

WEEK 2

R BASICS AND R QUARTO

TODAY'S TOPICS

- A few more R and RStudio tips (projects, style)
- R Packages
- Intro to the “tidyverse”
- Pipes
- R Markdown: present and share your work in a report
- Paths and files

EFFECTIVE USE OF



RStudio[®]

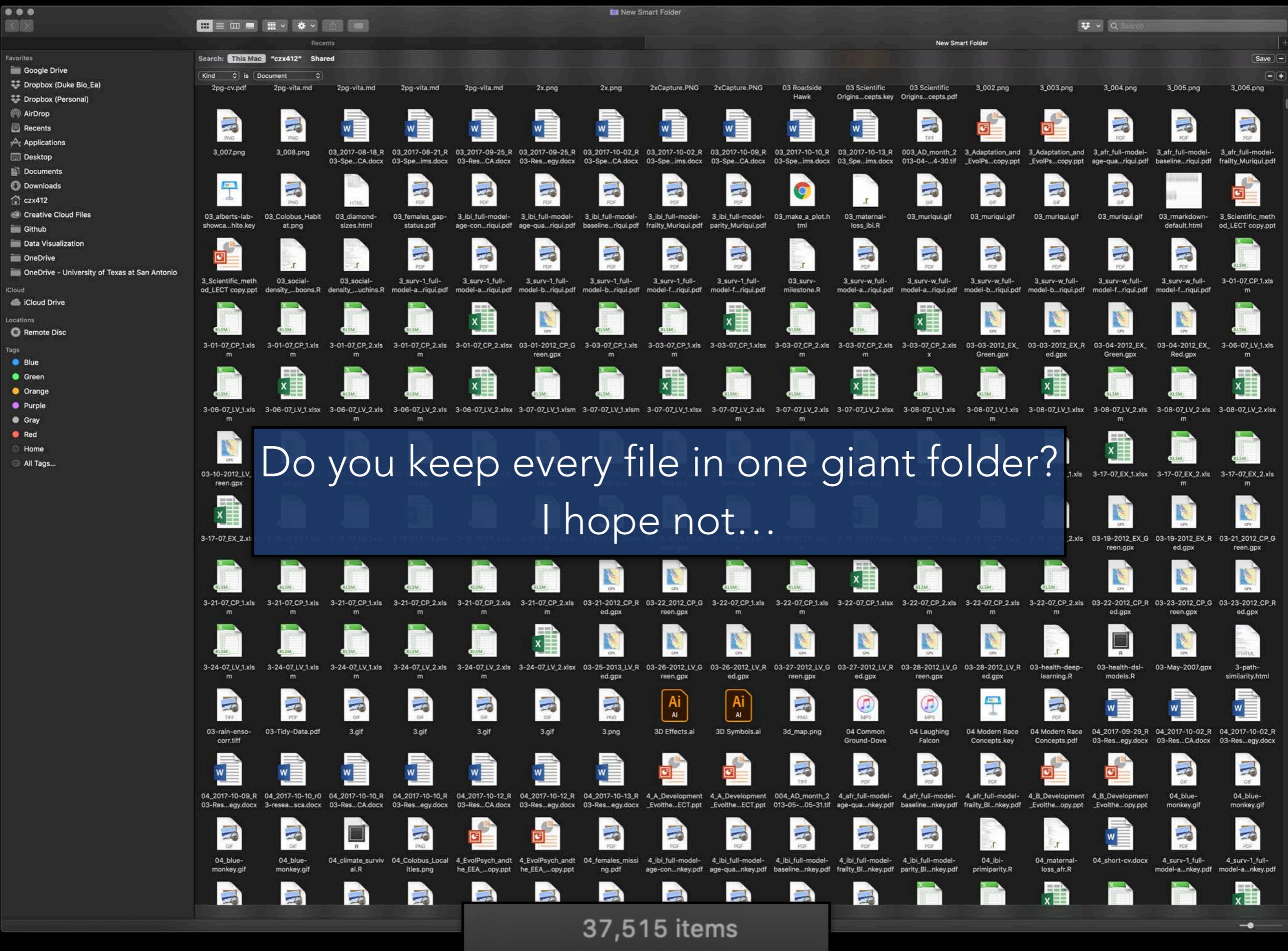
- Last time:
 - Work in the RStudio app, not the R app
 - Tab for autocompletion and helpful tips
 - Panes: Source, Console, Environment, Viewer

EFFECTIVE USE OF



Studio[®]

- Use RStudio to divide your work into distinct **Projects**



EFFECTIVE USE OF



- An RStudio Project is associated with one **folder** on your computer (called the “working directory”)
 - R will save files to and open files from this folder
 - Use a different project for any distinct “unit” of your work
 - Examples: analysis for one manuscript, thesis chapter, or class project
- Tip: for now, when creating a new Project always use a new empty folder

YOUR TURN



- Create a new Project called “Data Viz Activities”
- Create a new R script file called “hello.R”
- In hello.R, use the `print()` function to print the word "Hello!" to the console.
`print("Hello!")`
- Save the file
- Source the file (i.e., run the entire script) using Cmd/Ctrl + Shift + S
- Close the Project
- Re-open the “Data Viz Activities” project.

5:00

EFFECTIVE USE OF



- A project remembers the state of your “working environment” and keeps it separate from that of other projects.
 - Which objects you have created in R
 - Which files are open
 - A history of commands that you have used
- When you reopen a project, you can restore the workspace in same condition that you left it, but I don't recommend it.

BASIC R: IMPORTANT REMINDERS

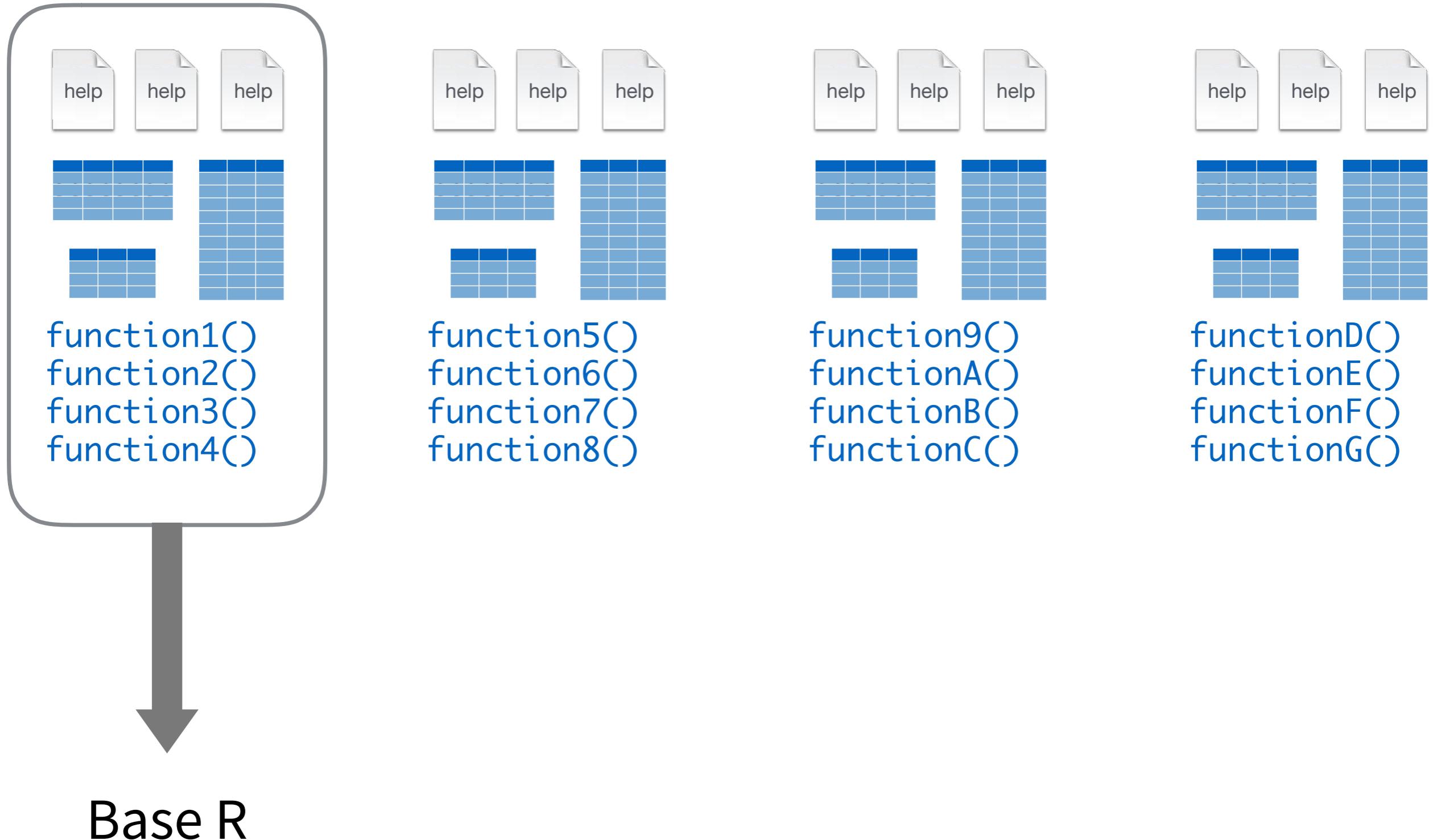
- In R everything has a name
 - Aim for short, meaningful names
 - Style guide recommends names in all lowercase, with words separated by underscores (e.g., `my_name`).
- You do things using functions and operators
 - Example functions: `c()`, `mean()`, `summary()`
 - Example operators: `+`, `←`
 - Tip: for assignment (`←`), use RStudio shortcut ⌘ + - in macOS, Alt + - in Windows

