bit.ly/2020-05-12-rice-doc

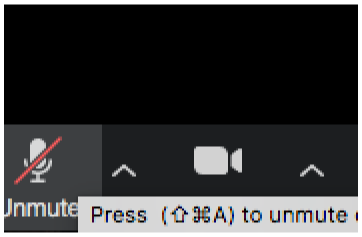
Workshop links

- Zoom: https://riceuniversity.zoom.us/j/423377104.
- CoC: http://docs.carpentries.org/topic_folders/policies/code-of-conduct.html
- Page: https://maurolepore.github.io/2020-05-12-rice/
- Curriculum: Open Science with R (parts 1-6)
- Guide: https://carpentries.org/blog/2020/04/great-online-learning-student/
- Survey (pre): https://carpentries.typeform.com/to/wi32rS?slug=2020-05-12-rice
- R & RStudio from the web browser: https://rstudio.cloud/project/1133167
- Maybe install R & RStudio: https://happygitwithr.com/install-r-rstudio.html
- Code demo: https://github.com/maurolepore/2020-05-12-open-science-with-r
- Docs line numbering for Chrome: https://bit.ly/2SSDoDJ
- Inspiring workshop: https://github.com/rstudio-conf-2020/data-science-tidy
- Book: https://r4ds.had.co.nz/
- Report: https://docs.google.com/document/d/1hToaOv-7zCGB0JnjUXC5ojyCtnSisbWgNJyDkiBD4PI/edit?usp=sharing
- Rice resources:
    - http://library.rice.edu/services/data-and-donuts-courses
    - http://library.rice.edu/data-assistance
    - http://library.rice.edu/research-data-services
-

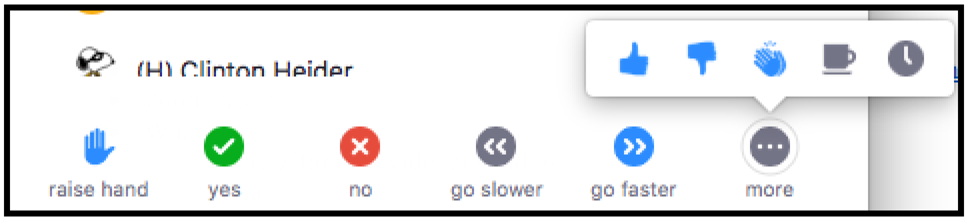
Day 1

Before we start (10’)

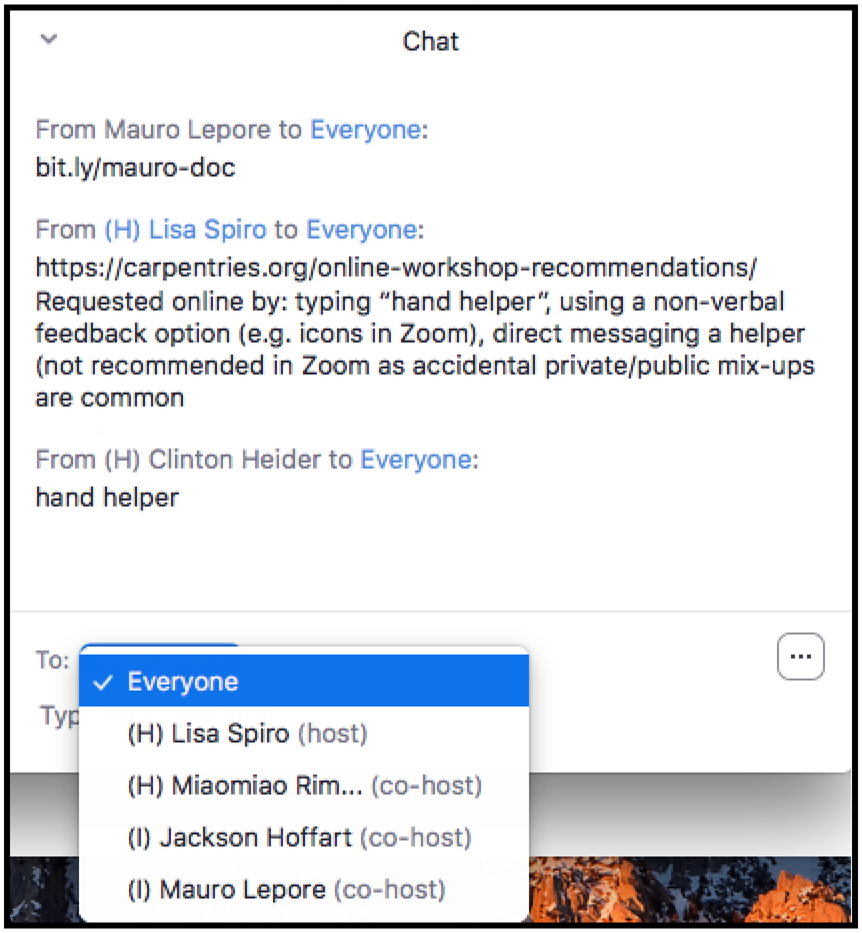
- Welcome (Lisa)
    - Research Data Services: http://library.rice.edu/research-data-services
    - Code of Conduct: https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html
- Zoom basics (John):
    - Functions
        - Mute/Unmute, video/no video



        - Chat:
            - Buttons: Raise hand, Yes, No, Slower, Faster



- Chat with helpers directly/privately



From Mauro Lepore to Everyone:
bit.ly/mauro-doc

From (H) Lisa Spiro to Everyone:
https://carpentries.org/online-workshop-recommendations/
Requested online by: typing "hand helper", using a non-verbal feedback option (e.g. icons in Zoom), direct messaging a helper (not recommended in Zoom as accidental private/public mix-ups are common

From (H) Clinton Heider to Everyone:
hand helper

To: ✓ Everyone
(H) Lisa Spiro (host)
(H) Miaomiao Rim... (co-host)
(I) Jackson Hoffart (co-host)
(I) Mauro Lepore (co-host)

  - Communication exercise:
    - Raise your hand
    - Send a message to a helper
    - Say "no"
    - Say "yes"
    - Applaud yourself

Sign in (5’)

Format:

- Name, (optional) institution, (optional) public profile

Examples:

- Mauro
- Mauro Lepore, 2 Degrees Investing
- Mauro Lepore, https://2degrees-investing.org/, twitter mauro_lepore

Instructors

Prefix your zoom-name with (I), e.g. (I) Mauro

1. Mauro Lepore, https://2degrees-investing.org/, twitter mauro_lepore
2. Jackson Hoffart, https://2degrees-investing.org/, github jdhoffa twitter jdhoffart

Helpers

Prefix your zoom-name with (H), e.g. (H) Lisa

- Lisa Spiro (H): lspiro@rice.edu, Executive Director of Digital Scholarship Services, Fondren Library
- Clinton Heider (H(: heider@rice.edu, Center for Research Computing
- Miaomiao Rimmer (H): mr66@rice.edu, Fondren Library

Learners

1. Athanasios Papastathopoulos-Katsaros, Rice university
2. Dongya Jia, Rice University
3. Lina Luo, Rice University, Dept of Civil and Environmental Engineering
4. Rosa Guerra-Resendez, Rice University
5. Felix Wu, Rice University, Department of Psychological Sciences
6. Sungmin Lee, Rice University
7. Ted Loch, Rice
8. Linnea Ng
9. Kathleen Wu, Rice University
10. Ilinca Stanciulescu, Rice University
11. Romanos Fasoulis, Rice University
12. Dibya Mishra, Rice
13. Dhiraj Jain, Rice University Department of Systems, Synthetic and Physical Biology
14. Isabel Bilotta, Rice University, Department of Psychological Sciences
15. Swetha Sridhar
16. Eric Antley, Rice

17. Paul Treacy, Rice
18. Prabhas Hundi, Rice University, Department of Civil Engineering
19. Jacopo de Rossi
20. Evan Mulfinger, Rice University, Department of Psychological Sciences
21. Shannon Cheng, Rice University, Dept of Psychological Sciences
22. Qinyou Hu
23. Michelle Kim

What to expect and do (5’)

- Why this lesson?
  - Based on Software Carpentries
  - Reflects the current state of R; relevant to what most want to achieve in workshop
- How closely/loosely we’ll follow the lesson?
  - Loosely, since Carpentries’ lessons are for 2 full-day, in-person workshops.
- How to manage your attention?
  - Difficult to focus on the speaker and activities given limited screen real estate.
  - To cope with this challenge, the lessons are structured so that you first focus on the instructor, then do the activity.
- How to manage your screen real estate?
- How to ask for help?
  - Use chat.
- What to do if you couldn’t finish a task on time?
  - That’s fine. The workshop materials are at https://carpentries-incubator.github.io/open-science-with-r/
- What is the general structure of each day?

