Tidybiology +DS: Session 2

An Introduction to Biological Data Science in R

Matthew Hirschey, Ph.D.

April 8-9, 2020

Welcome back

“The simple graph has brought more information to the data analyst’s mind than any other device.” – John Tukey

Visualize Data with

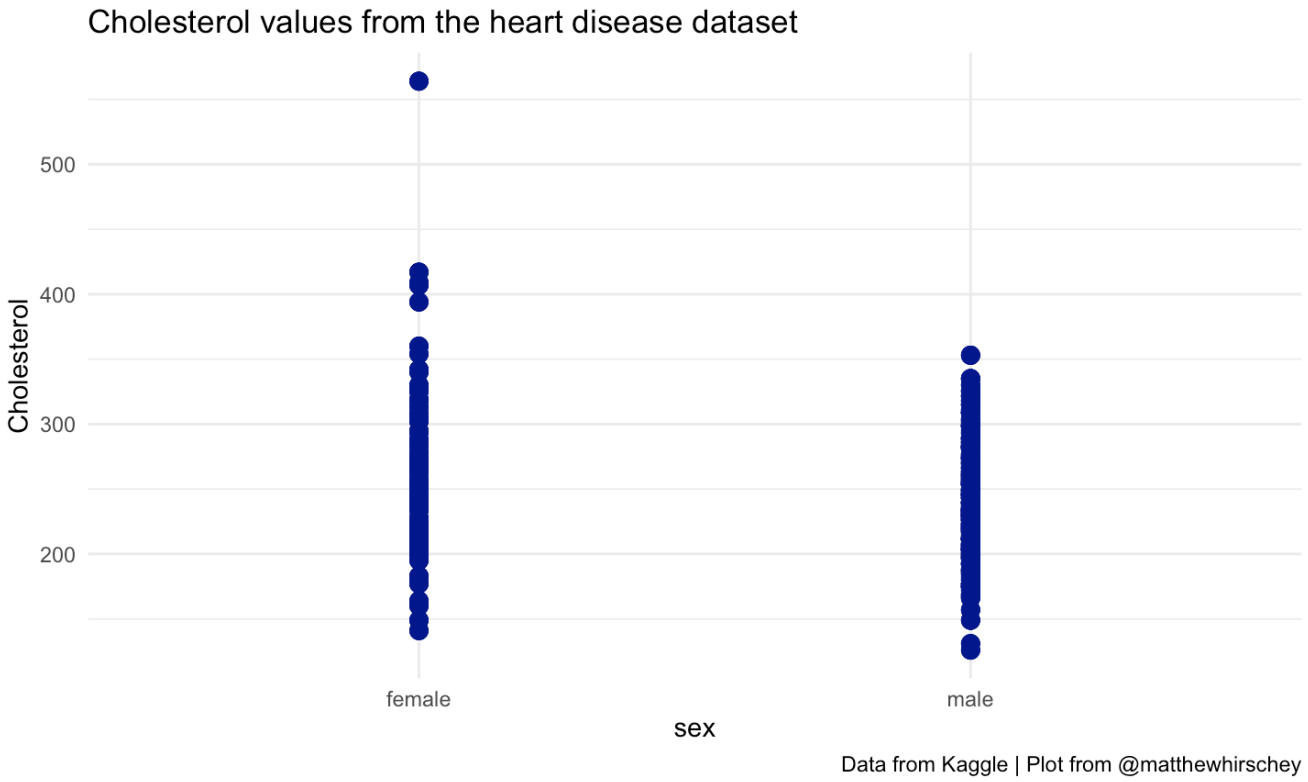
https://ggplot2.tidyverse.org (https://ggplot2.tidyverse.org)

A basic plot using `ggplot` in R

We ended the last class with:

```
ggplot(heart, aes(x = sex, y = chol)) +
  geom_point(color = "darkblue", size = 3) +
  labs(x = "sex",
       y = "Cholesterol",
       title = "Cholesterol values from the heart disease dataset",
       caption = "Data from Kaggle | Plot from @matthewhirschey") +
  theme_minimal() +
  NULL
```

which built this…



Any questions from last week?