BASIC R: IMPORTANT REMINDERS

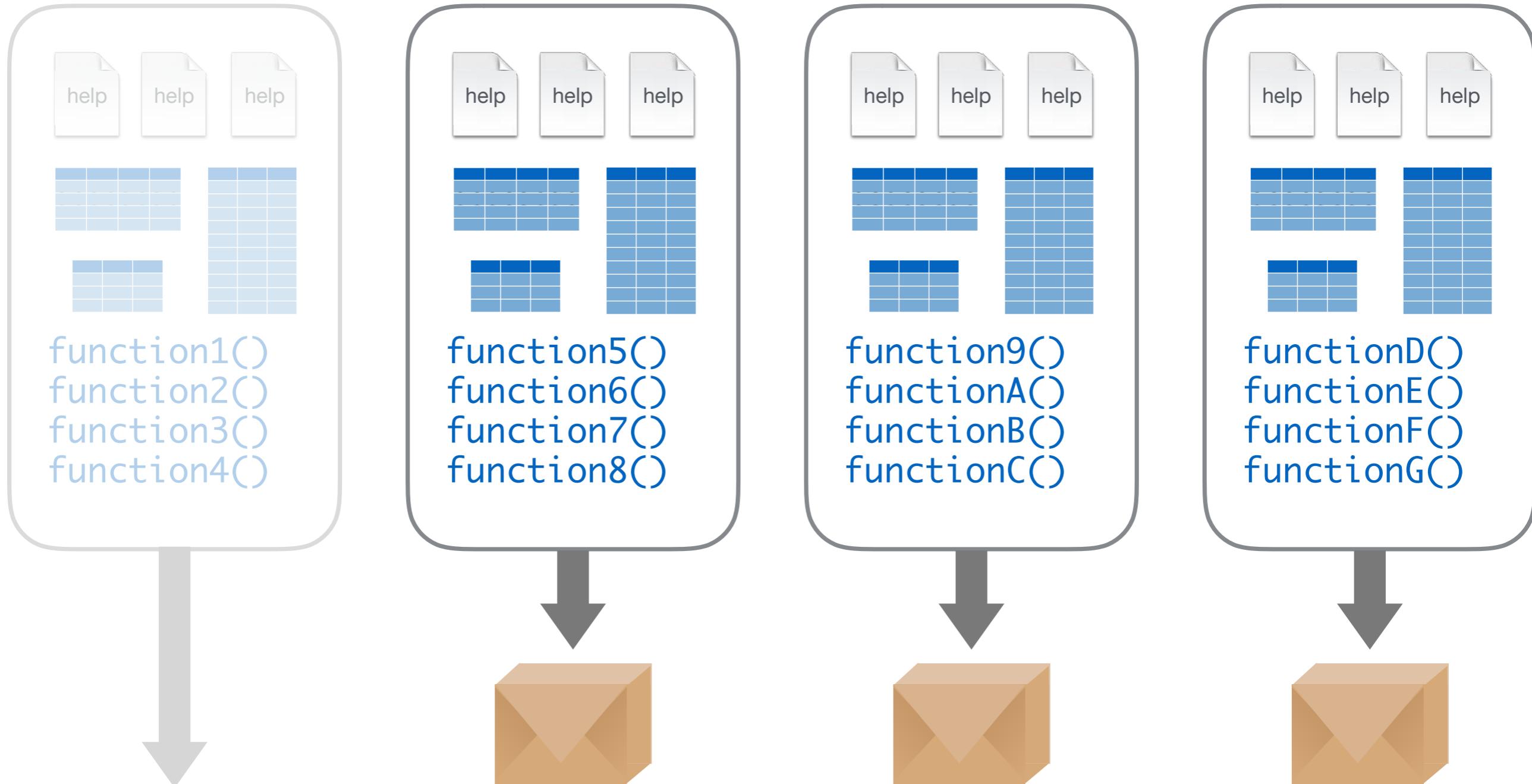
- Comments (lines that R does not try to evaluate) start with **#**
 - Good code includes lots of comments!
- Computers are stupid: R can't guess what you want to do. Attention to detail is important!

R PACKAGES

R PACKAGES



R PACKAGES



Base R

R Packages

R PACKAGES

The screenshot shows the homepage of The Comprehensive R Archive Network (CRAN). At the top, there's a navigation bar with tabs for "The Comprehensive R Archive" and a search bar with the URL "https://cran.r-project.org". Below the header is the CRAN logo and a sidebar with links to "CRAN Mirrors", "What's new?", "Task Views", and "Search". The main content area features a large blue banner with white text that reads: "There are now > 10,000 R packages on CRAN. You will never use >99% of them." To the right of the banner, the text "Available CRAN Packages By Name" is displayed above a grid of package names and their descriptions. The packages are listed in alphabetical order from A to Z.

