         11: vapply and tapply         12: Looking at Data
13: Simulation                14: Dates and Times           15: Base Graphics
```

I recommend trying 1 - 4.

If you wish to try the others feel free (but skip 10 & 11)

What's next?

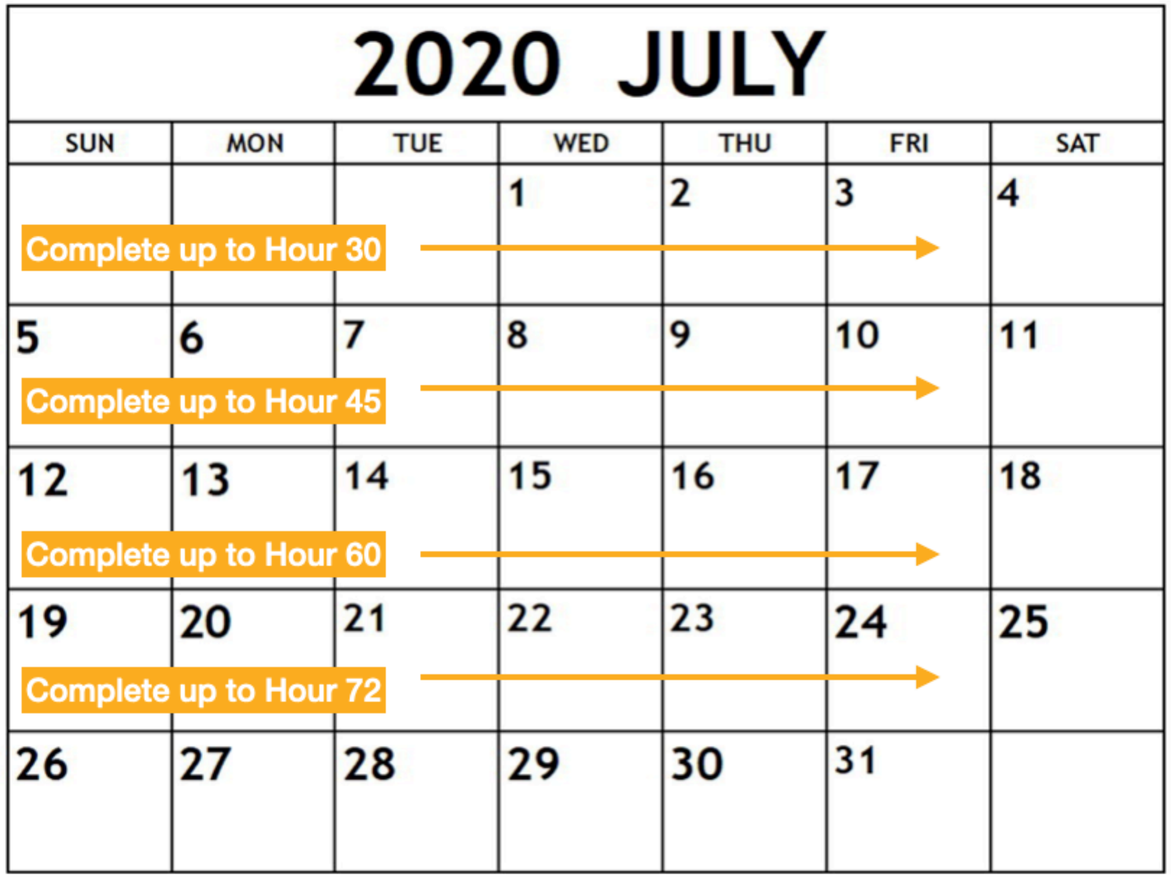
1. Read Data → July 21st

2. Tidy
3. Process

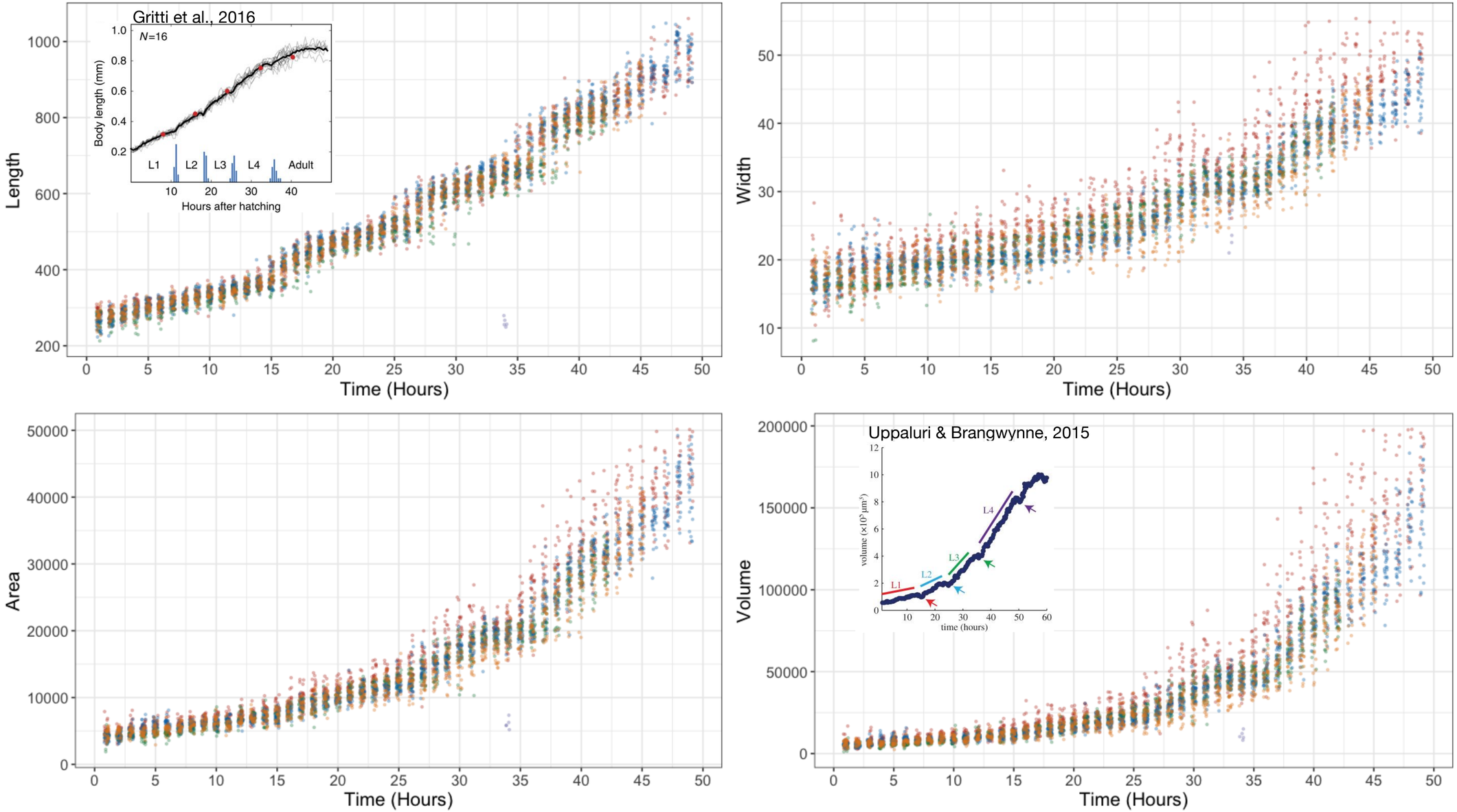
} July 28th

4. Plot
5. Present!

} August 4th



Sneak Peak!


Gritti et al., 2016
N=16
Body length (mm)
L1
L2
L3
L4
Adult
Hours after hatching
Length
Time (Hours)
Width
Time (Hours)
Area
Time (Hours)
Uppaluri & Brangwynne, 2015
volume (×10³ µm⁻³)
time (hours)
Volume
Time (Hours)

Intro to R week 2 To Dos:

1. What are basic data types in R?
2. What are different data structures used?
3. Tidyverse!
4. Start with an R script file
5. Get started (see Read_Data.html in my folder)

Basic Data Types in R

<u>Atomic vector types</u>

Atomic = only holds data of one type

- Character
- Numeric
  - Double
  - Integer
- Logical