Basic ggplot2

Basics of a ggplot code

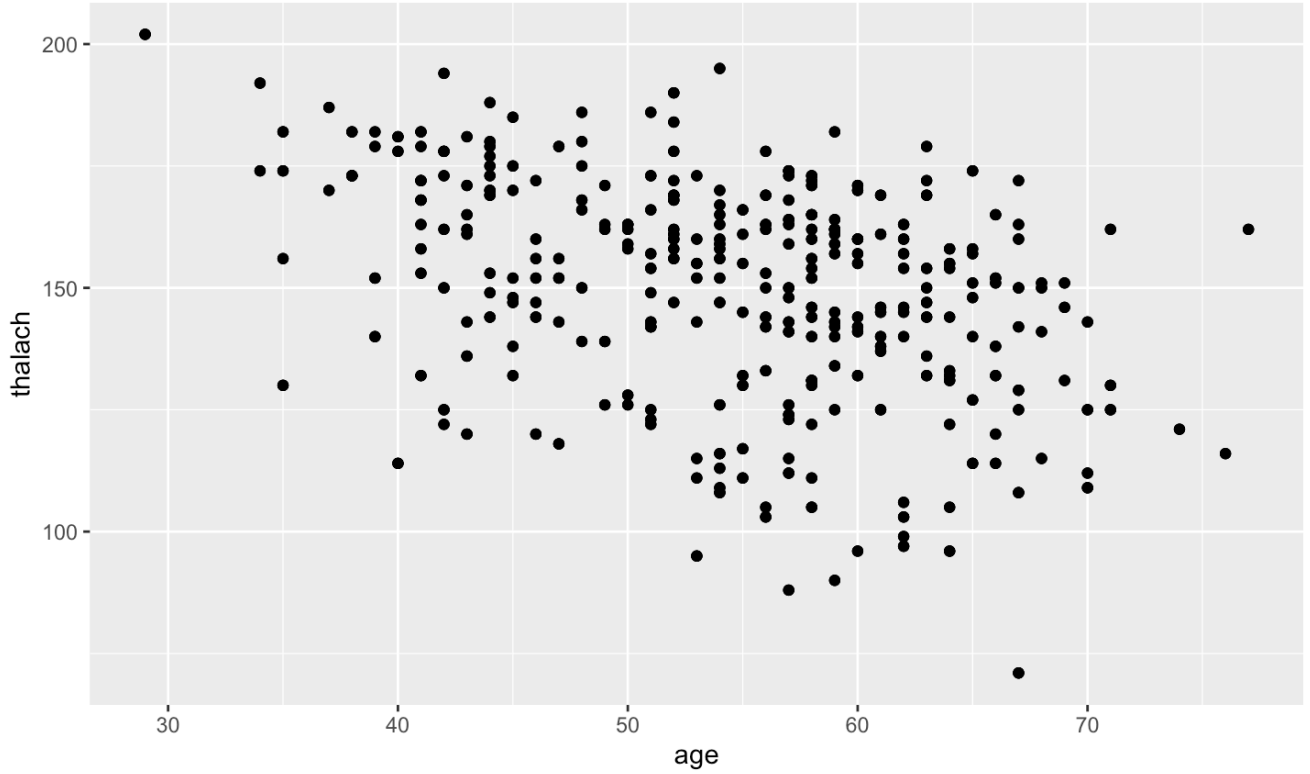
Below is an example of the most basic form of the ggplot code

```
ggplot(data = dataframe) +
geom(mapping = aes(x, y))
```

Take a moment to look back at the code template. You can see that in that code we assigned a `dataset` and the information we needed to `map` it to a type of plot

Basics of a ggplot code

```
ggplot(data = heart) +
geom_point(mapping=aes(x = age, y = thalach))
```



Build a `ggplot`

- "Initialize" a plot with `ggplot()`

```
ggplot(data = heart) +
```

- Add layers with `geom_functions`

```
geom_point(mapping=aes(x = age, y = thalach))
```

Notes on `ggplot` style

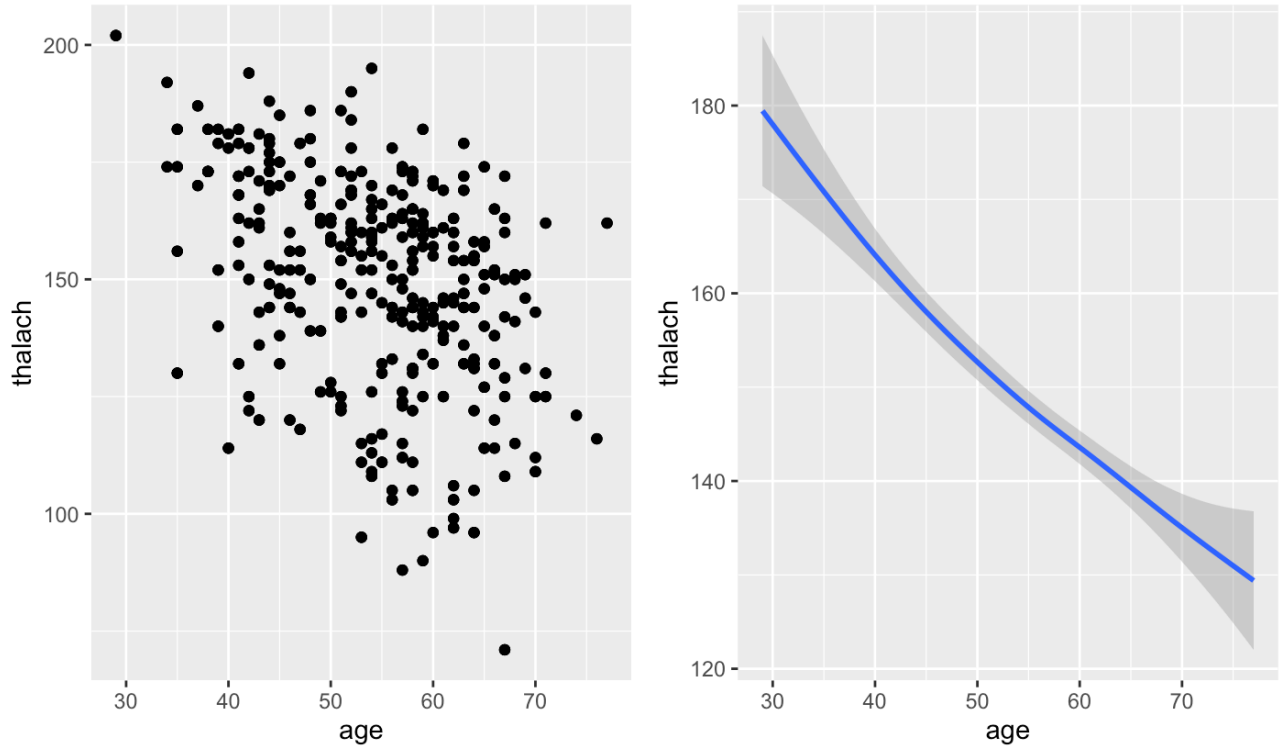
- Generally, different people have **strong** opinions about style and data visualization