Available CRAN Packages By Name	
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z	
A3	Accurate, Adaptable, and Accessible Error Metrics for Predictive Models
abyyR	Access to Abbyy Optical Character Recognition (OCR) API
abc	Tools for Approximate Bayesian Computation (ABC)
abc.data	Data Only: Tools for Approximate Bayesian Computation (ABC)
ABC.RAP	Array-Based C-Gaussian Analysis Pipeline
abcdeFBA	ABCDE_FBA: A-Biologist-Can-Do-Everything of Flux Balance Analysis with this package
ABCoptim	Implementation of Artificial Bee Colony Optimizer for ABC
ABCp2	Approximate Bayesian Computational Model for Estimating P2
abctf	Approximate Bayesian Computation via Random Forests
abctools	Tools for ABC Analyses
abd	The Analysis of Biological Data
abe	Augmented Backward Elimination
abf2	Load Gap-Free Axon ABF2 Files
ABHgenotypeR	Easy Visualization of ABH Genotypes
abind	Combine Multidimensional Arrays
abjutils	Useful Tools for Jurimetical Analysis Used by the Brazilian Jurimetrics Association
abn	Modelling Multivariate Data with Additive Bayesian Networks
abnormality	Measure a Subject's Abnormality with Respect to a Reference Population
abodOutlier	Angle-Based Outlier Detection
ABPS	The Abnormal Blood Profile Score to Detect Blood Doping
AbsFilterGSEA	Improved False Positive Control of Gene-Permuting GSEA with Absolute Filtering
AbSim	Time Resolved Simulations of Antibody Repertoires
abstractr	An R-Shiny Application for Creating Visual Abstracts
abundant	High-Dimensional Principal Fitted Components and Abundant Regression
Ac3net	Inferring Directional Conservative Causal Core Gene Networks
ACA	Abrupt Change-Point or Aberration Detection in Point Series
acc	Exploring Accelerometer Data

USING PACKAGES

1

```
install.packages("foo")
```

Downloads files to computer

1 x per computer

2

```
library("foo")
```

Loads package

1 x per R Session

THE TIDYVERSE



THE TIDYVERSE

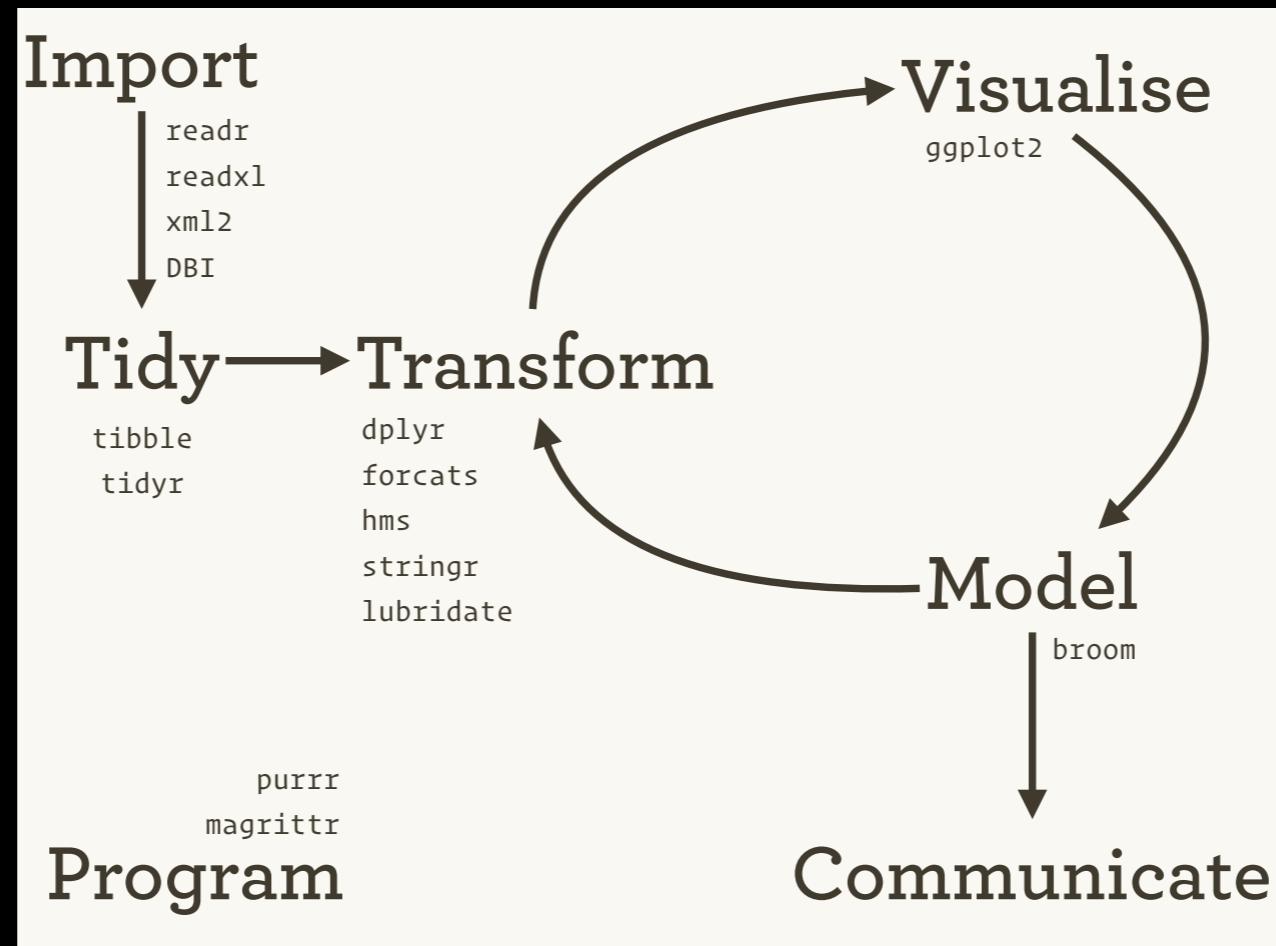


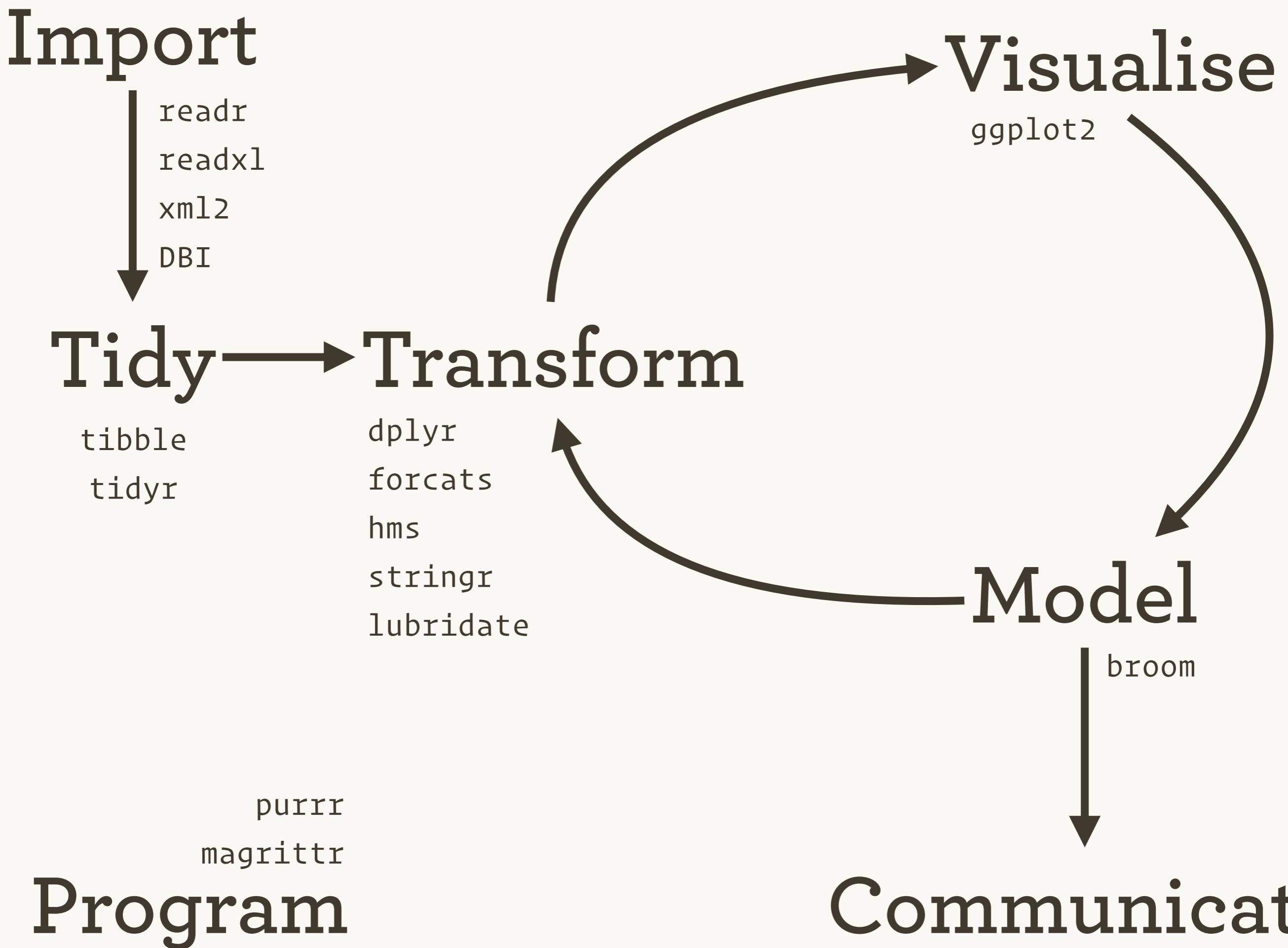
- A collection of R packages for working with and visualizing data that share an underlying philosophy and are designed to work together.