  DAY 1

  Block 1

  - Before we start
  - Setup
  - 1. Introduction to the data science workflow

  Block 2

  - 2. R, RStudio, .RMarkdown

  Block 3

  - 3 Visualize data

  DAY 2

  Block 1

  - 3. Visualize data
  - 4. Transform data

  Block 2

  - 4. Transform data
  - 5. Tidy data

Block 3

- 5. Tidy data
- Buffer. Maybe 6. Programming.
- Wrap up

Setup (15’)

Ignore the Setup section of the lesson. We’ll use a simpler, more robust setup.

Objective

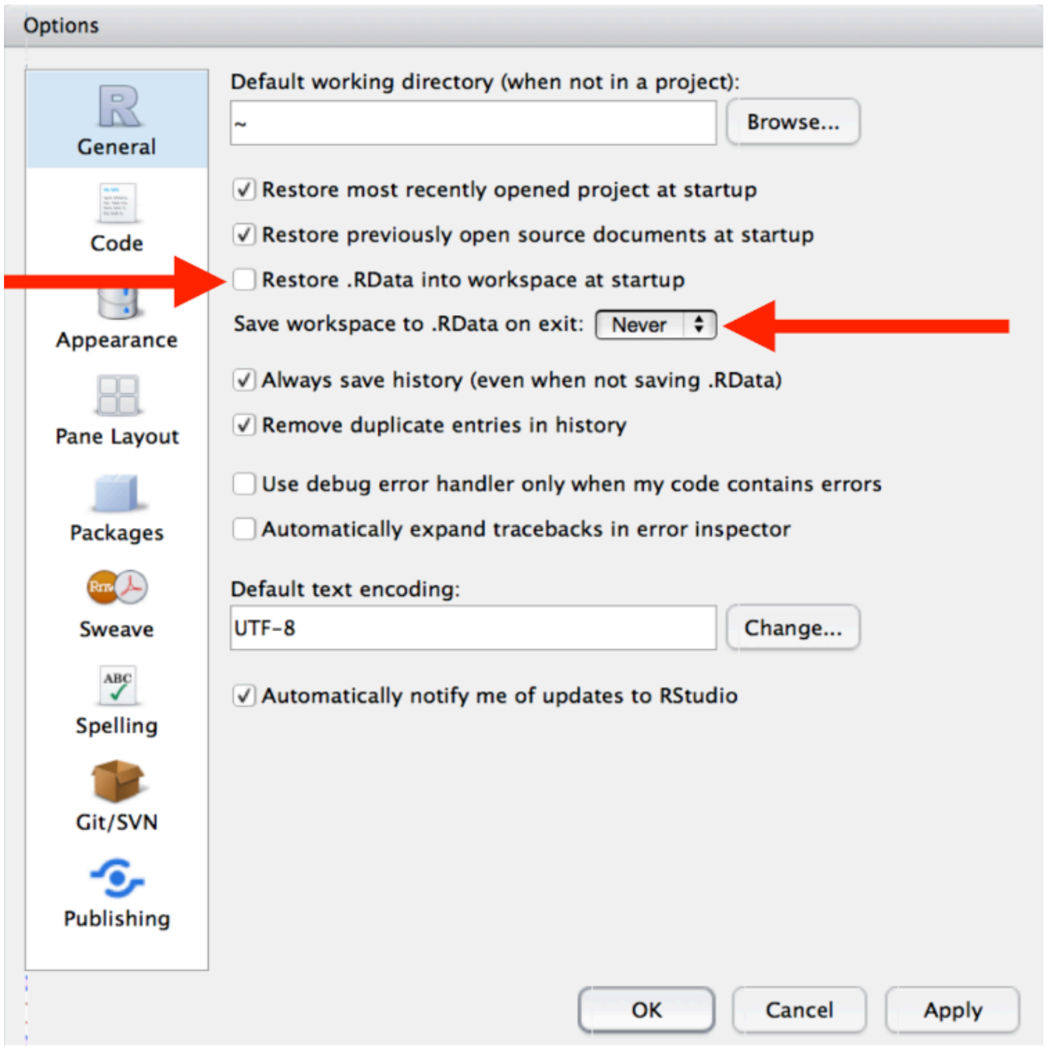
- Get an R environment identical to everyone else’s.

Your turn

Create a permanent copy of the workshop’s rstudio.cloud project: https://rstudio.cloud/project/1133167

I highly recommend that you instruct RStudio not to preserve your workspace between sessions -- https://r4ds.had.co.nz/workflow-projects.html

Tools > Global Options … > General



Default working directory (when not in a project): ~

✓ Restore most recently opened project at startup

✓ Restore previously open source documents at startup

☐ Restore .RData into workspace at startup

Save workspace to .RData on exit: Never

✓ Always save history (even when not saving .RData)

✓ Remove duplicate entries in history

☐ Use debug error handler only when my code contains errors

☐ Automatically expand tracebacks in error inspector

Default text encoding: UTF-8

✓ Automatically notify me of updates to RStudio

Questions? / Quick feedback

- It will help if you mention when there are differences from what you describe and what you would recommend when using a local installation
    - *All recommendations will be identical between the cloud project and a local installation (not restoring .Rdata is recommended for both cloud projects and local projects)*

Question: the RStudio cloud page looked different than yours. That's not some shared page right? So all of our should look different? I didn't have the ReadMe tab/script.

It should look similar, it is a shared project. You should see files such as 02-r_rstudio_rmarkdown.Rmd in the files pane in the bottom right.

Yeah, I've got all those. I was only worried because Mauro had a ReadMe script that i didn't have. My Console just has a bunch of fine print and copywrite info. Cool?

Cool!

My RStudio Cloud tab is stuck on the "opening project" step when I use your link.

Message me if this is still happening. It can take a while to deploy the project sometimes.

Question: Is it better to work on the R.Studio cloud than in the desktop based software?

Not necessarily. The power of the cloud is that we can ensure that every learner today has access to the same files, functions, packages, version of R, version of RStudio etc. This removes the installation headaches.

Question: What does restoring the R.data at startup do?

R objects are stored from each working session in the global environment. Saving and restoring .Rdata can be "easy" as you won't "clear your history" on every RStudio session, however it also makes it more difficult to demonstrate reproducibility in your code (you could be relying on relics from a previous session). Truly reproducible code/ research should be able to be recreated from any environment.

(This is a bit of an advanced concept, but is important to build these good practices from the start)

- All clear

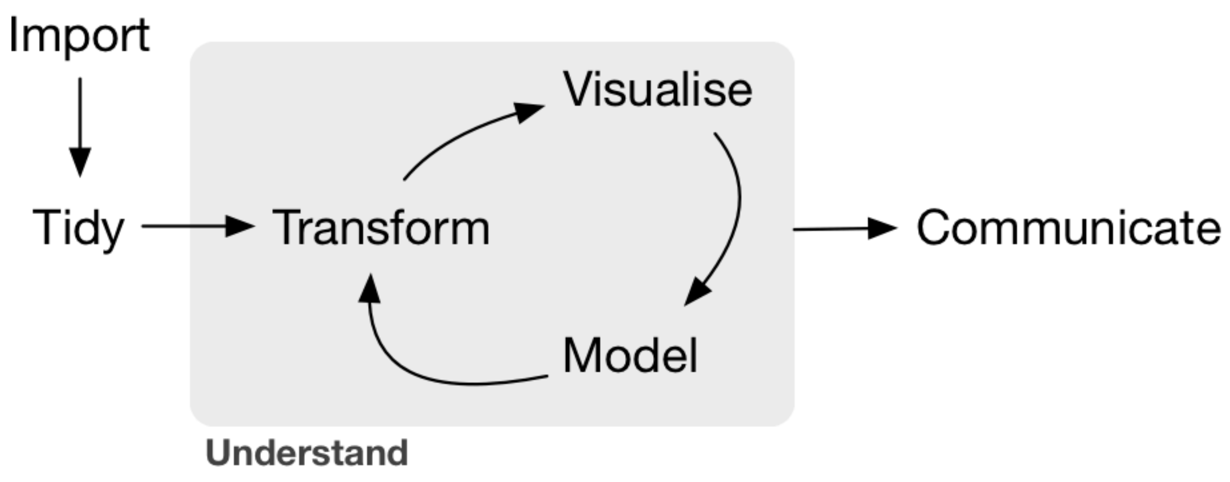
1 Introduction (15’)