- Data visualization is a rich and complex area of study and is beyond the scope of this introductory course

That being said, here are a few style tips:
- While you can put the + at the beginning of the next line, it is generally put at the end of the previous line

- Arguments do not need to be explicit: this chunk of code is equivalent to the previous chunk (data, mappings) `ggplot(heart) + geom_point(aes(x = thalach, y = thalach))`

How are these two plots similar?



The `geom` is different between these plots

`geom` is short for geometric object, which is the visual object used to represent the data

```
plot1 <- heart %>%
ggplot(aes(age, thalach)) +
geom_point()
```

```
plot2 <- heart %>%
ggplot(aes(age, thalach)) +
geom_smooth()
```

Choosing your (geom)itries

Different data types require different plot types.

When plotting your data, it is often helpful to take a glimpse at the data you intend to plot to know what kinds of variables you will be working with

```
glimpse(heart)
```

Code | Start Over | Run Code

```
1
2
3
```

Geoms

So now that you know your variable types, how do you know what geoms to use??

Use the following resources to match your data type to the appropriate `geoms`

https://rstudio.com/resources/cheatsheets/
(https://rstudio.com/resources/cheatsheets)

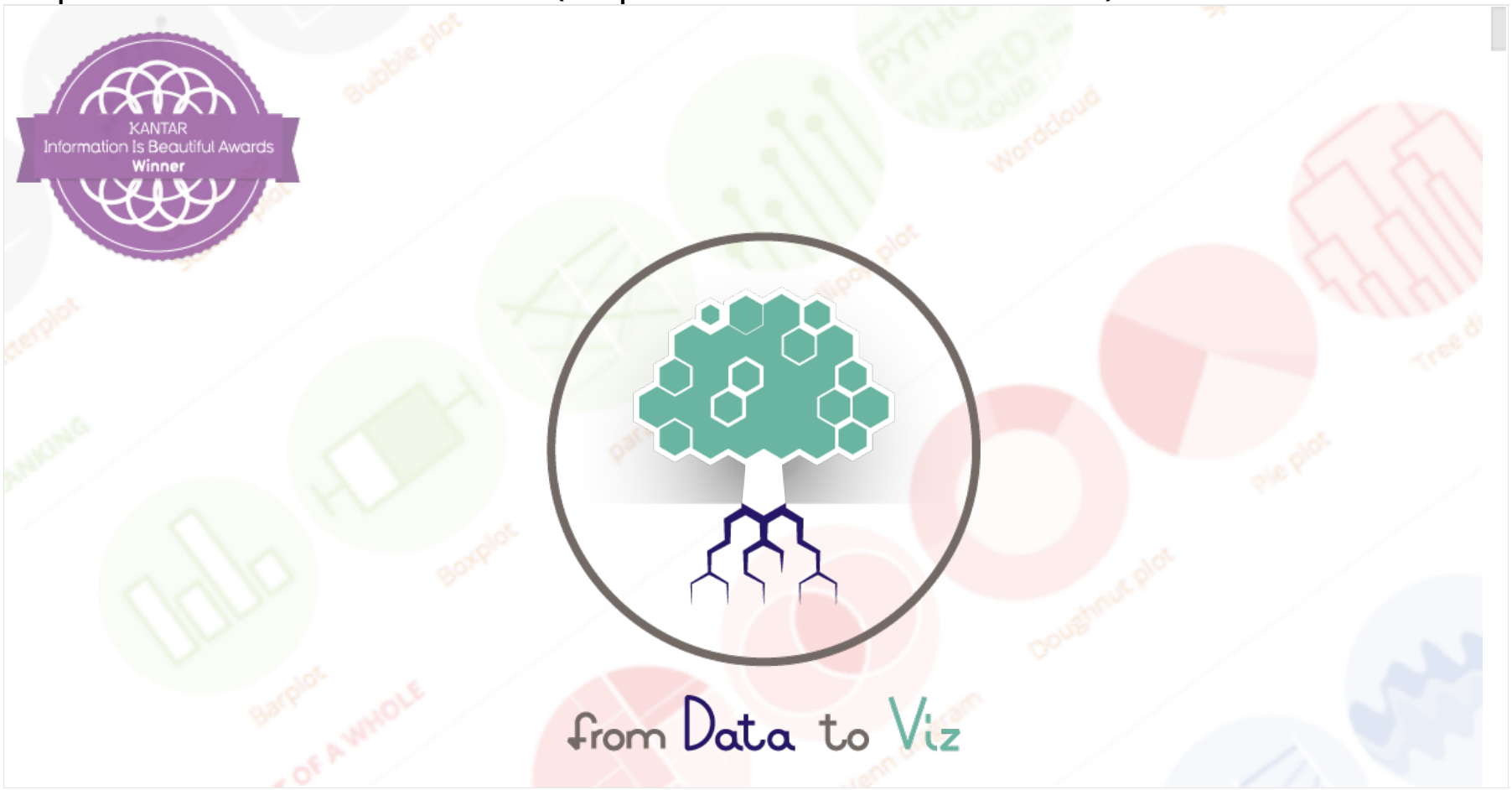
RStudio Cheatsheets

The cheatsheets below make it easy to use some of our favorite packages. From time to time, we will add new cheatsheets. If you'd like us to drop you an email when we do, click the button below.

SUBSCRIBE TO CHEATSHEET UPDATES

CONTRIBUTED CHEATSHEETS

https://www.data-to-viz.com/ (https://www.data-to-viz.com/)



Change this scatterplot code to draw boxplots

```
ggplot(heart) + geom_point(aes(x=sex,y=age))
```

Code | Start Over | Run Code

```
1
2
3
```

Use the cheatsheet. Try your best guess.

What will this code do?

```
ggplot(heart) +
geom_boxplot(aes(x=sex,y=age)) +
geom_point(aes(x=sex,y=age))
```

Try including two `geoms` for yourself

Code | Start Over | Run Code

```
1
2
3
```

Each new geom adds a new layer

Aesthetics

Everything up to this point gets you a basic graph; but what about colors, shapes and overall style?

You can change 5 basic aesthetics
1. **Color**- changes the outline color of your datapoints
2. **Size** - choose the size of the datapoint
3. **Shape** - choose a pre-defined shape
4. **Alpha**- changes the transparency of each point
5. **Fill**- changes the fill color of your points

Go to code/
Open 04_ggplot2.Rmd
Complete the exercise.

Mapping these aesthetics to data

Beyond simply changing the size or color of the variables in your plot, you can encode more information by mapping these values to data in your data set.

Go to code/
Open 05_aes.Rmd
Complete the exercise.