```
> # Assign a number to the variable "y"
> y <- 2
> y
[1] 2
>
```

```
> # Assign a sequence of numbers into the variable "y"
> y <- 1:10
> y
 [1]  1  2  3  4  5  6  7  8  9 10
>
```

```
> view(y)
> class(y)
[1] "integer"
> str(y)
 int [1:10] 1 2 3 4 5 6 7 8 9 10
>
```

Basic Data Types in R

<u>Atomic vector types</u>

Atomic = only holds data of one type

Character

Numeric

- Double
- Integer

Logical

```
> y <- as.numeric(y)
> y
 [1]  1  2  3  4  5  6  7  8  9 10
> class(y)
[1] "numeric"
> str(y)
 num [1:10] 1 2 3 4 5 6 7 8 9 10
>
```

```
> y <- as.character(y)
> y
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
> class(y)
[1] "character"
> str(y)
 chr [1:10] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10"
>
```

```
> x <- c("Joy", "Hannah", "Jordan", "Iris", "Justine", "Izzy")
> x
[1] "Joy"     "Hannah"  "Jordan"  "Iris"    "Justine" "Izzy"
> class(x)
[1] "character"
> str(x)
 chr [1:6] "Joy" "Hannah" "Jordan" "Iris" "Justine" "Izzy"
>
```

Data Structures in R

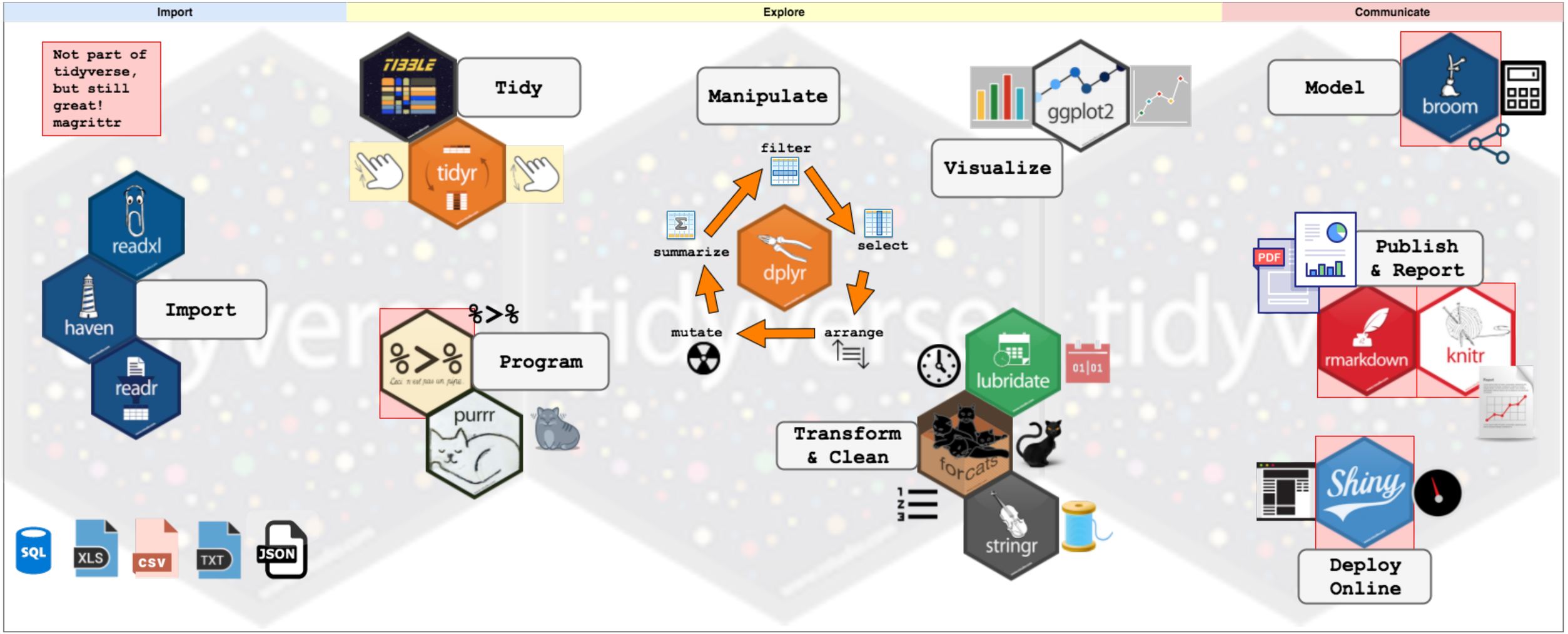
1. Vectors - one dimensional data sets of ONE data type

2. Data frames - two dimensional data sets of ANY data type

3. Lists - groups of vectors, data frames, or other lists

Install tidyverse package

Tidyverse is a collection of R packages
that work well together as part of a
larger data analysis pipeline.



Install tidyverse package

```
> install.packages("tidyverse")
trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.6/tidyverse_1.3.0.tgz'
Content type 'application/x-gzip' length 433010 bytes (422 KB)
==================================================
downloaded 422 KB


The downloaded binary packages are in
        /var/folders/07/g76df2vs3h5gyx9rxl17l9940000gn/T//RtmpqDlgln/downloaded_packages
> library(tidyverse)
── Attaching packages ─────────────────────────────────────────────────────────────────────────── tidyverse 1.3.0 ──
✓ ggplot2 3.3.2     ✓ purrr   0.3.4
✓ tibble  3.0.3     ✓ dplyr   1.0.0
✓ tidyr   1.1.0     ✓ stringr 1.4.0
✓ readr   1.3.1     ✓ forcats 0.5.0
── Conflicts ────────────────────────────────────────────────────────────────────────────── tidyverse_conflicts() ──
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

Install here package

Constructs paths to your project's files
(so you don't have to)

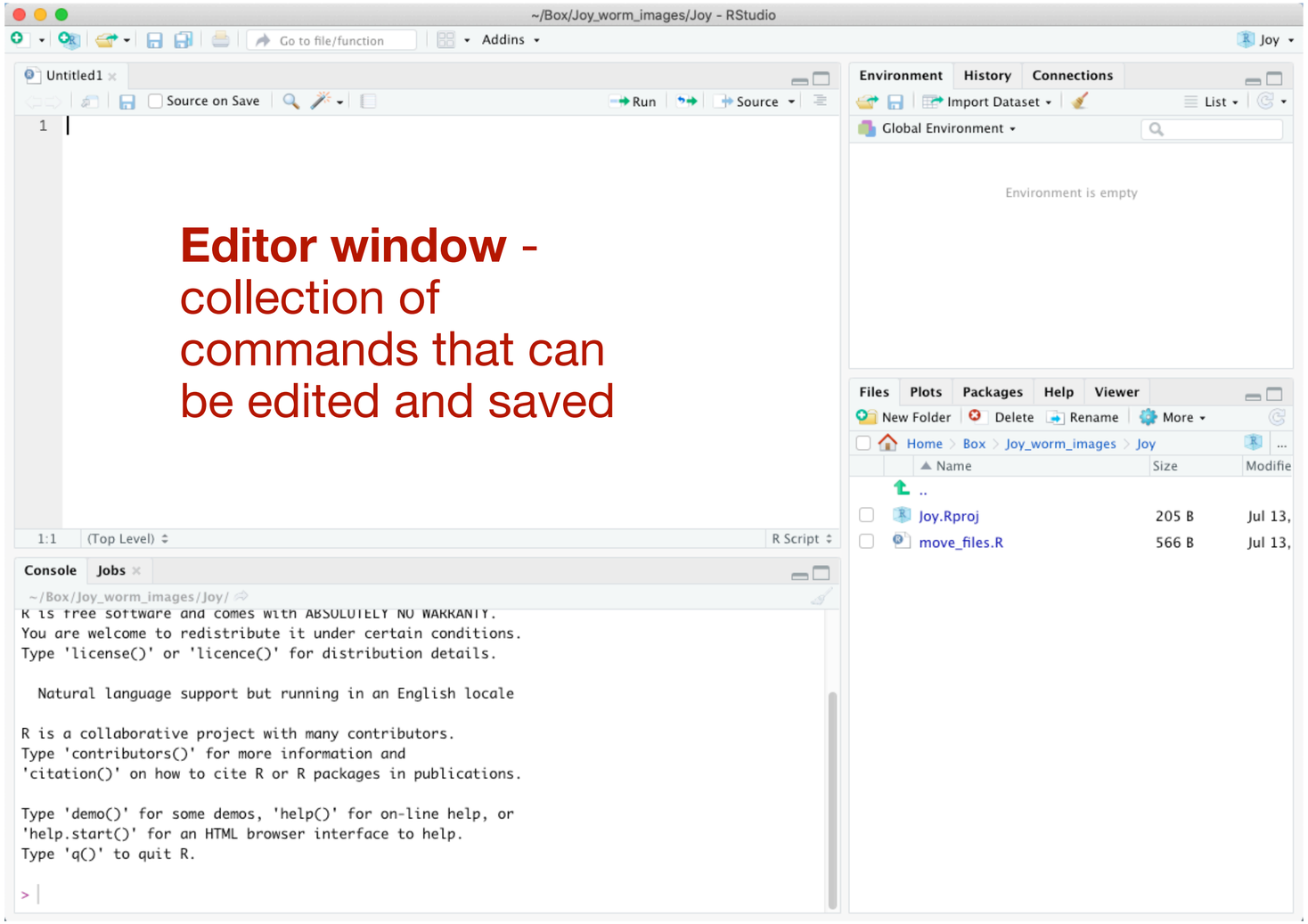
```
> install.packages("here")
trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.6/here_0.1.tgz'
Content type 'application/x-gzip' length 18187 bytes (17 KB)
==================================================
downloaded 17 KB


The downloaded binary packages are in
        /var/folders/07/g76df2vs3h5gyx9rxl17l9940000gn/T//RtmpqDlgln/downloaded_packages
> library(here)
here() starts at /Users/joy/Box/Joy_worm_images/Joy
>
```