THE TIDYVERSE



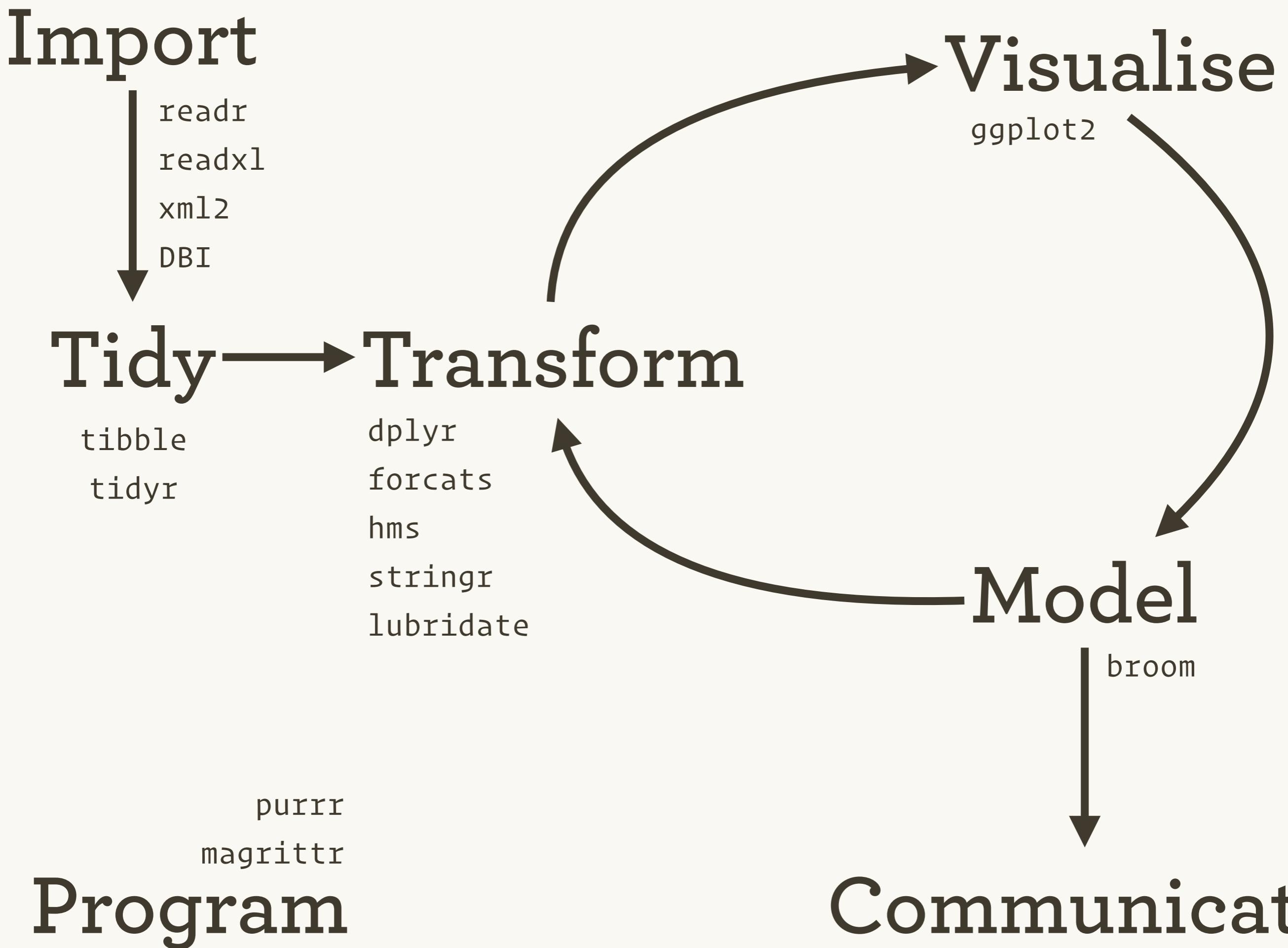
- Goal: solve complex problems by combining simple, uniform pieces.





No matter how complex and polished the individual operations are, it is often the quality of the glue that most directly determines the power of the system.

— *Hal Abelson*



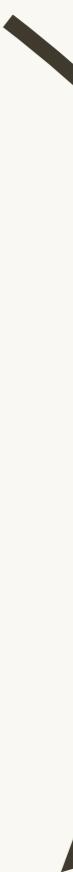
Import



Tidy → Transform



Visualise



Model



Communicate

THE TIDYVERSE



- Advantages:
 - Consistency: it's a *dialect* of R that is more readable, with pieces designed to work together
 - Coverage: end-to-end support for most data analysis goals and workflows
 - Critical Mass: huge and rapidly growing community of users—it's quickly becoming a standard
- It's a path of least resistance for inexperienced users to become proficient in data visualization and exploration

THE TIDYVERSE



Tidyverse

Packages Blog Learn Help Contribute

R packages for data science

The tidyverse is an opinionated **collection of R packages** designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

Install the complete tidyverse with:

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

THE TIDYVERSE



- For simplicity, there's an R package called "tidyverse" that's just a **short cut** for installing and loading this integrated system of packages that work together.
- Let's install the complete tidyverse by typing:

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

THE TIDYVERSE



This:

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

Does the equivalent of:

```
install.packages("ggplot2")
install.packages("dplyr")
install.packages("tidyr")
install.packages("readr")
install.packages("purrr")
install.packages("tibble")
install.packages("stringr")
install.packages("forcats")
install.packages("hms")
install.packages("lubridate")
install.packages("feather")
install.packages("haven")
install.packages("httr")
install.packages("jsonlite")
install.packages("readxl")
install.packages("rvest")
install.packages("xml2")
install.packages("DBI")
install.packages("modelr")
install.packages("broom")
install.packages("cli")
install.packages("crayon")
install.packages("dbplyr")
install.packages("magrittr")
install.packages("pillar")
```

THE TIDYVERSE



But this:

```
library("tidyverse")
```

Only loads the core:

```
library("ggplot2")
library("dplyr")
library("tidyr")
library("readr")
library("purrr")
library("tibble")
library("stringr")
library("forcats")
```

- These are the most general (and useful) parts.
- We will use all except for the “purrr” package.