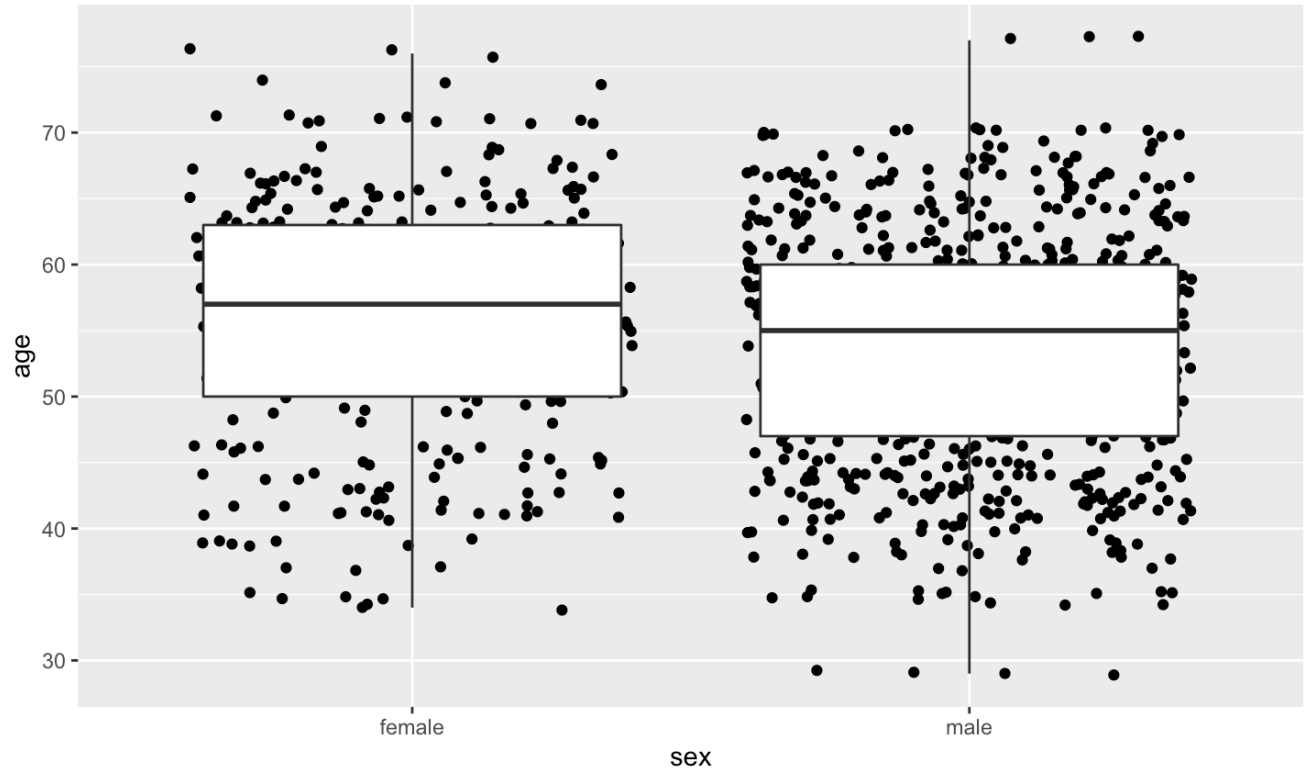
Global vs Local

In ggplot2, we have the options to set mappings globally or locally. Setting a mapping globally means to set those values in the original ggplot function.

Example: Earlier in class you made this graph:

```
ggplot(heart) +
geom_jitter(aes(x=sex, y=age))+
geom_boxplot(aes(x=sex, y=age))
```

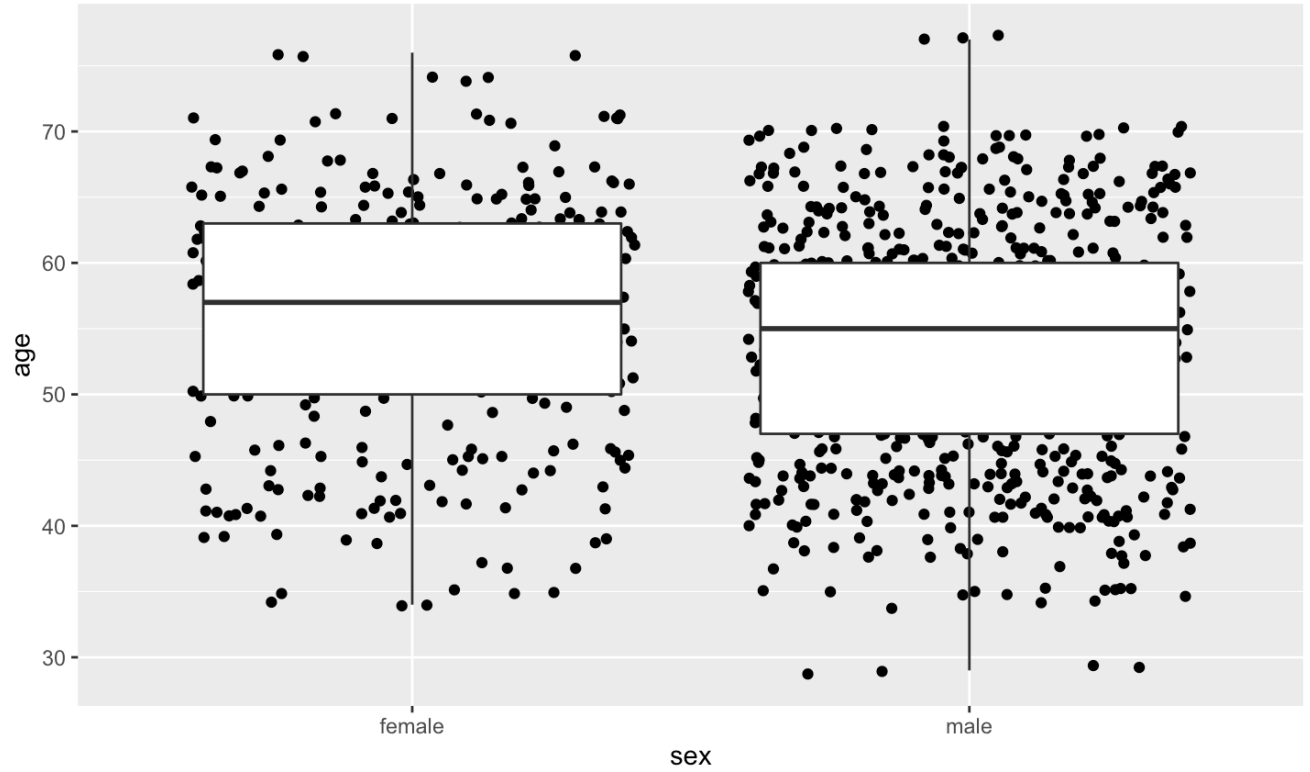
Global vs Local



Global vs Local

However, if we map our x and y values in the ggplot function we find that we generate the same graph

```
ggplot(heart, aes(x=sex, y=age) +
geom_jitter() +
geom_boxplot()
```