Objective

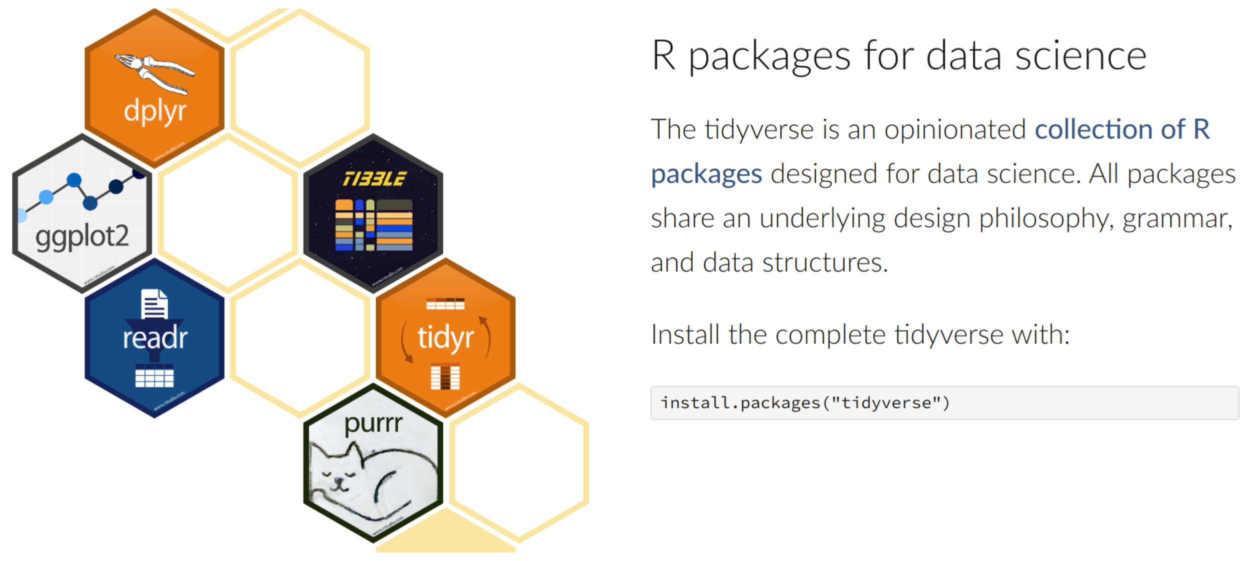
- Understand the tools you need in a typical data science project.
- Understand the structure of “tidy” data.

The tools needed in a typical d20-05-12-rice - Google Docsata science project

https://r4ds.had.co.nz/introduction.html



https://www.tidyverse.org



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:

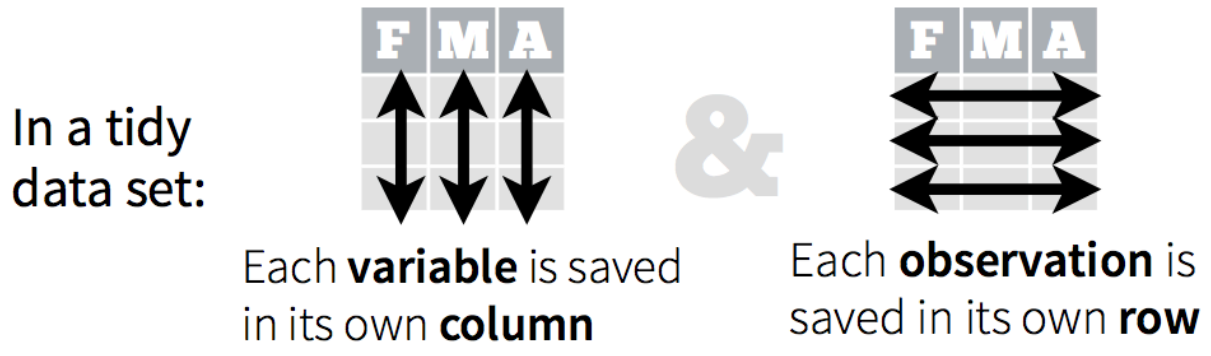
```
install.packages("tidyverse")
```

- Cheat Sheets: https://rstudio.com/resources/cheatsheets/

The structure of tidy data

https://vita.had.co.nz/papers/tidy-data.pdf

Tidy datasets are easy to manipulate, model and visualise, and have a specific structure: each variable is a column, each observation is a row


In a tidy data set:
F M A
Each variable is saved in its own column
&
F M A
Each observation is saved in its own row

For example, sample = one row/ observation; one variable (e.g. ID, year of collection) in column

Questions? / Quick feedback

Should we have a dataset ready to work on? Or are you giving us data for exercises?

Hi, yes -- Data will be provided for the exercises. (Awesome--thanks)

R & RStudio, Rmarkdown
https://carpentries-incubator.github.io/open-science-with-r/02-Rstudio/index.html

https://rstudio.cloud/project/1133167

Important parts of data science pipeline.

- Click save a permanent copy to save to your R Studio Online account.
- R= programming language; a computer language for scientists. R is closer to human syntax than to machine language; it is relatively easy to read R code.
- R is a series of functions , datasets, and help files that make up base R.
- For example, math functions like add numbers.
- R has external packages that extend functionality.
- Packages exist on cran-- cran.r-project.org, https://cran.r-project.org/
- Need to install packages and then call them each session through (library)
- Tidyverse= set of R packages for data science tasks. Embed best practices for reproducible science; designed to work well together.
    - Tidyverse.org
    - library("tidyverse") to load
    - install.packages("tidyverse")
- R studio provides common tools for writing, visualizing, and testing R code. Software program like MS Word, etc. Includes version control, debugging tools, etc.
- R studio interface:
    - Bottom left: console, engine where code is executed.
    - Bottom right: view of all files in active directory.
        - Also has help tab, where you can pull up documentation for any function installed into R.
        - Can visualize and export plots (as various file types)
        - View history of last commands
    - Adding new pane-- text editor, like Notepad. Document commands that we write down and combine into useful workflows.
- RMarkdown
    - Authoring format for data science
    - Embedded R script + text document. Can document entire data analysis workflow from start to finish, including explanations of data, discrepancies in data, other portions of experiment you want to highlight. Can embed R code directly into this doc.
- Try it out:
    - Navigate to projects
    - Click on play/ green sideways triangle to run code chunks
- Assigning variables is how we communicate with R
- Comments
    - If line is preceded by #, it won't be executed. Helpful way to document your code.
- Naming your objects

    - Can name any object in R just about whatever you want-- can't have a digit at the beginning or contain commas, spaces, etc.
    - Define a convention for how you are naming objects; easier to have one standard
    - Snakecase is common-- all_lower_case_with_underscore
    - You can start typing the variable name and R Studio will autofill; hit tab to print the rest
    - Some people use periods to separate words, others use CamelCase
- R can handle data other than numbers.
- Most common building blocks in R:
    - String,: text, e.g. sentence
    - Double: Number, e.g. 12
    - Logical: Result of true or false
    - Vectors: sequence of values (e.g. numbers, text, logicals)
    - Data frames
- Any character object has to be enclosed in double or single quotation marks, e.g. sentence
- Exercise:
    - Logical operators
- Vectors contain multiple values
- Functions-- built in to R
    - Example: sum to add the values in a vector
    - To find functions, type ? and name of function to discover what it does, usage, arguments.
- install.packages ("praise") to install praise package
- library("praise") to get package into working session
- Why work in RMarkdown-- to document your workflow
    - Knit button will build a readable document with code and output-- ensures readable workflow
- Google how to do xyz in R to find lots of documents

Comments:

- The panels may be moved around.
- You can insert a new code chunk with the "Insert" icon (green).
- Code chunks may have arbitrary labels. But the "setup" label is special: it forces the code in that chunk to run always before everything else.
- The gear icon controls where to send the output of a chunk Inline or To the console
- You may notice that the tidyverse warns you about some conflicts (e.g. between the function "filter" from the base package and the function filter from the dplyr package). If two packages have functions with the same name, the one that wins is the one you loaded last.