BASIC R: OPERATORS

ASSIGNMENT OPERATORS

OPERATOR	DESCRIPTION
←	ASSIGN A VALUE TO A NAME
=	SAME BUT MORE LIMITED (DON'T USE FOR NOW)

```
my_name ← "Fernando Campos"  
my_height ← 182
```

Tip: use RStudio shortcut (**⌘ + -** in macOS, **Alt + -** in Windows)

BASIC R: OPERATORS

ARITHMETIC OPERATORS

OPERATOR	DESCRIPTION
+	ADDITION
-	SUBTRACTION
*	MULTIPLICATION
/	DIVISION
^	EXPONENTIATION

```
2 + 4  
2 - 4  
2 * 4  
2 / 4  
2 ^ 4
```

BASIC R: OPERATORS

COMPARISON OPERATORS

OPERATOR	DESCRIPTION
<	LESS THAN
\leq	LESS THAN OR EQUAL TO
>	GREATER THAN
\geq	GREATER THAN OR EQUAL TO
=	EXACTLY EQUAL TO
\neq	NOT EQUAL TO

```
# What kind of values will be returned?
```

```
2 < 4
```

```
2  $\geq$  4
```

```
2 = 4
```

```
2  $\neq$  4
```

BASIC R: OPERATORS

LOGICAL OPERATORS

OPERATOR	DESCRIPTION
!	NOT
	OR
&	AND

```
!TRUE  
TRUE | FALSE  
TRUE & FALSE  
(1 < 5) & (7 > 1)
```

```
my_name ← "Fernando Campos"  
my_height ← 182  
(my_height > 190) | (my_name == "Fernando Campos")
```

PIPES

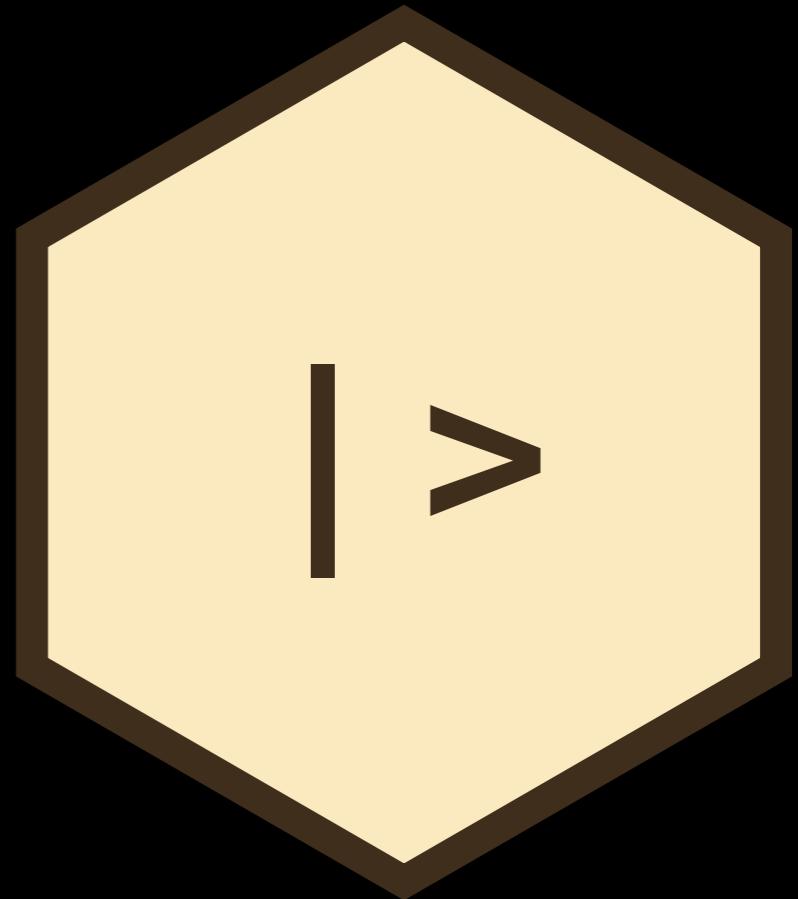


PIPES

First popular R pipe



New native pipe



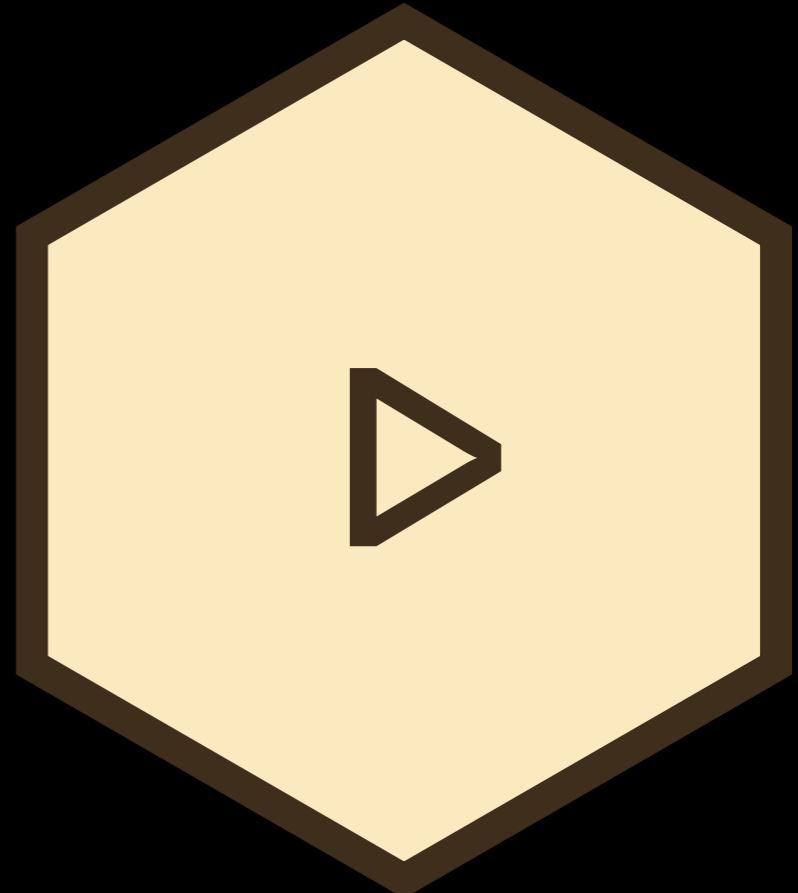
PIPES

First popular R pipe

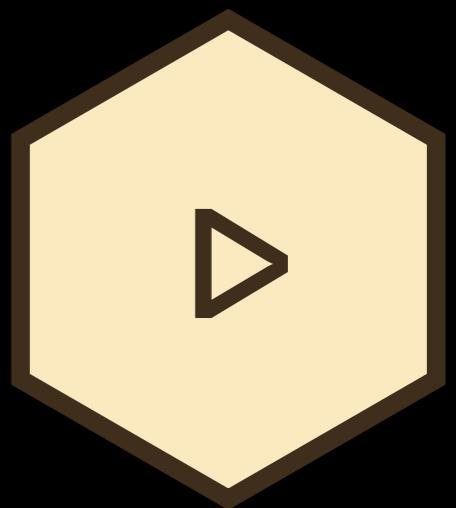


New native pipe

Or using font with ligatures:
(Jetbrains mono)



PIPES

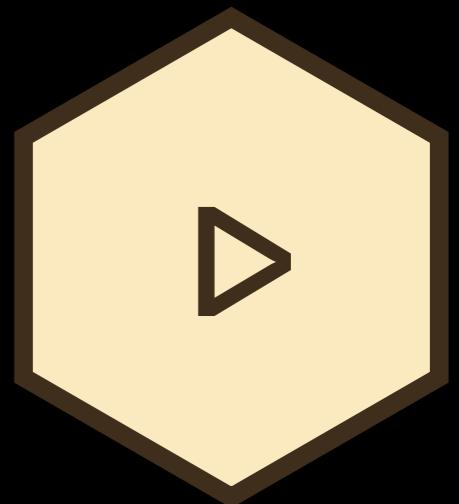


- Recall: in R, you do most things using functions

```
my_numbers <- -1:5
```

```
mean(my_numbers)
```