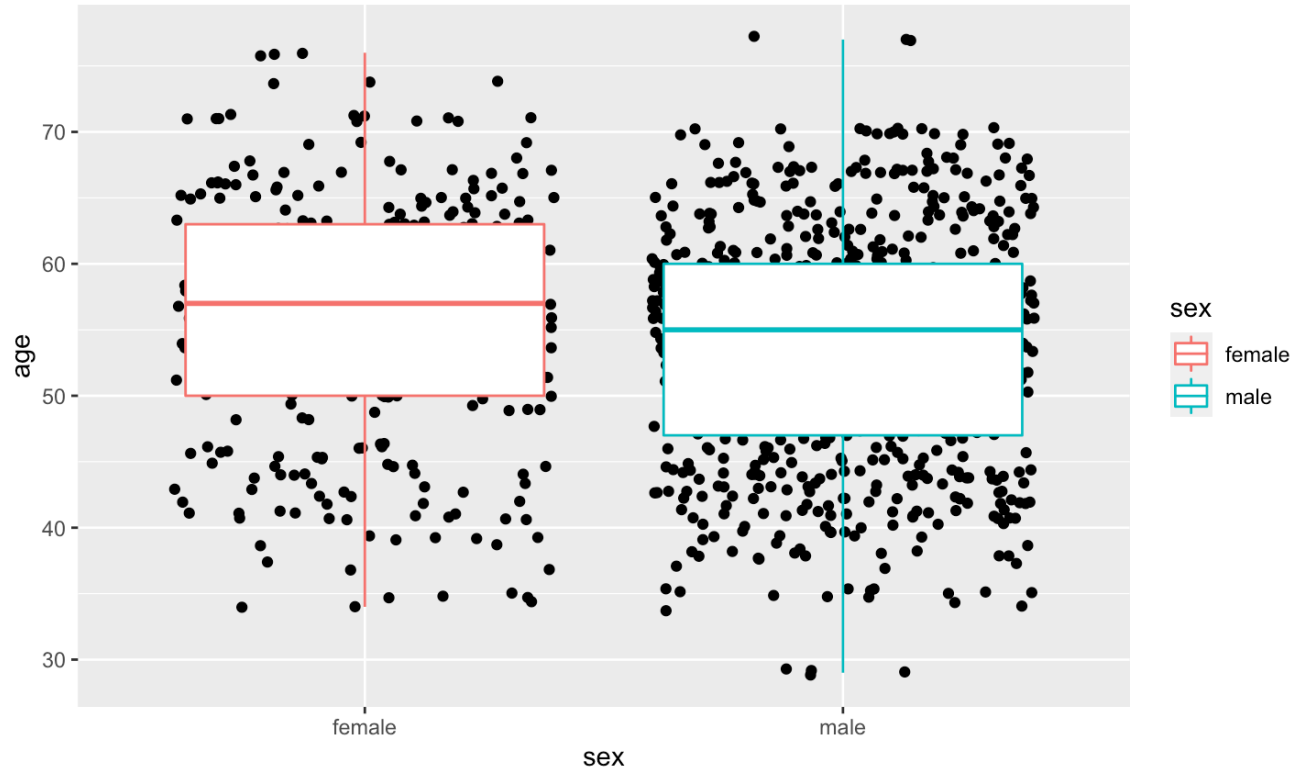
Global vs Local

This is because when you set the aes mappings in the original `ggplot` function you are setting the `aes` **globally**.
This means all the functions afterwards will inherit that mapping. So in our example, this means that both the jitter and boxplot geoms know to graph the same information

You can also set aes values **locally** within the geom function. Doing so will only change the values in that geom

Global vs Local

```
ggplot(heart, aes(x=sex, y=age) +
geom_jitter() +
geom_boxplot(aes(color = sex))
```



Global vs Local

Data can also be set locally or globally. For this example, let's filter our original data first using the `dplyr::filter` function

```
df_filter <- heart %>% filter(thalach > 195)
```