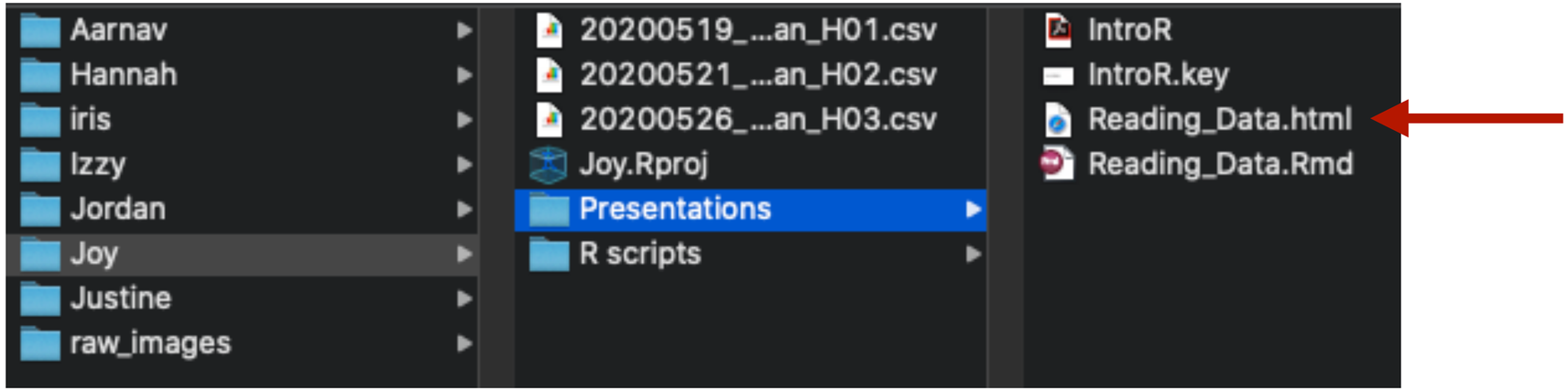
Open a new R script file

Last week we talked briefly about R script files…
we will be using these to do our scripting.

See slide 14 for a reminder of how to open up a new R script file.



Get started reading in data



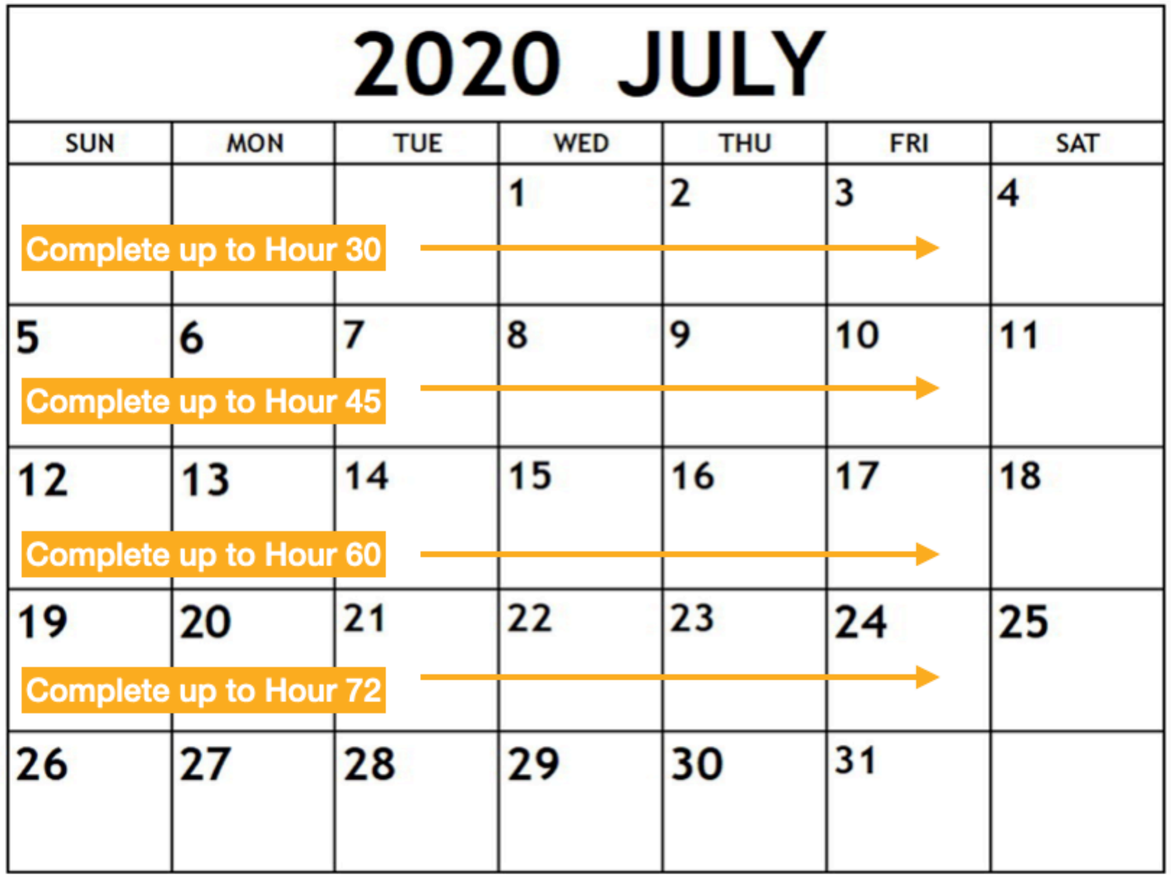
What's next?

1. Read Data → Today
2. Tidy
3. Process

} July 28th

4. Plot
5. Present!

} August 4th



Intro to R week 3 To Dos:

1. What is tidy data?
2. Messy vs. Clean Data
3. Data Formats: long vs wide
4. Manipulate Data with **dplyr**
5. Tidy Data with **tidyr**
6. Get started (see Tidy&Process.html in my folder)

What is tidy data?