PIPES

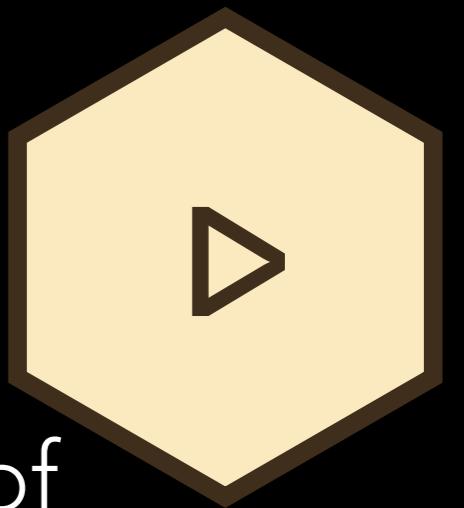


- Pass thing on left into function on the right (as the first argument)

`x > f()` becomes `f(x)`

`x > f(y)` becomes `f(x, y)`

PIPES

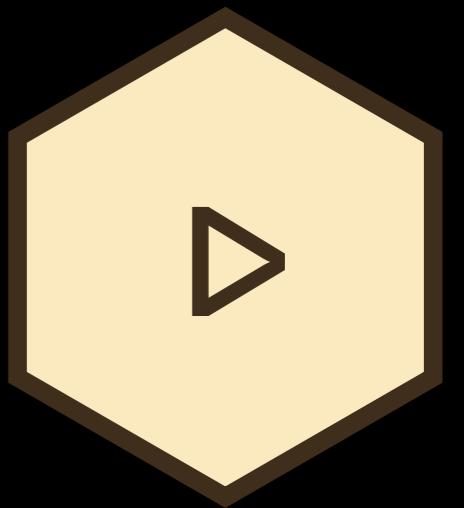


- Why use this? Suppose you have a *sequence* of operations to perform on your data.

```
my_numbers ← -1:5
```

- "Take the data, first take the square root of each number, then exponentiate each number, then take the mean."

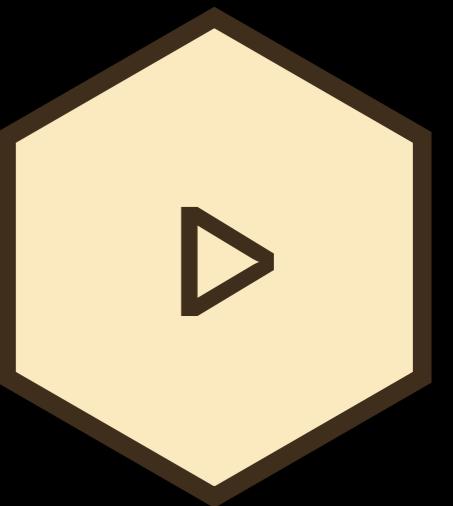
PIPES



- Option 1: execute each function one by one, using the previous result in each subsequent function

```
result_1 ← sqrt(my_numbers)
result_2 ← exp(result_1)
result_final ← mean(result_2)
```

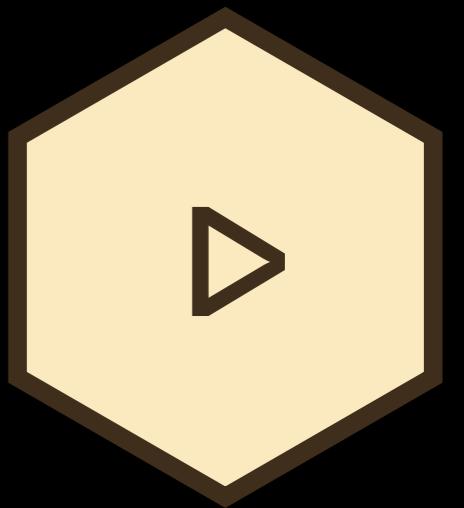
PIPES



- Option 2: nest the functions; functions will be evaluated from inner-most to outer-most

```
result_final ← mean(exp(sqrt(my_numbers)))
```

PIPES



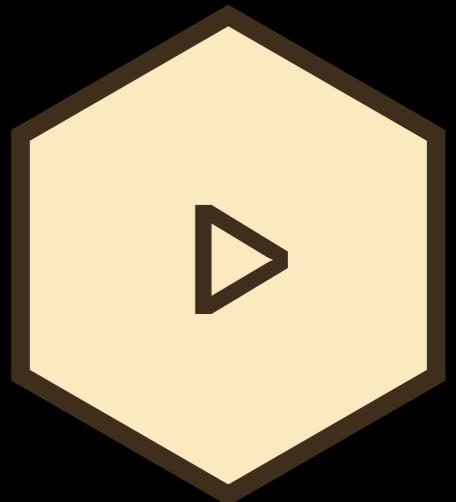
- Option 3: chain the functions together in their natural sequence using pipes

```
result_final ← my_numbers %>  
  sqrt() %>  
  exp() %>  
  mean()
```

- More human-readable:

"Take the data, then take the square root of each number, then exponentiate each number, then take the mean."

A SHORTCUT TO THE PIPE



- Instead of typing out "▷" each time...

Cmd

+

Shift

+

M

(Mac)

Ctrl

+

Shift

+

M

(Windows)

A SHORTCUT TO THE PIPE



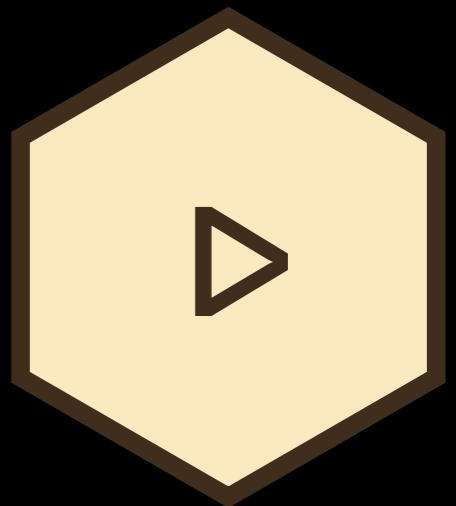
- Why M? The pipe comes from the **m**agrittr package.



"The Treachery of Images"

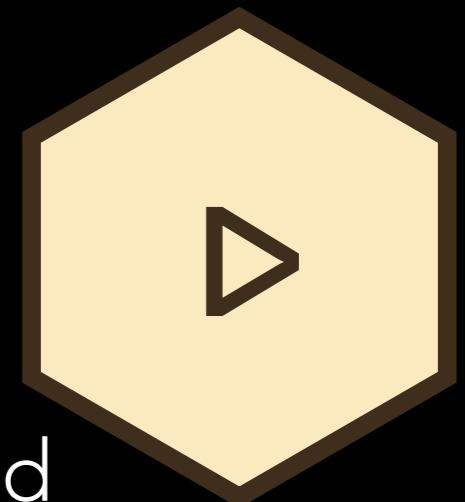
Famous 1929 painting by surrealist painter René **M**agritte

WHEN TO USE PIPES



- Pipes are most useful for rewriting fairly short linear sequences of operations.
- If you have lots of steps (~10 or more), it's still a good idea to make an intermediate object to make code more readable.