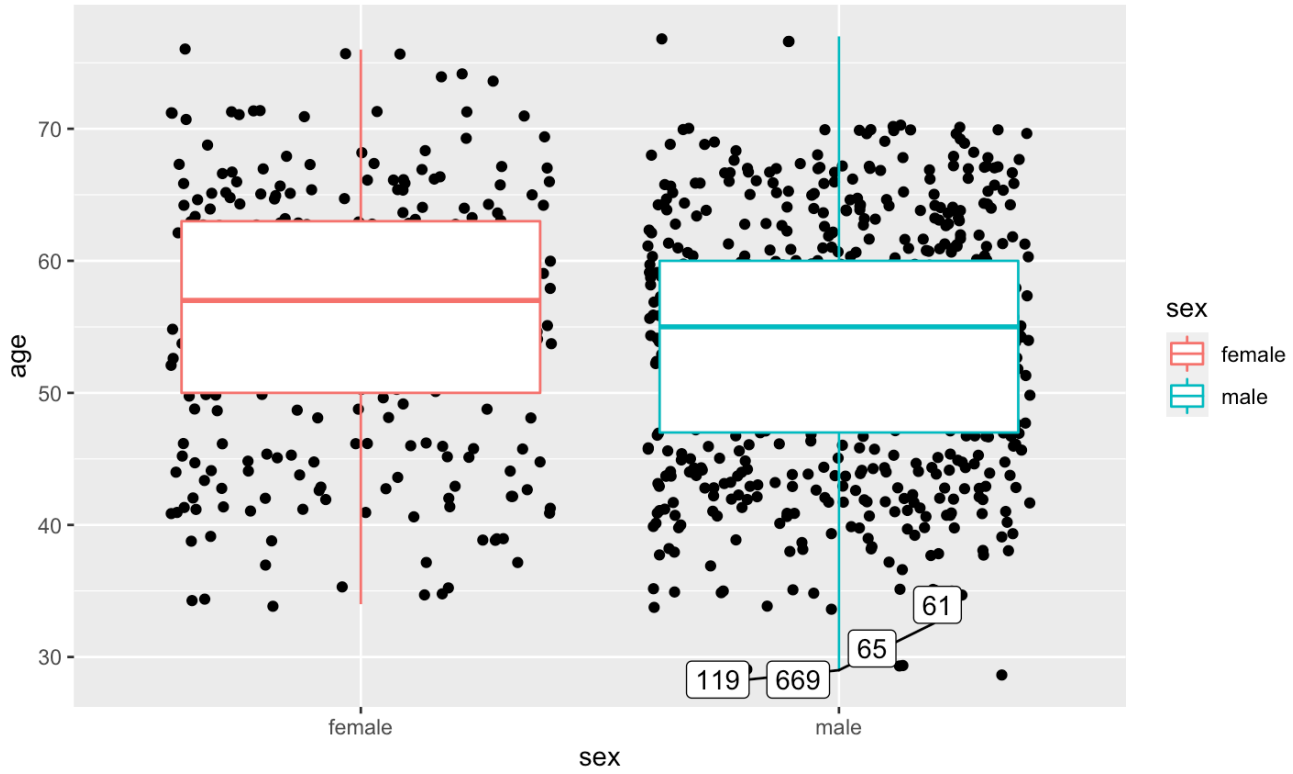
*this number is two standard deviations above the `mean` value of thalach

Global vs Local

Now, let's identify only the patients in our data that are outliers, more than 2SD above the mean, by setting data locally in a new geom

```
ggplot(heart, aes(x=sex, y=age) +
geom_jitter() +
geom_boxplot(aes(color = sex)) +