- Warnings, such as the conflicts that tidyverse reports, are there to call your attention but you may choose to do nothing about it. Errors are different. They stop you from moving on.
- If you are about to pick a name convention, avoid using dot.case and prefer snake_case. Learn more at https://style.tidyverse.org/
- In RStudio, the “Packages” tab helps you to find what packages you have installed, and also helps you to install new packages or update existing ones.

Questions? / Quick feedback

- It seems like the functions are case-sensitive...is that true?
    - *Yes*
- How do we save the code in RStudioCloud that we’ve been working on here for later reference and access?
    - *If there is a temporary project label, it is not saved. Otherwise, you have saved this R Cloud session and can reopen it.*
- We will probably do this later but, are we going to use an example of a data file (excel, etc) and do the tidy step/import to R?
    - *Yes*
- How do we access the webpage that you showed at the end where we can see all the codes you run? I think it was R studio cloud page.
    - *You click the knit button on the R Markdown file; you will render that file. .rmd means r markdown file, .md designates a regular markdown file.*
- Question about accessing multiple values by index.
    - *Example answer: > names[c(1,2)]*

Why do we use <- as an assignment operator and not = usually?

- *Great question! You can use both, = or “<-” but in R we prefer “<-” -- it’s more idiomatic. There are other reasons but they are too subtle to worry about.*

Visualizing data with ggplot2

https://carpentries-incubator.github.io/open-science-with-r/03-ggplot2/index.html

https://ggplot2.tidyverse.org/ will need to visit to copy code; has lots of useful info about ggplot, including a cheatsheet and reference tab that gives you a visual overview of what you can accomplish with functions and examples

Many tidyverse packages have cheatsheets, and all of them have online documentation and example exercises to help you build your knowledge. If you're ever curious about what you can do, check out tidyverse.org and look through the other functions and packages that are around.

aes is a function for aesthetics

Adding colors needs to be specified through color

Shift command or control O shows outline view of r markdown file

Use head() function to get first 6 rows by default

Think through what you expect to see before you run a plot

Control Alt C will run the code (keyboard shortcuts; see menu in R Studio)

Customizing

- Defaults are good, but you may still want to customize
- Add layers to plot-- create a custom plot
- Can use themes to avoid making manual tweaks, eg. bw to change gray background to black and white
- labs() for labels

On the topic of shortcuts, there is an RStudio cheat-sheet here (a bit overwhelming right now, but useful in the long run): https://rstudio.com/wp-content/uploads/2016/01/rstudio-IDE-cheatsheet.pdf

It is considered rude to install software in someone else's computer; comment that out.

Can begin to type in package name in R Studio to see the functions available

(syntax is `` `package::function()` ``)

Exercise: type this code from ggplot website. Insert chunk into rmarkdown.

```
library(ggplot2)
```

```
ggplot(mpg, aes(displ, hwy, colour = class)) +
  geom_point()
```

Autocomplete in RStudio helps you avoid typos:

- Begin typing whatever you want to type, and RStudio will suggest objects, packages, functions etc. that it has available.
- Use the arrows to select the appropriate entry, and hit tab to autocomplete
- Autocomplete guesses if you make typos

Note:

- `library(tidyverse)` will load `ggplot2` amongst other packages (it will load every package in the `tidyverse`)

- `library(ggplot2)` will not load `tidyverse`

Questions? / Quick feedback

What did you like and should be continued? What can be improved?

How do I solve this? Thanks ...there are specific functions for x, y, and title labels. Find them.

> Try visiting the Reference section of the ggplot2 website. See what you find. What you are looking for is likely under "scales".

Good job with the fill in the blanks exercises! They were very intuitive and helped me see the logic behind the functions.

I thought that the ggplot section needed more time, which was taken more by the earlier chapter. Overall, it was good, thanks! One thing that would be helpful is to have answers for the bored? questions.

All solutions will be available in the 03-visualize-solution.Rmd file

Great workshop! The workshop didn't go too fast, or too slow it was just right. It'll be great if we can learn more about how to graphs that we commonly use (line graphs, bar graphs) in APA style and learn how to incorporate the graphs directly in our papers.

I liked the "Bored?" sections. Would be nice to have that throughout! Overall great workshop; thanks!

What is the function of '+' before the geom_point() function while using ggplt()?

Great question! It looks the same as the `+` you type to add 1 + 1. ggplot2 contains a some "magic" for `+` to behave understand that, in the context of ggplot2, `+` should add layers.

https://carpentries-incubator.github.io/open-science-with-r/img/rstudio-cheatsheet-ggplot.png

I got confused/lost when the code chunks were out of order in the ggplot sections -that was difficult to follow.

Good point. Thanks for the feedback. I'll try to keep things as stable as possible.

Not sure how I'm saving all the code we did today. I suppose it's intuitive and easy to figure out. → hit save (the floppy disk icon) in the upper left.

Exactly. That's the right way to save your code. I forgot to mention because I do it automatically, like driving a car. Sorry. (I use a keyboard shortcut Control + S)

Yes! That's a good point and something we failed to cover. You will have to save each file in the RStudio session before closing it down.

I really appreciate all of the resources/cheat sheets provided, I think potentially we could move a little faster for the next workshop!

Thanks! We'll try to speed things up

Very well executed workshop. I learned a good deal. Thank you very much.

Thanks, nice to hear we are on the right track.

This workshop was amazing! I like that you kept checking on us so that no one was left behind.

Thanks, we'll try to keep doing that.

Could you potentially explain the grammar of ggplot2? It seems like a very modular and layered system and I think knowing what ggplot2 can do will be helpful. Thank you!

Very good question. Basically it follows the philosophy of the tidyverse team, which is that "any plot can be made using data, a geometric-representation, and a coordinate system". Check out the following diagram, which describes the basics of the grammar better than we can: https://carpentries-incubator.github.io/open-science-with-r/img/rstudio-cheatsheet-ggplot.png

Jackson and I discussed the idea of presenting the theory first. He leaned towards that. But I decided to let people play with code ASAP because the first part of today's workshop was too much "us talking". I take the blame :)

I thought the pace was good up until the last 15 minutes, went too fast. But my wifi also crashed for a few minutes so I was playing catch-up.

> *Thanks for sharing. Yeah, once you loose track it's really hard to catch up. I hope the materials we will leave you allow you to retry on your own time.*

- Question about why `ggplot(data=ca,aes(x=year,y=visitors,color=class))` throws an error
    - *Answer: use `head(ca)` to see the columns. You'll see that "class" isn't a valid header in that dataset.*

Day 2

Plan

DAY 2

Block 1

- 3. Visualize data
- 4. Transform data

Block 2

- 4. Transform data
- 5. Tidy data

Block 3

- 5. Tidy data
- Buffer / Wrap up

Sign in (whenever you can)