MORE USEFUL EXAMPLE



- gapminder package provides a data set on GDP and life expectancy over time, by country and continent

```
install.packages("gapminder")
library("gapminder")
View(gapminder)
```

country	continent	year	lifeExp	pop	gdpPercap
Afghanistan	Asia	1952	28.801	8425333	779.4453
Afghanistan	Asia	1957	30.332	9240934	820.8530
Afghanistan	Asia	1962	31.997	10267083	853.1007
Afghanistan	Asia	1967	34.020	11537966	836.1971
Afghanistan	Asia	1972	36.088	13079460	739.9811
Afghanistan	Asia	1977	38.438	14880372	786.1134
Afghanistan	Asia	1982	39.854	12881816	978.0114
Afghanistan	Asia	1987	40.822	13867957	852.3959
Afghanistan	Asia	1992	41.674	16317921	649.3414
Afghanistan	Asia	1997	41.763	22227415	635.3414

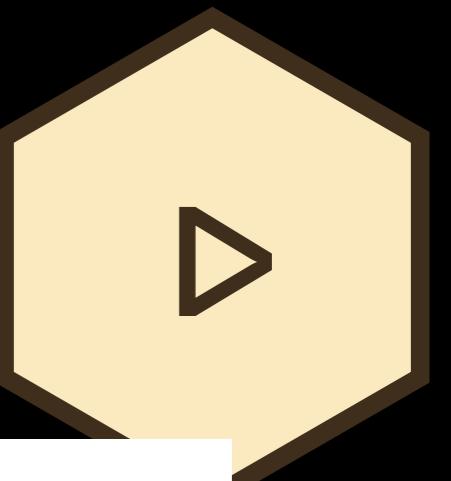
PIPES



```
count(filter(gapminder, year = 2007), continent)
```

continent	n
Africa	52
Americas	25
Asia	33
Europe	30
Oceania	2

PIPES



```
count(filter(gapminder, year = 2007), continent)
```

↑
Data

continent	n
Africa	52
Americas	25
Asia	33
Europe	30
Oceania	2

PIPES

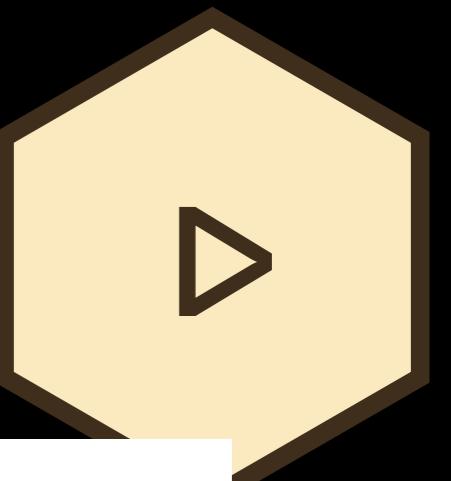


```
count(filter(gapminder, year = 2007), continent)
```

Step 1

continent	n
Africa	52
Americas	25
Asia	33
Europe	30
Oceania	2

PIPES

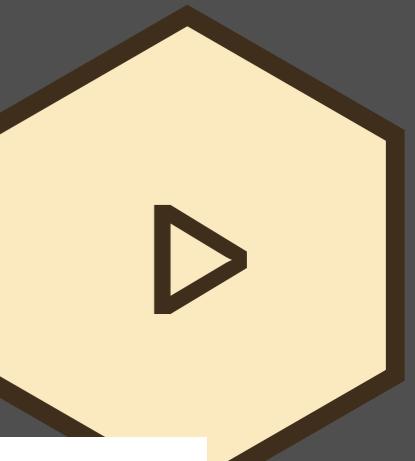


```
count(filter(gapminder, year = 2007), continent)
```

← Step 2 →

continent	n
Africa	52
Americas	25
Asia	33
Europe	30
Oceania	2

REWRITE THIS USING PIPES



```
count(filter(gapminder, year = 2007), continent)
```

continent**n**

Africa

52

Americas

25

Asia

33

Europe

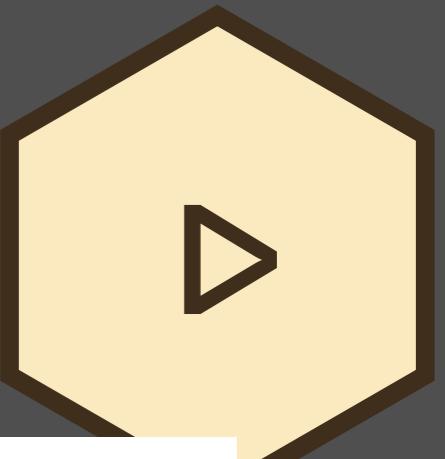
30

Oceania

2

3:00

REWRITE THIS USING PIPES



```
gapminder >  
filter(year == 2007) >  
count(continent)
```

← Data
← Step 1
← Step 2

continent	n
Africa	52
Americas	25
Asia	33
Europe	30
Oceania	2

3:00

R Markdown & Quarto



Quarto



- Basic idea behind it
- How to write in markdown
- Putting together a Quarto file

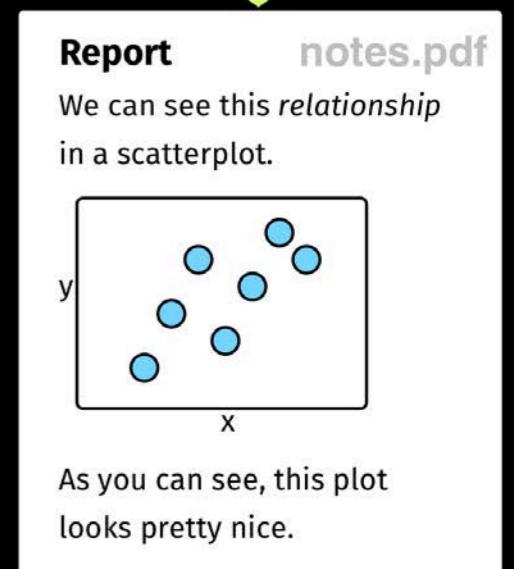
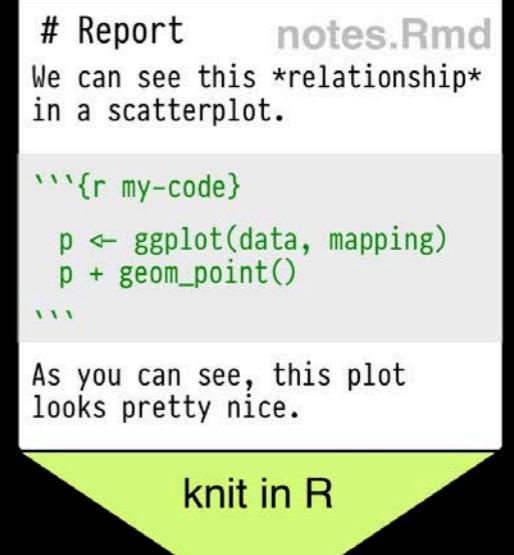


- Sometimes, you want narrative/explanation and outputs (e.g. visualizations) in the same document.
- You might even want to share or show the analysis steps and R code that you used.
- What are some situations where this would be useful?

Quarto



- R Markdown and Quarto are systems for “knitting together” these different things into a single high-quality document, report, or presentation.



EXAMPLES



- But **why?** Why not do it all in Word?
 - Copying tables and code is annoying and error-prone.
 - Word is bad at keeping a consistent layout when there are figures.
 - If data or analysis changes, just re-knit and entire report is updated.

lourdes @gossipgrill

using microsoft word

moves an image 1 mm to the left

all text and images shift. 4 new pages appear.
in the distance, sirens.