- Tidy data are easy to manipulate and visualize
- Each column is a variable (sample, replicate, hour)
- Each row is an observation

It might seem like the data in some columns are repetitive

Messy vs. Clean Data

First 10 rows of Izzy's data

X1	Label	Area	Angle	Length
1	p01-growth-H01-2X_F01.TIF	79	0.000	78.675
2	p01-growth-H01-2X_F01.TIF	7	-69.444	5.696
3	p01-growth-H01-2X_F01.TIF	83	0.000	82.100
4	p01-growth-H01-2X_F01.TIF	6	29.982	5.003
5	p01-growth-H01-2X_F01.TIF	65	0.000	64.593
6	p01-growth-H01-2X_F01.TIF	5	0.000	4.868
7	p01-growth-H01-2X_F01.TIF	86	0.000	85.628
8	p01-growth-H01-2X_F01.TIF	4	36.870	2.500
9	p01-growth-H01-2X_F01.TIF	81	0.000	80.141
10	p01-growth-H01-2X_F01.TIF	6	59.349	5.231

Rules for tidy data

- Each **variable** must have its own **column**

Messy vs. Clean Data

First 10 rows of Izzy's data

	X1	Label	Area	Angle	Length
1		p01-growth-H01-2X_F01.TIF	79	0.000	78.675
2		p01-growth-H01-2X_F01.TIF	7	-69.444	5.696
3		p01-growth-H01-2X_F01.TIF	83	0.000	82.100
4		p01-growth-H01-2X_F01.TIF	6	29.982	5.003
5		p01-growth-H01-2X_F01.TIF	65	0.000	64.593
6		p01-growth-H01-2X_F01.TIF	5	0.000	4.868
7		p01-growth-H01-2X_F01.TIF	86	0.000	85.628
8		p01-growth-H01-2X_F01.TIF	4	36.870	2.500
9		p01-growth-H01-2X_F01.TIF	81	0.000	80.141
10		p01-growth-H01-2X_F01.TIF	6	59.349	5.231

Rules for tidy data

- Each **variable** must have its own **column**
- Each **observation** must have its own **row**

Messy vs. Clean Data

First 10 rows of Izzy's data

	X1	Label	Area	Angle	Length
1		p01-growth-H01-2X_F01.TIF	79	0.000	78.675
2		p01-growth-H01-2X_F01.TIF	7	-69.444	5.696
3		p01-growth-H01-2X_F01.TIF	83	0.000	82.100
4		p01-growth-H01-2X_F01.TIF	6	29.982	5.003
5		p01-growth-H01-2X_F01.TIF	65	0.000	64.593
6		p01-growth-H01-2X_F01.TIF	5	0.000	4.868
7		p01-growth-H01-2X_F01.TIF	86	0.000	85.628
8		p01-growth-H01-2X_F01.TIF	4	36.870	2.500
9		p01-growth-H01-2X_F01.TIF	81	0.000	80.141
10		p01-growth-H01-2X_F01.TIF	6	59.349	5.231

Rules for tidy data

- Each **variable** must have its own **column**
- Each **observation** must have its own **row**
- Each **value** must have its own **cell**

Data Formats: Long vs. Wide

Wide

Long

Data Formats: Long vs. Wide

Wide

name	midterm_1	midterm_2	midterm_3
samantha	72	80	81
taylor	91	92	90
kelsey	83	74	90
ramona	65	71	75

Long

name	midterm	score
samantha	midterm_1	72
samantha	midterm_2	80
samantha	midterm_3	81
taylor	midterm_1	91
taylor	midterm_2	92
taylor	midterm_3	90
kelsey	midterm_1	83
kelsey	midterm_2	74
kelsey	midterm_3	90
ramona	midterm_1	65
ramona	midterm_2	71
ramona	midterm_3	75

Which is better? - depends on your analysis

Data Formats: Long vs. Wide

Wide

name	midterm_1	midterm_2	midterm_3
samantha	72	80	81
taylor	91	92	90
kelsey	83	74	90
ramona	65	71	75

Q: Find the average of each student's midterms

average = mean(midterm_1, midterm_2, midterm_3)

Wide

name	midterm_1	midterm_2	midterm_3	average
samantha	72	80	81	77.6
taylor	91	92	90	91
kelsey	83	74	90	82.3
ramona	65	71	75	70.3

Imagine you have 100 midterms to average… this would be difficult to code

Data Formats: Long vs. Wide

First 10 rows of Izzy's data

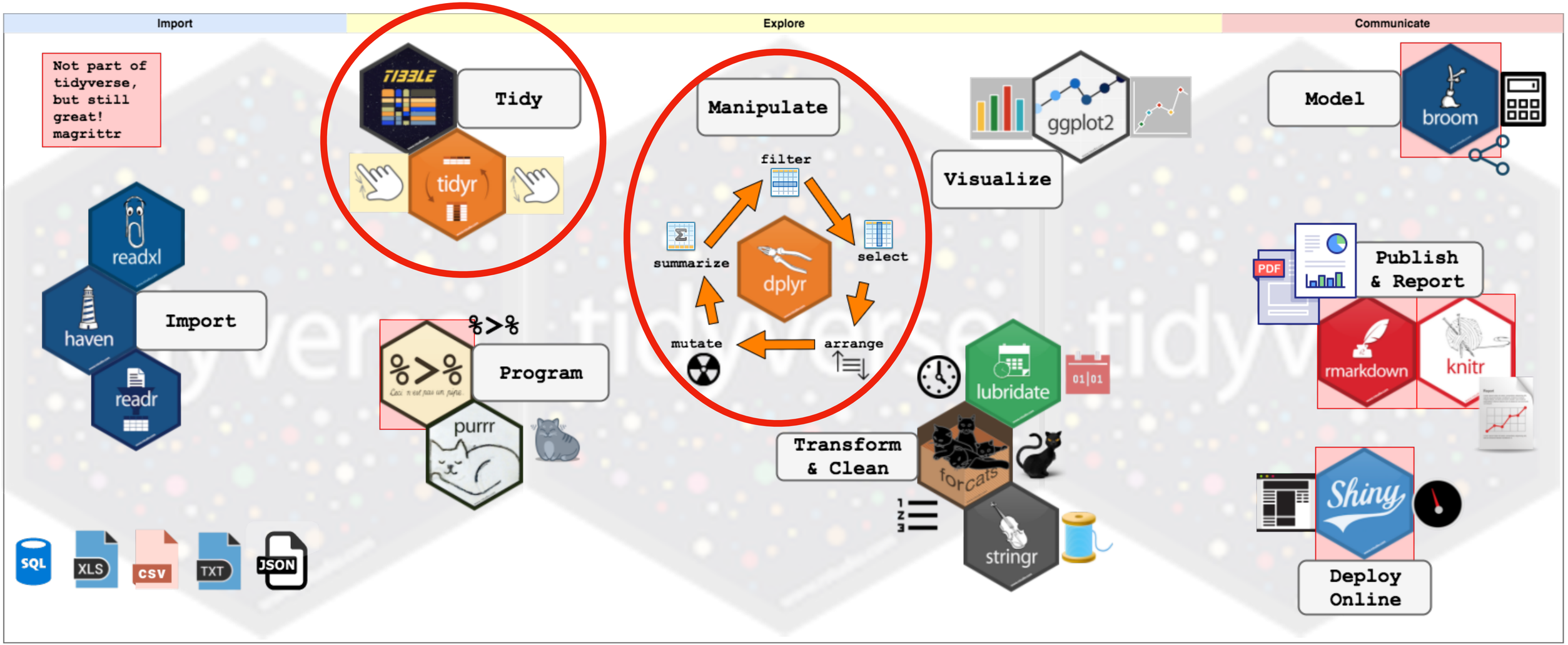
X1	Label	Area	Angle	Length
1	p01-growth-H01-2X_F01.TIF	79	0.000	78.675
2	p01-growth-H01-2X_F01.TIF	7	-69.444	5.696
3	p01-growth-H01-2X_F01.TIF	83	0.000	82.100
4	p01-growth-H01-2X_F01.TIF	6	29.982	5.003
5	p01-growth-H01-2X_F01.TIF	65	0.000	64.593
6	p01-growth-H01-2X_F01.TIF	5	0.000	4.868
7	p01-growth-H01-2X_F01.TIF	86	0.000	85.628
8	p01-growth-H01-2X_F01.TIF	4	36.870	2.500
9	p01-growth-H01-2X_F01.TIF	81	0.000	80.141
10	p01-growth-H01-2X_F01.TIF	6	59.349	5.231

p01 - Plate
growth - Experiment
H01 - Hour
2X - Magnification
F01 - Well

Contains both Length and Width measurements

Tidyverse

Tidyverse is a collection of R packages
that work well together as part of a
larger data analysis pipeline.