- Mauro Lepore

Instructors

Prefix your zoom-name with (I), e.g. (I) Mauro

3. Jackson Hoffart, https://2degrees-investing.org/ github: jdhoffa, twitter: jdhoffart
4. Mauro Lepore

Helpers

Prefix your zoom-name with (H), e.g. (H) Lisa

- (H) Lisa
- (H) John Mulligan
- (H) Miaomiao Rimmer
- (H) Clinton Heider

Learners

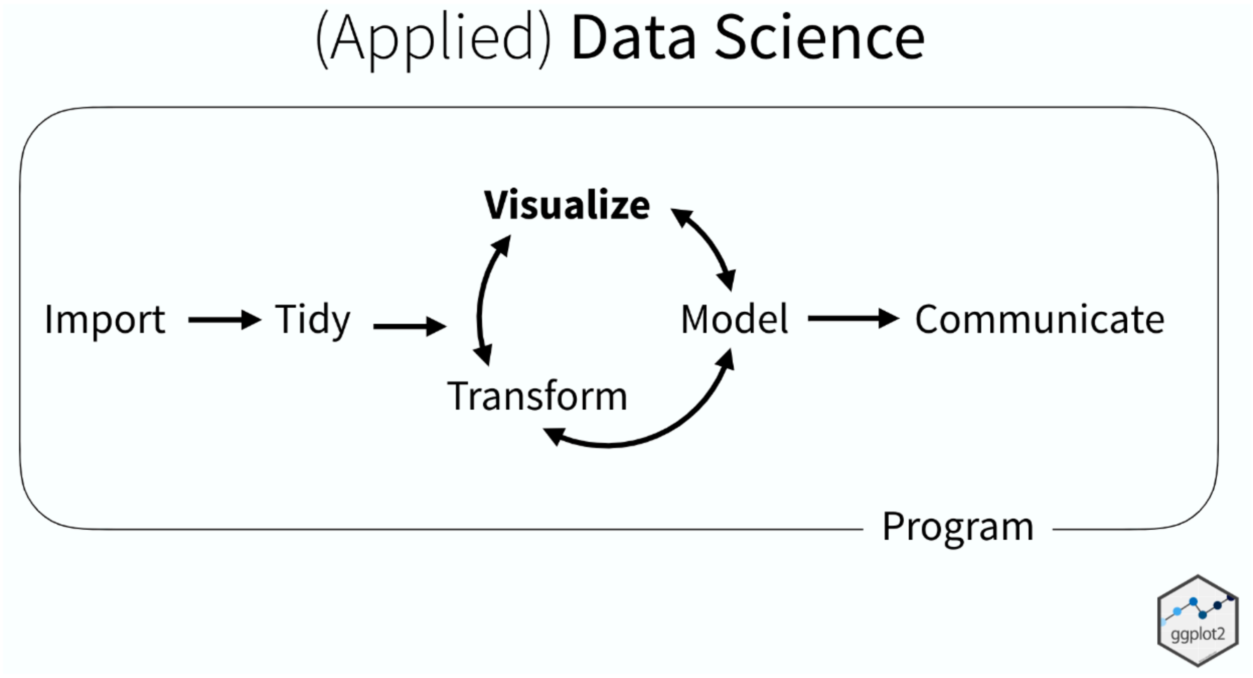
1. Dhiraj Jain, Rice
2. Paul Treacy, Rice.
3. Dongya Jia, Rice University
4. Swetha Sridhar
5. MIchelle Kim
6. Kathleen Wu, Rice University
7. Sungmin Lee, Rice University
8. Rosa Guerra-Resendez, Rice University
9. Ted Loch, Rice
10. Isabel Bilotta, Rice University, Dept. of Psychological Sciences
11. Felix Wu, Rice University, Dept. of Psychological Sciences
12. Jacopo de Rossi, Rice University
13. Shannon Cheng, Rice, Dept of Psychological Sciences
14. Eric Antley, Rice University
15. Ilinca Stanciulescu
16. Lina Luo, Rice University
17. Linnea Ng
18. Prabhas Hundi

Visualizing data with ggplot2 (continued)

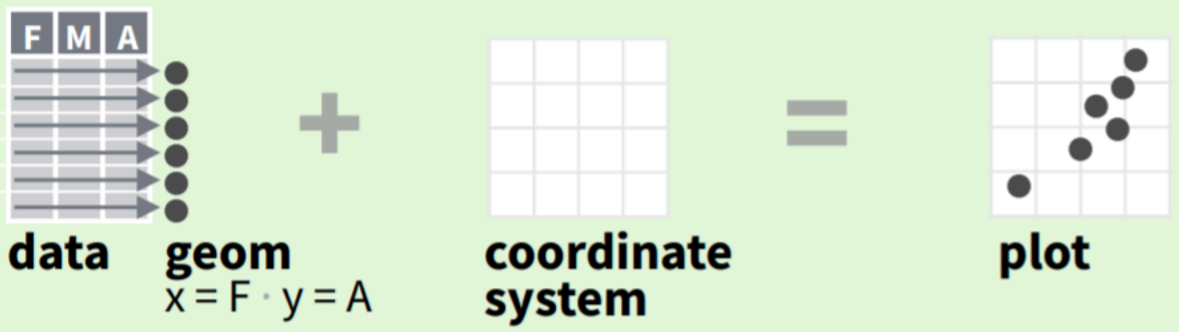
https://carpentries-incubator.github.io/open-science-with-r/03-ggplot2/index.html

https://ggplot2.tidyverse.org/

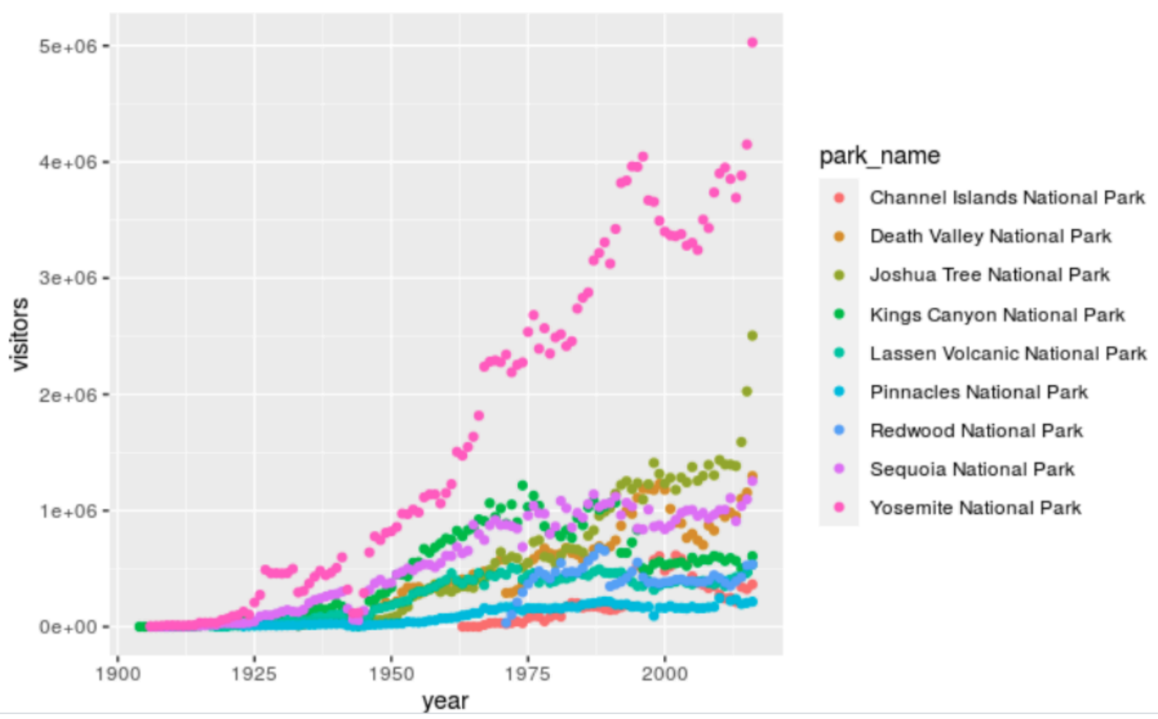
Recap (5')



ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.



~~~
```{r}
ggplot(ca, aes(year, visitors, color = park_name)) +
  geom_point()
```
~~~



Next

1. Open yesterday's rstudio.cloud project
2. Open the file 03-visualize.Rmd (or 03-visualize-solution.Rmd)
3. Scroll down to the section Facetting at line 144
4. On Zoom, click "Yes".

Faceting allows you to split big plot into smaller plots.

Facet_wrap

Geom for geometric representation

Alpha: set transparency level from 0 to 1

Jitter: randomly rearranges so that points aren't overlapping

geom_boxplot() to create boxplots

Barplot: geom_col()

Position = "dodge" to compare bars side by side

Save with ggsave

Ggplotly transform ggplot into interactive plot

Questions? / Quick feedback

- Any good (non-google) documentation/resources that have more examples, beyond the built-in 'help' function, on R?
    - *The tidyverse packages have a lot of documentation and examples directly on the various package websites (e.g. https://ggplot2.tidyverse.org/articles/extending-ggplot2.html)*
    - *Beyond this, it points to many free online "books" written about the topics. (e.g. https://r4ds.had.co.nz/data-visualisation.html)*
    - *Just about anything you find online written by Hadley Wickham and/ or Jenny Bryan will be a fantastic resource to learn R from.*
    - *See also the Learn more section at the end of this document.*
- Can we get the answer to the "Bored" question * Move the smoothed mean to become the second layer. What subtle thing happened?
    - *If you move the reorder of the layers on the code, their representation on the plot also gets reordered. For example, you may have the points under the semi-transparent uncertainty area. To move those points to the top layer, you can reorder the code so that the `geom_point()` is the last layer you call -- so it's printed at the top.*
- Why do you have an if(interactive) statement at the beginning of the plotly command?