geom_label(data=df_filter, aes(label=patient_id))
```

Global vs Local



Global vs Local

You notice we have to indicate the new dataset, but because it has the same x and y values, we did not need to set those mappings

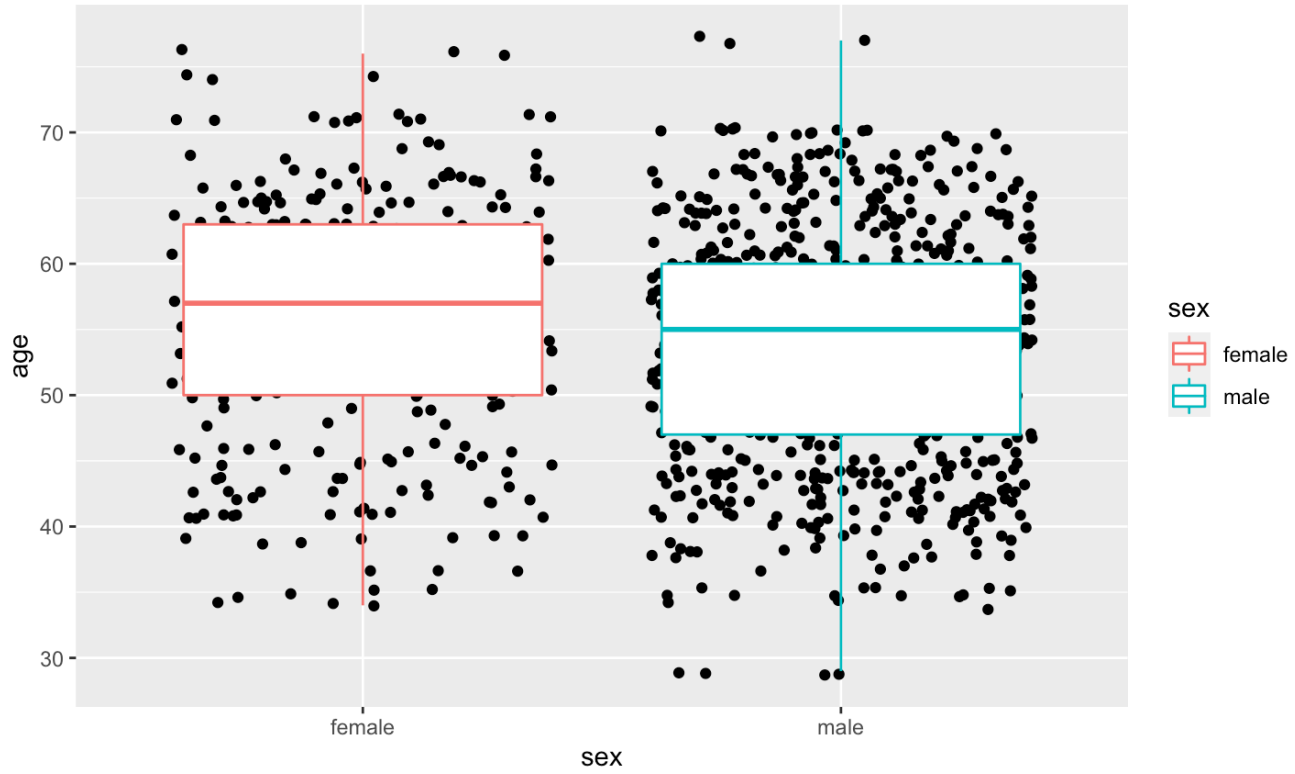
Go to code/
Open 06_global_v_local.Rmd
Complete the exercise to practice mapping locally and globally.