Manipulate data with **dplyr**

Collection of functions as **verbs** to easily describe what you want to do with your data

- `mutate()` to add new (or change existing) columns
- `select()` to keep columns based on names
- `group_by()` to group rows by columns

1

`dplyr::mutate()`

`dplyr::mutate()` to add new (or change existing) columns

Goal: Add a column indicating student

`dplyr::mutate(`dataframe, new_column = expression`)`

First 10 rows of Izzy's data

X1	Label	Area	Angle	Length
1	p01-growth-H01-2X_F01.TIF	79	0.000	78.675
2	p01-growth-H01-2X_F01.TIF	7	-69.444	5.696
3	p01-growth-H01-2X_F01.TIF	83	0.000	82.100
4	p01-growth-H01-2X_F01.TIF	6	29.982	5.003
5	p01-growth-H01-2X_F01.TIF	65	0.000	64.593
6	p01-growth-H01-2X_F01.TIF	5	0.000	4.868
7	p01-growth-H01-2X_F01.TIF	86	0.000	85.628
8	p01-growth-H01-2X_F01.TIF	4	36.870	2.500
9	p01-growth-H01-2X_F01.TIF	81	0.000	80.141
10	p01-growth-H01-2X_F01.TIF	6	59.349	5.231

dplyr::mutate()

Goal: Add a column indicating student

```
dplyr::mutate(worms, Student = "Izzy")
```

First 10 rows of Izzy's data

X1	Label	Area	Angle	Length	Student
1	p01-growth-H01-2X_F01.TIF	79	0.000	78.675	Izzy
2	p01-growth-H01-2X_F01.TIF	7	-69.444	5.696	Izzy
3	p01-growth-H01-2X_F01.TIF	83	0.000	82.100	Izzy
4	p01-growth-H01-2X_F01.TIF	6	29.982	5.003	Izzy
5	p01-growth-H01-2X_F01.TIF	65	0.000	64.593	Izzy
6	p01-growth-H01-2X_F01.TIF	5	0.000	4.868	Izzy
7	p01-growth-H01-2X_F01.TIF	86	0.000	85.628	Izzy
8	p01-growth-H01-2X_F01.TIF	4	36.870	2.500	Izzy
9	p01-growth-H01-2X_F01.TIF	81	0.000	80.141	Izzy
10	p01-growth-H01-2X_F01.TIF	6	59.349	5.231	Izzy

★ We can change existing columns if we use the same name

dplyr::mutate()

★ We can change existing columns if we use the same name

Goal: Change column Area so all entries are 0

```
dplyr::mutate(worms, Area = 0)
```

First 10 rows of Izzy's data

X1	Label	Area	Angle	Length	Student
1	p01-growth-H01-2X_F01.TIF	0	0.000	78.675	Izzy
2	p01-growth-H01-2X_F01.TIF	0	-69.444	5.696	Izzy
3	p01-growth-H01-2X_F01.TIF	0	0.000	82.100	Izzy
4	p01-growth-H01-2X_F01.TIF	0	29.982	5.003	Izzy
5	p01-growth-H01-2X_F01.TIF	0	0.000	64.593	Izzy
6	p01-growth-H01-2X_F01.TIF	0	0.000	4.868	Izzy
7	p01-growth-H01-2X_F01.TIF	0	0.000	85.628	Izzy
8	p01-growth-H01-2X_F01.TIF	0	36.870	2.500	Izzy
9	p01-growth-H01-2X_F01.TIF	0	0.000	80.141	Izzy
10	p01-growth-H01-2X_F01.TIF	0	59.349	5.231	Izzy

2

`dplyr::select()`

`dplyr::select()` to keep columns based on names

Goal: Select and reorder data to include columns - X1, Label, Student, Length

```
dplyr::select(dataframe, columns_to_keep)
```

First 10 rows of Izzy's data

X1	Label	Area	Angle	Length	Student
1	p01-growth-H01-2X_F01.TIF	0	0.000	78.675	Izzy
2	p01-growth-H01-2X_F01.TIF	0	-69.444	5.696	Izzy
3	p01-growth-H01-2X_F01.TIF	0	0.000	82.100	Izzy
4	p01-growth-H01-2X_F01.TIF	0	29.982	5.003	Izzy
5	p01-growth-H01-2X_F01.TIF	0	0.000	64.593	Izzy
6	p01-growth-H01-2X_F01.TIF	0	0.000	4.868	Izzy
7	p01-growth-H01-2X_F01.TIF	0	0.000	85.628	Izzy
8	p01-growth-H01-2X_F01.TIF	0	36.870	2.500	Izzy
9	p01-growth-H01-2X_F01.TIF	0	0.000	80.141	Izzy
10	p01-growth-H01-2X_F01.TIF	0	59.349	5.231	Izzy

★ Reorder columns with select too!

dplyr::select()

Goal: Select and reorder data to include columns - X1, Label, Student, Length

```
dplyr::select(worms, c(X1, Label, Student, Length))
```

First 10 rows of Izzy's data

X1	Label	Student	Length
1	p01-growth-H01-2X_F01.TIF	Izzy	78.675
2	p01-growth-H01-2X_F01.TIF	Izzy	5.696
3	p01-growth-H01-2X_F01.TIF	Izzy	82.100
4	p01-growth-H01-2X_F01.TIF	Izzy	5.003
5	p01-growth-H01-2X_F01.TIF	Izzy	64.593
6	p01-growth-H01-2X_F01.TIF	Izzy	4.868
7	p01-growth-H01-2X_F01.TIF	Izzy	85.628
8	p01-growth-H01-2X_F01.TIF	Izzy	2.500
9	p01-growth-H01-2X_F01.TIF	Izzy	80.141
10	p01-growth-H01-2X_F01.TIF	Izzy	5.231

3

`dplyr::group_by()`

`dplyr::group_by()` to group rows by columns

★ Doesn't change how the data looks, changes how the data interacts with other dplyr verbs

First 10 rows of Izzy's data

X1	Label	Student	Length
1	p01-growth-H01-2X_F01.TIF	Izzy	78.675
2	p01-growth-H01-2X_F01.TIF	Izzy	5.696
3	p01-growth-H01-2X_F01.TIF	Izzy	82.100
4	p01-growth-H01-2X_F01.TIF	Izzy	5.003
5	p01-growth-H01-2X_F01.TIF	Izzy	64.593
6	p01-growth-H01-2X_F01.TIF	Izzy	4.868
7	p01-growth-H01-2X_F01.TIF	Izzy	85.628
8	p01-growth-H01-2X_F01.TIF	Izzy	2.500
9	p01-growth-H01-2X_F01.TIF	Izzy	80.141
10	p01-growth-H01-2X_F01.TIF	Izzy	5.231

Tidy data with **tidyr**

Collection of functions as **verbs** to easily “tidy” your data

- `separate()` to split one column into two
- `pivot_wider()` to expand one column into multiple

1

`tidyr::separate()`

`tidyr::separate()` to split one column into two

Goal: Separate Label column by "_"

```
dplyr::select(dataframe, old_column, into = c(new_columns), sep = separator)
```

First 10 rows of Izzy's data

X1	Label	Student	Length
1	p01-growth-H01-2X_F01.TIF	Izzy	78.675
2	p01-growth-H01-2X_F01.TIF	Izzy	5.696
3	p01-growth-H01-2X_F01.TIF	Izzy	82.100
4	p01-growth-H01-2X_F01.TIF	Izzy	5.003
5	p01-growth-H01-2X_F01.TIF	Izzy	64.593
6	p01-growth-H01-2X_F01.TIF	Izzy	4.868
7	p01-growth-H01-2X_F01.TIF	Izzy	85.628
8	p01-growth-H01-2X_F01.TIF	Izzy	2.500
9	p01-growth-H01-2X_F01.TIF	Izzy	80.141
10	p01-growth-H01-2X_F01.TIF	Izzy	5.231

tidyr::separate()