    - This is a tricky one to answer. Normally the plots that are rendered are "static objects". To allow the plot to react to you clicking on the legend, etc. you need to specify that it is an "interactive" session

GO FASTER

- Keep doing everything faster rather than skip things
- Showing everything faster was a good idea, thank you! By doing this, we have a quick look of the possibilities with ggplot2.
- Go faster!
- I liked how you did it. Give us some time for some things then speed up. But you can do it a little faster too
- Keep the same pace
- This pace is good! Going through everything but quicker works well
- I thought it was still really helpful to go over all of this, even if it was at a faster pace

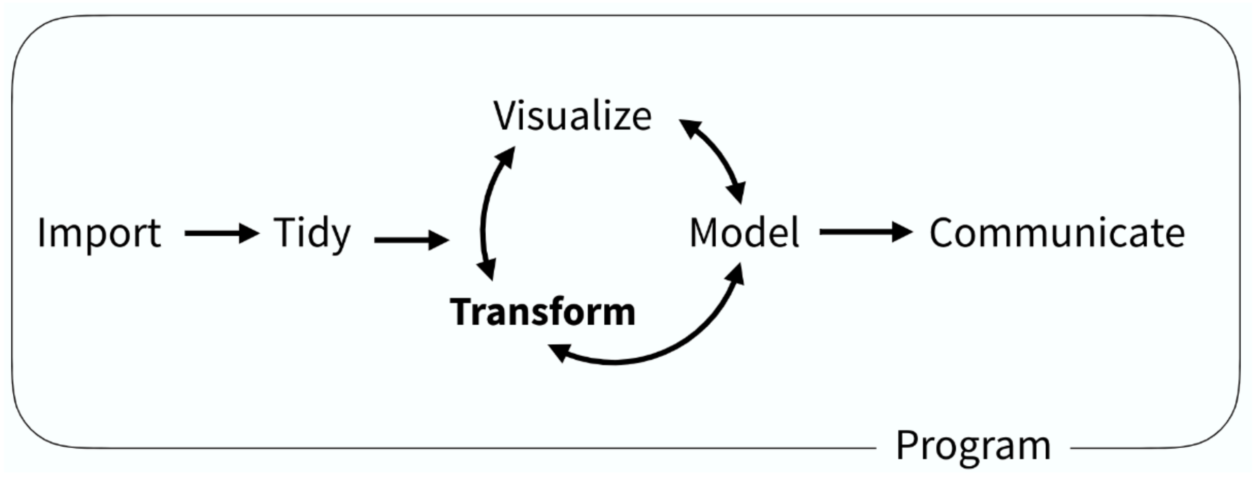
GO SLOWER

- Went a little too fast for me

Data transformation with dplyr

https://carpentries-incubator.github.io/open-science-with-r/04-dplyr/index.html

https://dplyr.tidyverse.org/



Wrangling dataset to select data, etc.

04-transform.Rmd

* Why do you think R doesn't automatically load every package you have installed on your computer every session? Would slow down R session.

## Data Transformation

Data Transformation or Data Wrangling is the process of adding, removing or doing calculations on existing columns or rows of data.

Doing this programatically in R, means that we can keep track of every bit of analysis we do between the raw data and what we actually model, visualize and/ or report.

Gapminder data: https://github.com/carpentries-incubator/open-science-with-r/blob/gh-pages/data/gapminder.csv

You can use import dataset in R Studio to import csv , Excel, etc.

str shows column name, type, first few rows

$ to select particular column/ variable in dataset

dplyr

This package has many incredibly useful functions, but we're going to focus on the five that you will probably use in every data science project that you work on:

* `filter()`: pick observations (rows) by their values

* `select()`: pick variables by their names

* `mutate()`: create new variables with functions of existing variables

* `summarise()`: collapse many values down to a single summary

* `arrange()`: reorder the observations

See https://i.imgur.com/uICSizl.png

All of these function take a dataframe as the first argument, subsequent args describe what you want to do

Result is transformed dataframe

## filter() observations

The filter function filters out rows of data, based on a "logical expression".

* The first argument is your dataset

* The second argument is a TRUE or FALSE question about one (or more) of the
  variables

## Introducing the pipe operator

The pipe operator (which was imported by `dplyr` but actually comes from the

`magrittr` package) is one of the most useful functions to exist in R, for data

pipelines.

The operator looks like this `%>%`, and you can type it easily using the RStudio

shortcut: Ctrl + Shift + M (Windows), Cmd + Shift + M (Mac)

When you see `%>%` just think "and then". It passes the output from the

left-hand side to the first argument of the right-hand side.

Take the output from left and give to first element on right. You can pipe together many functions.

## mutate() adds a new variable

So far we have just been working with, and removing, variables and observations

that already exist in the data. What if we wanted to add a new variable?

For example, gapminder has a population variable and a GDP per capita variable,

what if we wanted to recover the actual gdp?

Join

So far, we have only been working with one dataset. Before moving on, I want to

talk to you briefly about joining two different datasets by a common variable.

There are different ways to do this. Suppose we have two datasets a and b, that

we want to join together by the variable year.

We could:

* Join all matching rows from b to a

* Join all matching rows from a to b

* Join both datasets, retaining only rows that are present in both a an b

* Join both datasets, retaining all rows and values in both datasets

To get a better idea of what I mean by this, check out this diagram:

https://i.imgur.com/fV4St9d.png

Next

1. To get the latest lessons, you will need to open a NEW rstudio.cloud project: https://rstudio.cloud/project/1133167
2. And click "Save a Permanent Copy" again
3. At the end of this, you will have two different RStudio cloud projects, one with the work from Day 1, and the other from Day 2. Feel free to rename them

Questions? / Quick feedback

- Does str(csv_read(file)) auto detect data types?
    - *read_csv() itself does auto-detect column specifications from the first 1000 rows of the data, but you can also specify the column types explicitly using col_types()*
    - *If the question is, can str() detect data types prior to defining the data object, the answer is no*
    - Got it -- when it detects these, what's the order of operations? For instance, try integer, try float, try datetime, try string? Only asking because Python's csv reader is notoriously bad at unicode and one has to be rather careful with quote and separator characters, or if a column is integers but with leading zeroes -- what's a good data importation documentation page?
    - Thats a good question and im not actually sure off the top of my head. But, check out the `readr` package website and documentation for more info. Also

    if you're keen to learn more about data type handling, check out the `vctrs` project.
  - *Thanks!*

- If we have numbers in a column, can we sort the entire set to have the rows in increasing/decreasing order of a column?
  - *Yes, that's the goal of* `arrange()` *which you'll see soon.*
- Why do we need two mutates for Egypt?
  - *Not sure -- maybe a typo? Yep, a typo.*
- I didn't quite get the exact use of group()? Got the answer, It crease data summaries by grouping together variables with same value. Thank you for the for-loop analogy
  - *Thanks. I'll try to share some analogies. Let's see which analogy is best for you:*
    - *Did you understand facets in ggplot2?*
      - *Yes*
        - *Facets in ggplot2 allos you to specify subsets (or groups) of data on which to apply the same operation (i.e. the same kind of plot -- just on different panels).*
        - *Similarly, group_by() allows you to specify the groups on which to perform the the actions that follow in the transformation pipeline.*
    - *Have you ever programmed in any language?*
      - *Yes*
      - *Great. Do you know what a for loop is?*
        - *Yes*
          - *For (group in groups){do this}*
          - *Think of group_by() as a high level interface to split the data in subsets, then do the same thing for each subset. There is a for loop somewhere in the implementation of group_by(). You can surely do the same thing with a for loop, but it'll take a lot more code to accomplish (if you start from a data frame). So the tidyverse favors group_by().*

- *What is difference between using mutate and summarise to get the max of gdp?*
  - *I believethe answer here is that summarise reduces the size of the dataset to the selected variable*
  - *Yes exactly, but it also does a little more. When you summarise by group, you are collapsing all elements in the group to one row per group. To do this, you will lose all other columns than the ones that are grouped + the new column that is calculated.*