11:02 AM - 25 Mar 2016

65,445 Retweets 100,307 Likes

449 65K 100K

MARKDOWN



- The foundation is a method of formatting plain text called **Markdown**
- Philosophy: very easy to read and easy to write
- Widely used and can be converted to many output formats
- Simple formatting only
 - Paragraphs, headings, emphasis (bold/italic), numbered or bulleted lists, hyperlinks, images

The screenshot shows the RStudio interface with two panes. The left pane is the R Markdown editor for '01_markdown-basics.Rmd'. It contains the following code:

```
1 # Top-level heading
2
3 ## Second-level heading
4
5 ### Third-level heading
6
7 This is an R Markdown document that contains no code. The purpose is
8 to show you how to write in markdown.
9
10 Use asterisks to show emphasis. For example, you can make *italicized*
11 or **bold text** by enclosing in one or two asterisks.
12
13 Force a line break
14 by adding two spaces
15 at the end.
16
17 #### Lists
```

The right pane shows the rendered HTML output:

Top-level heading

Second-level heading

Third-level heading

This is an R Markdown document that contains no code. The purpose is to show you how to write in markdown.

Use asterisks to show emphasis. For example, you can make *italicized* or **bold text** by enclosing in one or two asterisks.

Force a line break
by adding two spaces
at the end.

Lists

If you forget, you can find a handy reference sheet in RStudio with
Help > Markdown Quick Reference

```
22 - A nested item is indented 4 spaces
23
24 Create an ordered list by using numbers:
25
26 1. Item 1
27 2. Item 2
28 3. Item 3
29
30 Plain http addresses are automatically converted to links:
31 https://www.campos-lab.net/courses/ant6973-s2019/
32 Or you can use a linked phrase like this: [Course
Website](https://www.campos-lab.net/courses/ant6973-s2019/)
```

1. Item 1
2. Item 2
3. Item 3

Plain http addresses are automatically converted to links: <https://www.campos-lab.net/courses/ant6973-s2019/>

Or you can use a linked phrase like this: [Course Website](#)

YOUR TURN

Create a basic CV in markdown
(you can make stuff up!)

In Rstudio:

File > New File > Markdown File

20:00

YOUR TURN

- Top-level heading is your name
- Include contact info: phone, email, and link to some website.
- Include 2nd-level headings for education, employment, publications, and teaching
- Each section should include a bulleted or numbered list of items, or a table
- Highlight years in bold and (possibly fake) journal names in italics
- Embed some kind of image at the bottom

02_short-cv.html | Open in Browser | Find

Fernando Campos

Curriculum Vitae

Phone: (210) 555-1234
Email: fernando.campos@utsa.edu
Website: campos-lab.net

Education

- 2014 Ph.D in Anthropology, University of Calgary
- 2008 M.A. in Anthropology, University of Calgary
- 2002 B.S. in Biology, Caltech

Professional Appointments and Employment

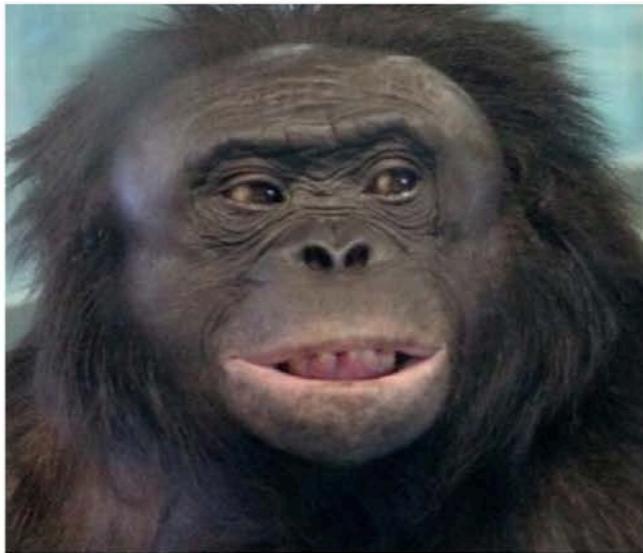
- 2018 Assistant Professor, University of Texas at San Antonio
- 2017 Postdoctoral Associate, Duke University
- 2016 Professor Practice, Tulane University
- 2015 Resereach Associate, University of Calgary

Publications

1. Campos FA, Fedigan LM. (2017). My amazing article, part 1. *Proceedings of the Royal Society* 23(11), 4907-4921.
2. Campos FA (2020). An even better follow-up article. *Nature* 12(34): 1-15.

Teaching Experience

Course Number	Course Title
ANT 2033	Introduction to Biological Anthropology
ANT 3843	Primates of the World
ANT 3333	Human Adaptability



20:00

Quarto



- What's in a Quarto file?

Quarto



- What's in a Quarto file?
 - An optional short header

| At the top

```
---  
<Header stuff goes here>  
---
```



- What's in a Quarto file?

- An optional short header

| At the top

```
---  
<Header stuff goes here>  
---
```

- Plain text with markdown formatting



- What's in a Quarto file?

- An optional short header

| At the top

```
---  
<Header stuff goes here>  
---
```

- Plain text with markdown formatting

| Interspersed

- Optional chunks of R code

```
```{r}  
<R code goes here>
```
```

Quarto



- What's in a Quarto file?

- An optional short header

```
---
```

<Header stuff goes here>

```
---
```

- Plain text with markdown formatting

- Optional chunks of R code

```
```{r}  
<R code goes here>
```
```

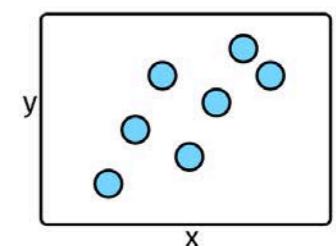
Report notes.Rmd
We can see this *relationship*
in a scatterplot.

```
```{r my-code}  
p <- ggplot(data, mapping)
p + geom_point()
```
```

As you can see, this plot
looks pretty nice.

knit in R

Report notes.pdf
We can see this *relationship*
in a scatterplot.



As you can see, this plot
looks pretty nice.

Quarto



```
```{r}
<R code goes here>
```
```



```
```{r}
#| <options for chunk behavior>
<R code goes here>
```
```



```
```{r}
#| label: fig-normal-distributions
#| fig-cap: "100 normally distributed points"
#| echo: false
plot(rnorm(100), rnorm(100))
````
```

- **label**: optional name for the chunk
- **fig-cap**: adds a caption to graphical results.
- **echo**: false prevents code, but not the results from appearing in the finished file. This is a useful way to embed figures without showing code.
- **warning**: false prevents warnings that are generated by code from appearing in the finished.
- **message**: false prevents messages that are generated by code from appearing in the finished file.



- Let's experiment. In RStudio, create a new Quarto file by going to:
 - File > New File > Quarto Document...

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

Untitled1* Render

Source Visual Outline

```
1 ---  
2 title: "Quarto Practice"  
3 format: html  
4 ---  
5  
6 ## Quarto  
7  
8 Quarto enables you to weave together content and executable code  
Quarto see <https://quarto.org>.  
9  
10 ## Running Code  
11  
12 When you click the **Render** button a document will be generated that includes both content and the output of  
embedded code. You can embed code like this:  
13  
14 ```{r}  
15 1 + 1  
16 ````  
17  
18 You can add options to executable code like this  
19  
20 ```{r}  
21 #| echo: false  
22 2 * 2  
23 ````  
24  
25 The `echo: false` option disables the printing of code (only output is displayed).  
26
```

Process the whole document

Header with information about document

Notes & discussion with formatting instructions

Chunk of R code

Run this chunk

Chunk of R code

Run all chunks up to here

2:24 # Quarto Practice Quarto

QUARTO



- That's the tip of the iceberg, but should be enough for creating a basic report.
- If you want to know how to do more, there's an entire (free!) guide on Quarto:
 - <https://quarto.org/docs/guide/>