Goal: Separate Label column by "_"

```
dplyr::select(worms, Label, into = c("Info", "Well"), sep = "_")
```

First 10 rows of Izzy's data

X1	Info	Well	Student	Length
1	p01-growth-H01-2X	F01.TIF	Izzy	78.675
2	p01-growth-H01-2X	F01.TIF	Izzy	5.696
3	p01-growth-H01-2X	F01.TIF	Izzy	82.100
4	p01-growth-H01-2X	F01.TIF	Izzy	5.003
5	p01-growth-H01-2X	F01.TIF	Izzy	64.593
6	p01-growth-H01-2X	F01.TIF	Izzy	4.868
7	p01-growth-H01-2X	F01.TIF	Izzy	85.628
8	p01-growth-H01-2X	F01.TIF	Izzy	2.500
9	p01-growth-H01-2X	F01.TIF	Izzy	80.141
10	p01-growth-H01-2X	F01.TIF	Izzy	5.231

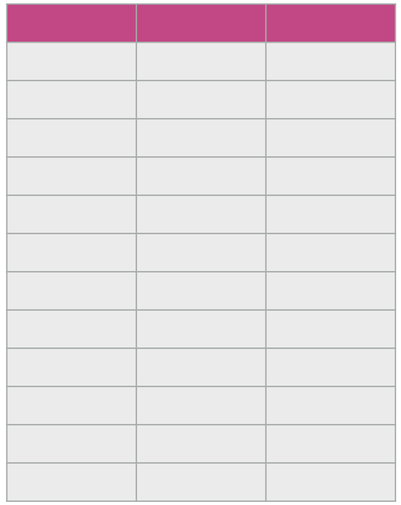
2

`tidyr::pivot_wider()`

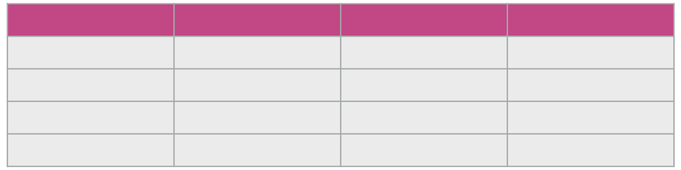
`tidyr::pivot_wider()` to expand one column to multiple

```
dplyr::pivot_wider(dataframe, names_from = new_column_name, values_from = values)
```

Long



Wide



`tidyr::pivot_wider()`

`tidyr::pivot_wider()` to expand one column to multiple

```
dplyr::pivot_wider(dataframe, names_from = midterm, values_from = score)
```

name	midterm	score
samantha	midterm_1	72
samantha	midterm_2	80
samantha	midterm_3	81
taylor	midterm_1	91
taylor	midterm_2	92
taylor	midterm_3	90
kelsey	midterm_1	83
kelsey	midterm_2	74
kelsey	midterm_3	90
ramona	midterm_1	65
ramona	midterm_2	71
ramona	midterm_3	75

name	midterm_1	midterm_2	midterm_3
samantha	72	80	81
taylor	91	92	90
kelsey	83	74	90
ramona	65	71	75

Piping in Tidyverse

`pipe (%>%)`

takes output of left side and makes it input of right side

`dplyr::mutate()` to add new (or change existing) columns

`dplyr::select()` to keep columns based on names

`dplyr::group_by()` to group rows by columns

`tidyr::separate()` to split one column into two

`tidyr::pivot_wider()` to expand one column into multiple

pipe (%>%)

takes output of left side and makes it input of right side

```
dplyr::mutate(worms, Student = "Izzy")
```

+

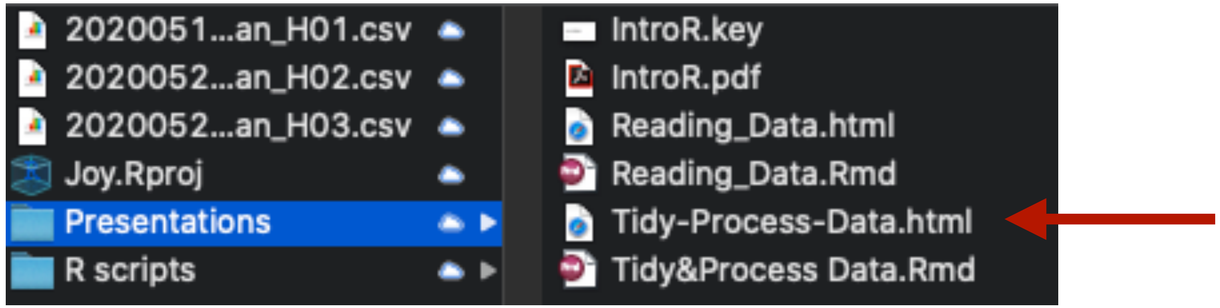
```
dplyr::select(worms, c(X1, Label, Student, Length))
```

```
worms %>%
dplyr::mutate(Student = "Izzy") %>%
dplyr::select(X1, Label, Student, Length)
```

First 10 rows of Izzy's data

X1	Label	Student	Length
1	p01-growth-H01-2X_F01.TIF	Izzy	78.675
2	p01-growth-H01-2X_F01.TIF	Izzy	5.696
3	p01-growth-H01-2X_F01.TIF	Izzy	82.100
4	p01-growth-H01-2X_F01.TIF	Izzy	5.003
5	p01-growth-H01-2X_F01.TIF	Izzy	64.593
6	p01-growth-H01-2X_F01.TIF	Izzy	4.868
7	p01-growth-H01-2X_F01.TIF	Izzy	85.628
8	p01-growth-H01-2X_F01.TIF	Izzy	2.500
9	p01-growth-H01-2X_F01.TIF	Izzy	80.141
10	p01-growth-H01-2X_F01.TIF	Izzy	5.231

Get started tidying and processing!