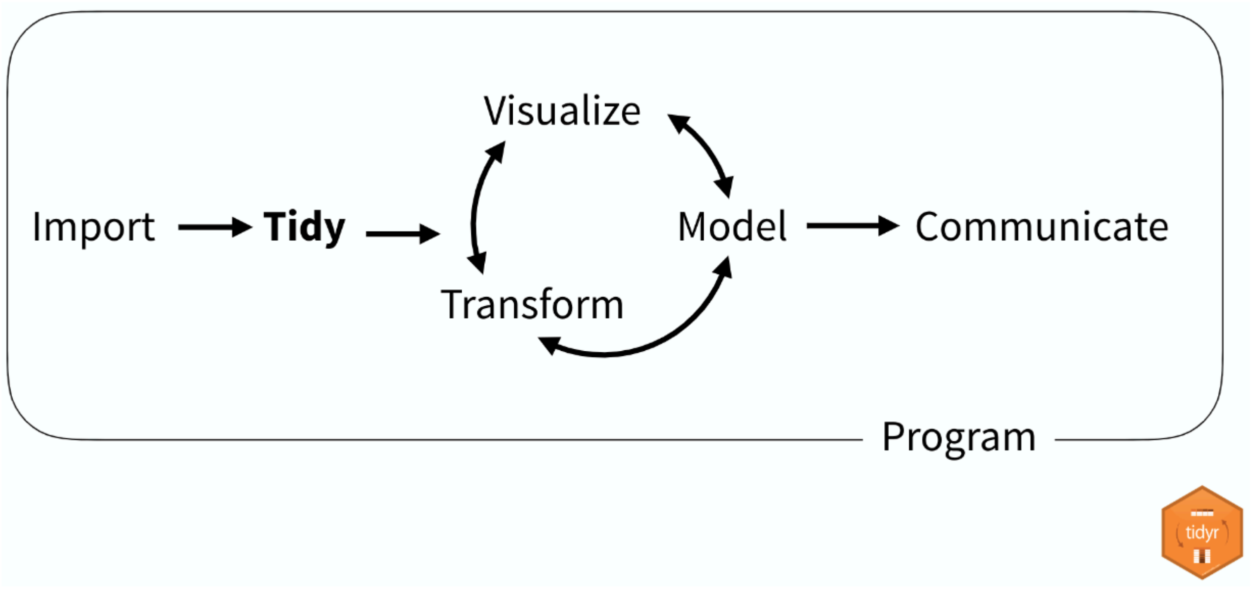
    - *So if you want to perform multiple operations, as in the example where we found the maximum life expectancy in Asia, the summarise function will remove the hooks you have into the various facets of the data, whereas mutate will retain these for your use.*
    - *Ya good example. If we group by continent, and summarise, we will lose all the more granular information at country-level. If we group by continent and mutate, we keep that granular information.*
    - *Another way to see it:*
        - *"group_by + mutate" will give you the same number of rows as you gave it*
        - *"group_by + summarise" will give you fewer rows than you gave it*
- Why do we need to use ungroup() before using arrange()?
    - *Grouped data affects the way functions behave. If you leave grouped data and forget, the output of the following functions may be unexpected.*
- Is using dplyr::select the same as select? Why would I choose one over the other syntax?
    - *Yes, good question. In general, you can always trust the syntax "package::function()", but for some obvious functions, typing out dplyr:: at the beginning every time can get pretty tedious. So often it gets dropped.*
    - *The reason you'd ever want to type "package::function()" over just "function()" is in case there are other functions in your R session that are being masked*
    - *Thanks I suspected that but wanted to make sure*
- We added gdp per capita using mutate. I was expecting the dimensions of the gapminder dataset in the upper right hand corner to change, but they stayed the same. Why didn't it change from 1704 obs of 6 variables to 7 variables? Thanks.

    - *It should definitely have changed from 6 to 7 when we mutate, as we are adding a variable. Perhaps we used select in the same pipe to drop a different variable?*

Data tidying with tidyr

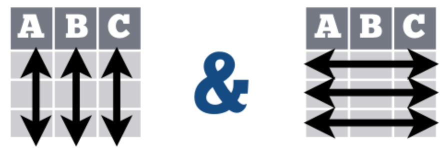
https://carpentries-incubator.github.io/open-science-with-r/05-tidyr/index.html

https://tidyr.tidyverse.org/



Tidy data is a way to organize tabular data. It provides a consistent data structure across packages.

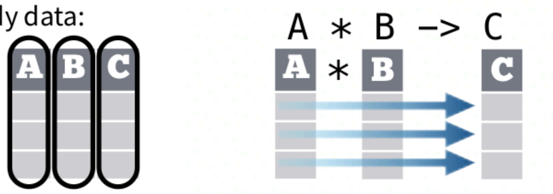
A table is tidy if:



Each **variable** is in its own **column**

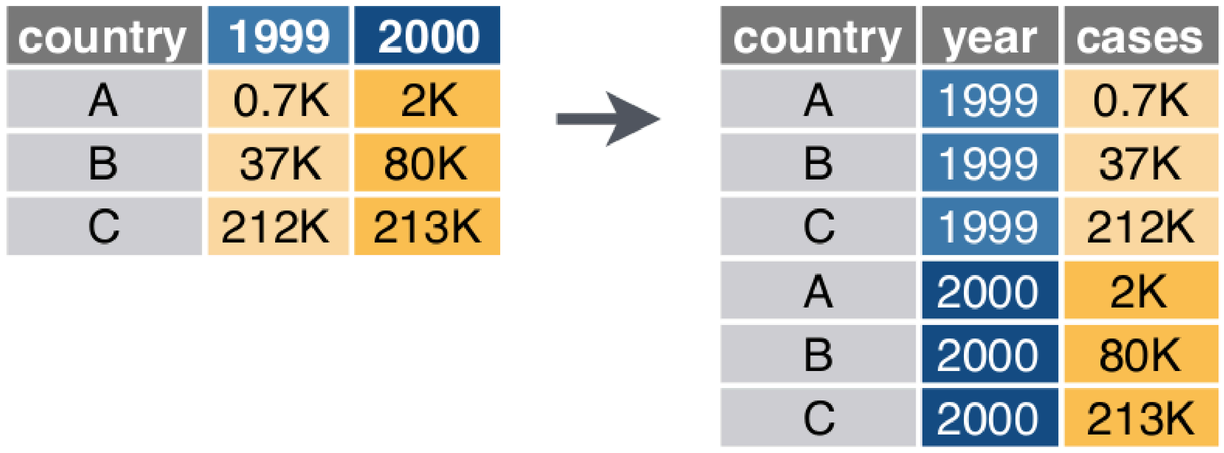
Each **observation**, or **case**, is in its own **row**

Tidy data:



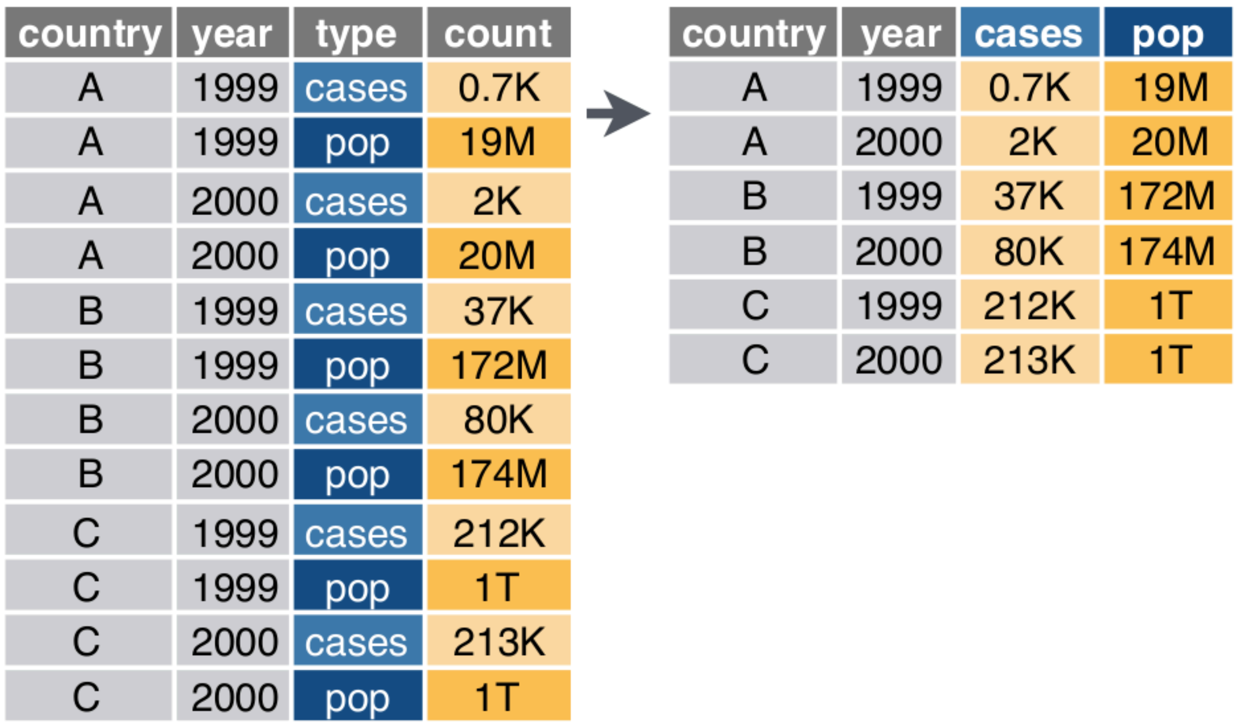
Makes variables easy to access as vectors

Preserves cases during vectorized operations



country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K



country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	213K	1T

Tidyr: clean data so that every column is a variable and every row is an observation

pivot_longer: “pivot_longer() makes datasets longer by increasing the number of rows and decreasing the number of columns” (https://tidyr.tidyverse.org/articles/pivot.html)

Questions? / Quick feedback

- Is the default separator =”_”?
  - *So check out ?separate, and in particular look at the argument “sep”*
  - *Basically, it detects if the value to separate is alphanumeric or not. If it is alphanumeric, then any non-alphanumeric is considered a separator. You can also specify the seperator explicitly by specifying the sep argument (e.g. sep = “_” would achieve the same thing)*
- How does R recognize a missing value? In stata it’s a dot. (like, if we didn’t know what the abundance value of Agarum in 2000 was) Thanks
  - *R has a specialized data type for missing values: NA*
  - *If you feed it a .csv that uses a different standard, you may need to do a bit of massaging. For instance, if you had a dot specifying missing values, and it wasn’t parsing in R as NA, you could try something like:*
    - *mutate(variable = case_when(variable == “.”~ NA))*
  - *(a little complicated for this workshop)*
- Are these replacement functions flexible enough to handle empty entries versus entries that have an explicit “null” or “none” from some prior export
  - *I believe not, but there are packages that do attempt to deal with this*

    - *You can also try running is.na(), which is a logical operator, and pass it to filter(). I'm not sure exactly which, but is.na() is robust to several different "types" of NAs*
- How is Mauro collapsing lines of script?
    - *The little arrows on the left next to the line numbers*
    - *Sorry, I didn't ask that well.*
    - *How does he bunch/ a number of rows into a single heading?*
    - *(like, the little arrow de-collapse. But how did he create it in the first place?*
    - *In RMarkdown, outside of "code chunks" is actually still a type of code. We sorta glossed over this due to time, but it's called "markdown" code (hence R and markdown). You can specify headings by # (for large headings) ## (for one size smaller) ### and so on. RStudio automatically recognizes each heading identified by these hashtags, and allows you to collapse them.*
    - *https://www.markdownguide.org/basic-syntax/*
- Why do we need list() in replace_na(list(abundance=0))?
    - *Because replace_na doesn't handle scalar values, it handles vectors or matrices -- so when you're replacing one value, you have to send it a 1x1 vector: https://tidyverse.tidyverse.org/reference/replace_na.html*
- How did Mauro create the output in a pdf and word option? I tried using the the knit, can't figure out. Thanks.
    - *Try with a new RMarkdown document.*
    - *Try removing the line at the top that says "outptu: github_document"*
    - *On rstudio.cloud it should work. Locally maybe not -- first you may need to install a .pdf tool. Error messages should guide you.*
- I'm used to working with a wide format so all of the data for each participant is in a single row. What are the benefits of using a long format?
    - *The rollup example we were discussing allows you to sort for a range of years by filtering on one column, whereas in the original dataset you would be searching for values on a number of columns -- you would have to construct all of your gdp searches according to gdp_percap_[YEAR]*
    - *The main benefit is you will be ready to go for using the tidyverse. ggplot2, for example, likes data in long format. But that should not stop you from structuring the data however you like -- you now know about pivot_longer() which you can use to create longer dataframes on the fly.*

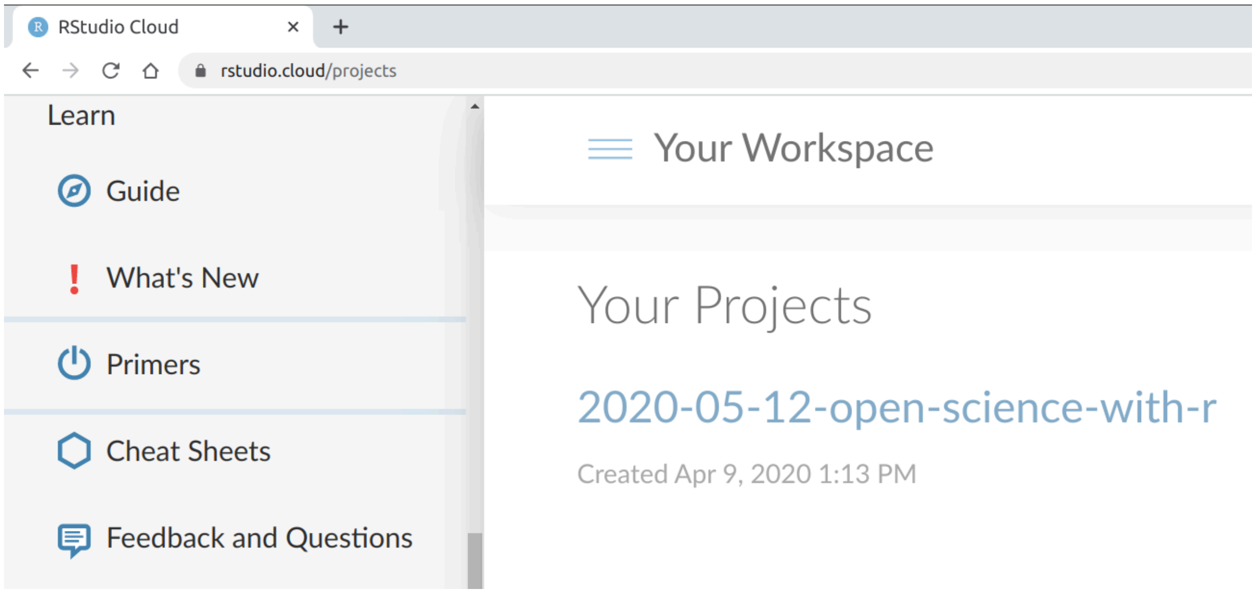
Wrap up

Use knit to create a report of your analysis; can export in various formats

Next

Learn more

- Come back: https://github.com/maurolepore/2020-05-12-open-science-with-r
- Similar workshop: https://github.com/rstudio-conf-2020/data-science-tidy
- Book: https://r4ds.had.co.nz/
- rstudio.cloud > Learn >
  - Primers (interactive tutorials)
  - Cheat Sheets



Community

- Twitter
- R-Ladies meetups: https://rladies.org/ladies-complete-list/locality/Houston/
- UseR meetups
- Rice (Miaomiao)
  - http://library.rice.edu/services/data-and-donuts-courses
  - http://library.rice.edu/data-assistance
  - http://library.rice.edu/research-data-services

Questions? / Quick feedback

- Great workshop! Very accessible intro to R. Good pacing.
- Thank you! Great workshop!
- Best workshop ever! I learned a lot in such a short period of time. Loved the style of the instructors and the pacing was perfect. Everything worked! Thank you so much!

- This was a great workshop, thank you! I really like that the program was divided into sections for data processing. Having the helpers answer questions was a good idea and a general Google docs so that the answers to the questions are written here.
- Thank you for the very productive sessions.