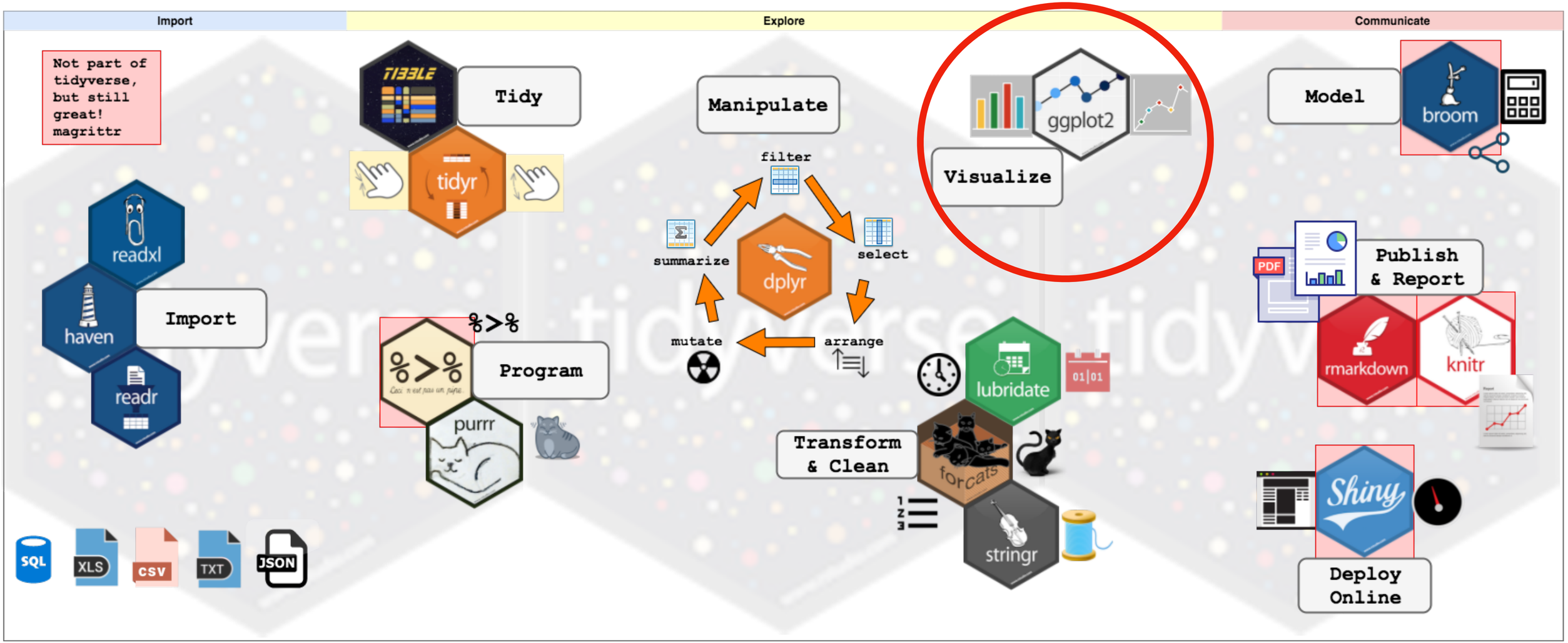


Intro to R week 4

1. ggplot2: the grammar of graphics

2. Get started (see Visualize.html in my folder)

Visualization with ggplot2



ggplot2::ggplot

ggplot2 is the package name

ggplot is the function name

but ggplot generally refers to both

ggplot2: the grammar of graphics

```
> rnorm(1000)
   [1]  0.7414303363 -0.9383127854 -0.5898356239 -1.4879381203 -0.1659582252  0.4690914210 -0.3598660699
  [11]  2.6412618078  0.2321025525 -0.1327265269  1.5190948454  0.9066730669  0.8596798670  1.4650258834
  [21]  0.3458054361 -0.4886197680  1.3973476592 -1.5638681539  1.2853007445  0.4101364885  0.2294735247
  [31]  0.4139866417 -0.6954449569  0.8041125473  0.5535330655 -0.4694144802  1.7690122917  0.4707698513
  [41] -0.4594998205  0.4043386537 -1.6870132729 -0.1942175306  1.1583540288 -0.0002630832 -0.1468545234
  [51] -0.4410132764  0.7364134275  2.0252124219 -1.4500256740  1.9125350969 -0.2343692491  1.3286159719
  [61]  1.1314206797 -0.9113800142  0.1240687944 -0.3060999484 -0.4709176421 -0.1122752856 -0.5401285711
  [71] -0.0686987744 -0.1373026497  0.6094719385 -1.4732265606  0.7573958380 -0.7515556914 -1.2857906361
  [81] -0.8857107791 -0.4069381352 -2.1758080948 -0.3569778668 -0.0397559943 -0.0961785023  0.6472138988
  [91] -0.8830039848 -2.0658918174  2.2363978861 -0.9000721943  1.1227886790  2.1469963330 -1.0971182540
 [101]  0.8612006384 -1.0684987091  1.5397207327 -0.0174112748  0.6287091546 -0.9850152543  1.4317789228
 [111] -0.9610323091 -0.8214297129  0.0698531890 -0.2544790671  0.9626996188  1.4312750227  1.1144196341
 [121]  0.8893473243 -1.2105287954 -1.2804874114 -1.5417165424 -0.5225043177 -0.2443883469  1.0395231050
 [131] -0.0216148381  1.0670464559 -1.0937062759 -0.3949936928  0.6399457290  0.3473726551 -0.5487464459
 [141]  2.1042373099 -0.8215960512  1.1647203780 -0.5018804363 -0.6276899976 -0.8121978140 -1.9868618662
 [151] -2.5904304497  0.5526988025 -0.3580881297 -0.3931144287 -0.6494195785 -0.1096485904  0.0678612489
 [161]  1.6343875443 -0.0683924766  1.2130360802 -0.6313426788  0.9838639622 -0.4797304977  0.1817758260
 [171]  0.0695651300  1.0314607326 -2.0653772732 -0.0865188406 -1.1631547204  0.5729574962 -1.2640545629
 [181] -0.9050088656  0.2930384939 -0.2051316675  0.9764933512 -0.2243143242 -0.9517134217 -0.3218631511
 [191] -0.1991329532 -0.0923899862 -1.9904200615 -1.3877169486 -0.7618046746  0.2040072200  1.9060898324
```

Data

x y z

color

Aesthetics

aesthetics are things we see on the graph — shapes, positions, colors, etc.

points boxplot

line

Geometric objects

In most cases you start with ggplot(), supply a dataset and aesthetic mapping (with aes()). You then add on layers (like geom_point())

How do we map data to aesthetics?

Data

var1	var2	var3	var4
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b

Aesthetics

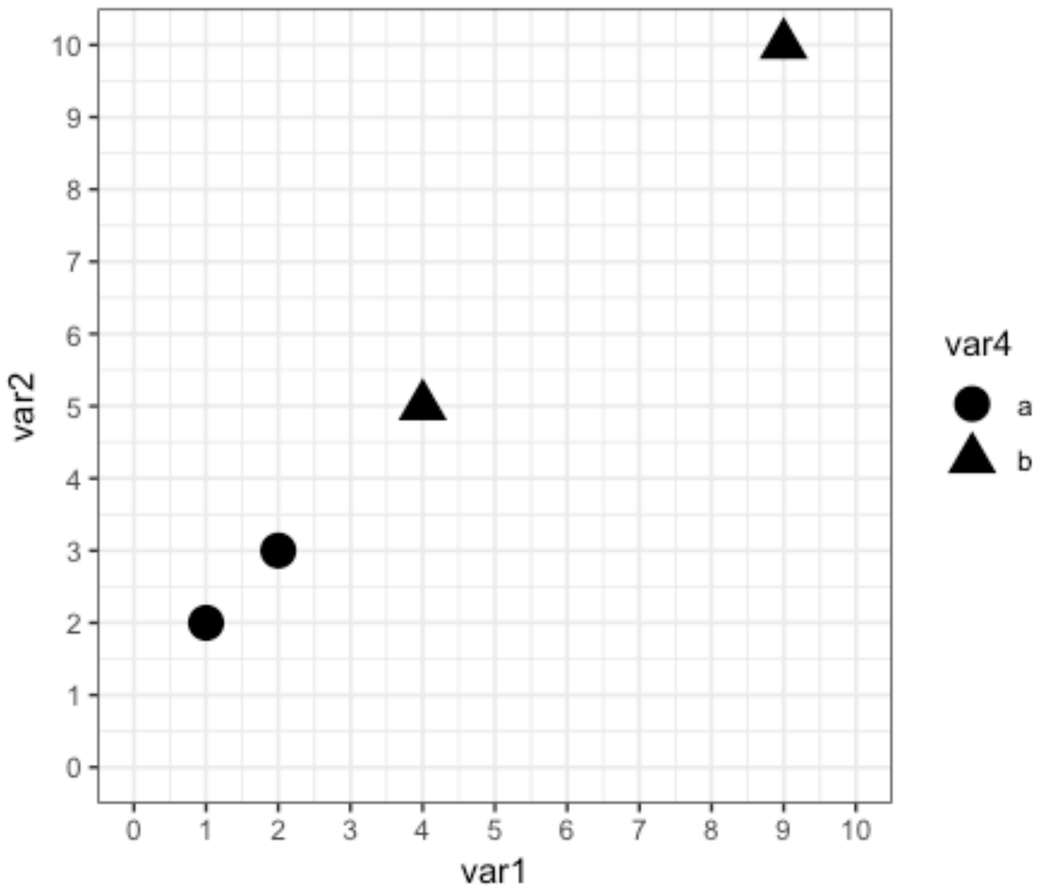
x	y	var3	shape
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b

How do we map data to aesthetics?

Aesthetics

x	y	var3	shape
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b

Geometric Object



```
ggplot(data=values) + aes(x=var1, y=var2, shape=var4) + geom_point()
```

Data

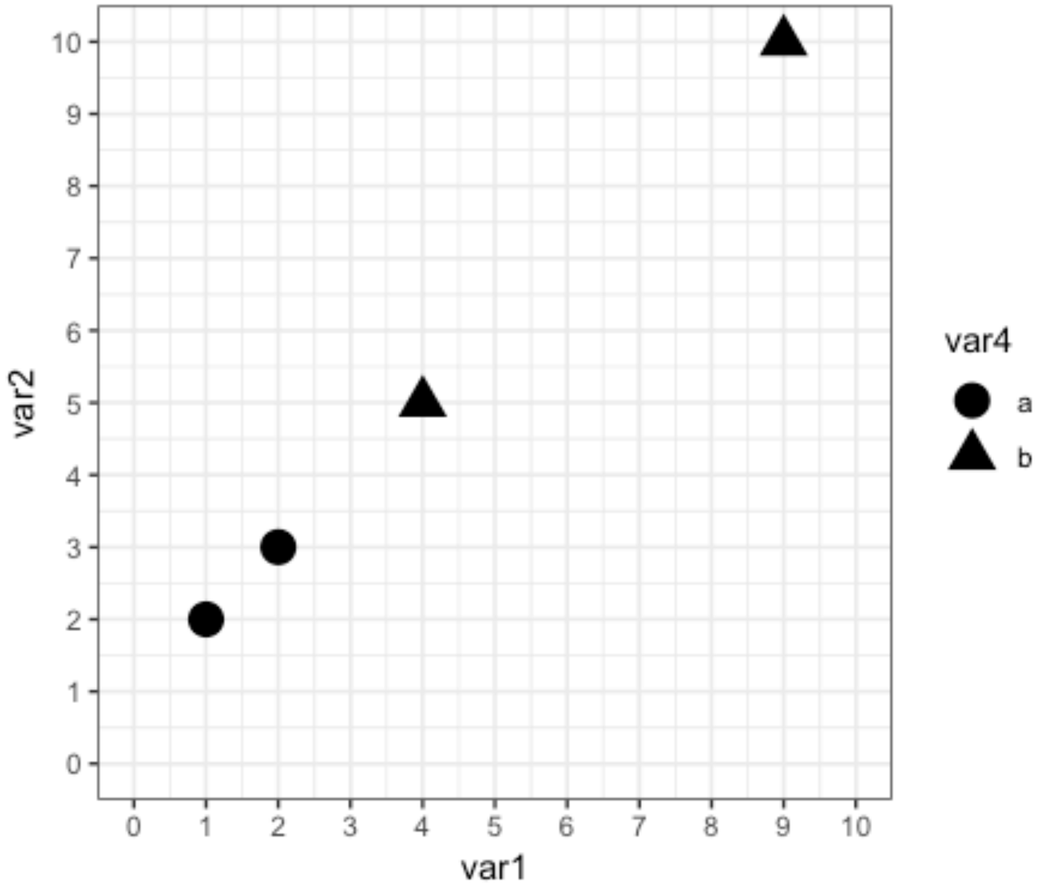
Add a layer or component to the graph

Aesthetics

Geometric object

Specifies the type of graph

var1	var2	var3	var4
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b



Common Aesthetics

position (x, y)

size

color

shape

fill

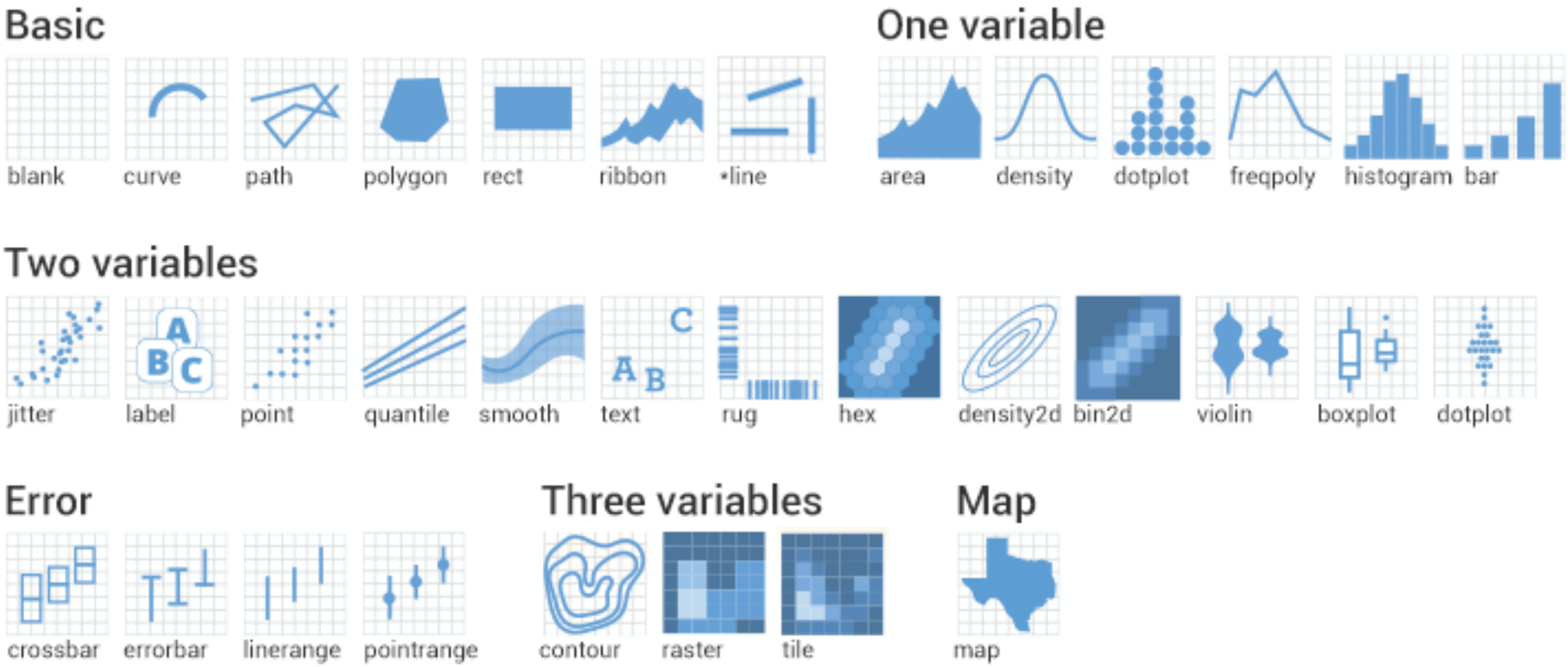
line type

transparency (alpha)

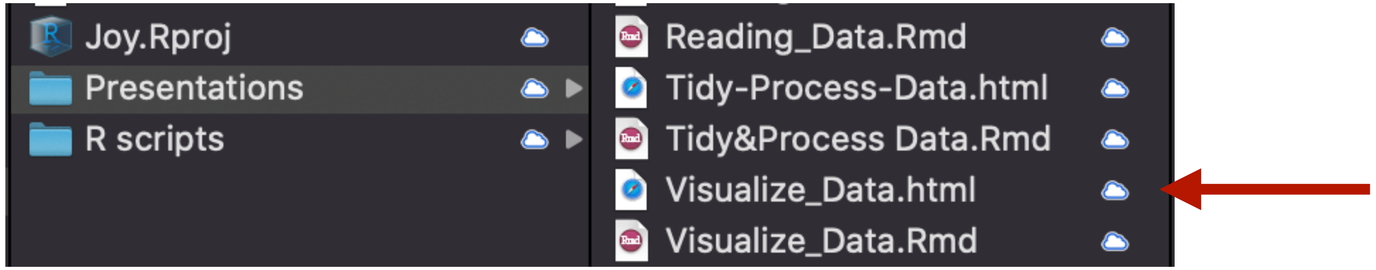
What aesthetics are relevant are determined by the plot type (ie. Geometric Object)

Geometric Shapes (geom)

`geom_...`



Get started plotting



First 4 rows of data

Plate	Experiment	Hour	Magnification	Well	Animal	Length	Width	Radius	Volume
p01	growth	01	2X	B01	1	264.9288	12.76309	6.381544	3124.370
p01	growth	01	2X	B01	2	252.9924	16.80116	8.400582	5170.208
p01	growth	01	2X	B01	3	276.7070	16.73200	8.365998	5608.381
p01	growth	01	2X	B01	4	231.9127	13.76108	6.880539	3179.445

For more on data viz