Labels and Legends

Several options exist to change the default labels and legends. Recall, this code:

```
ggplot(heart, aes(x=sex, y=age) +
geom_jitter() +
geom_boxplot(aes(color = sex))
```

Labels and Legends



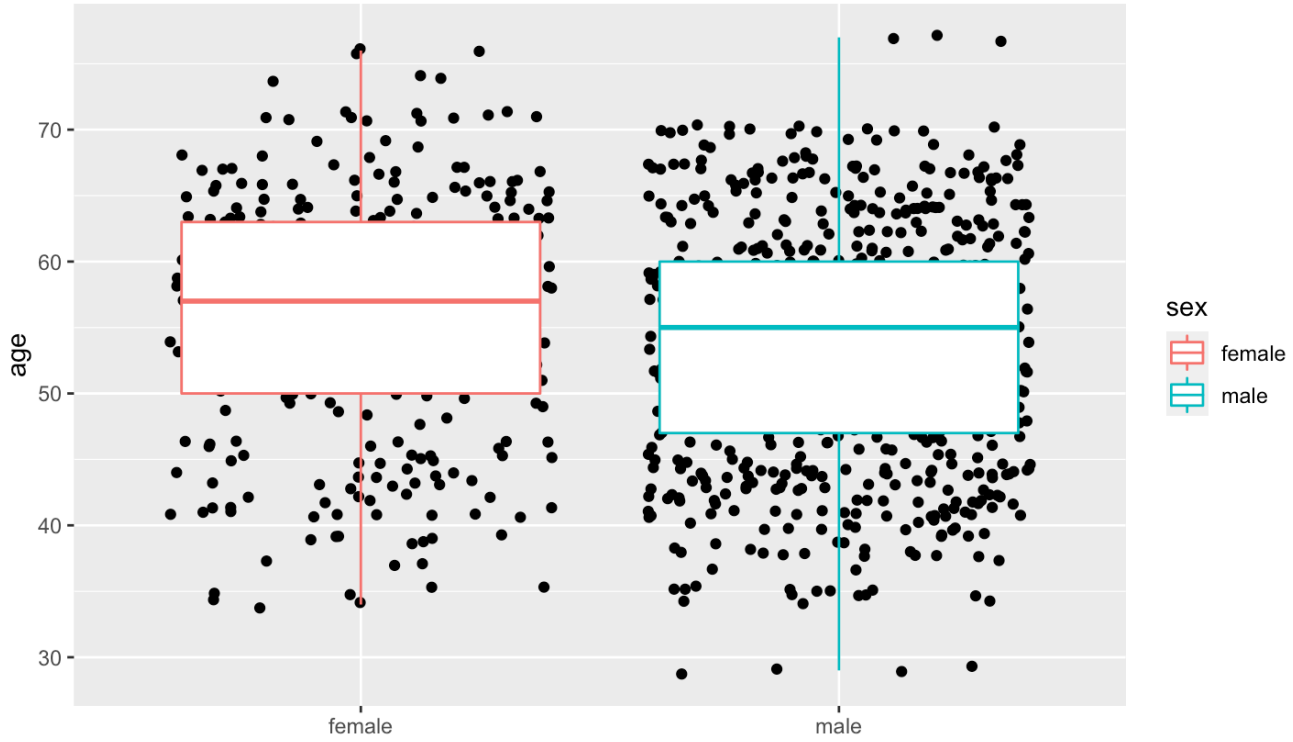
But it has two problems:
1. The x-axis label is redundant
2. The figure legend is also redundant

Change labels using `labs`

```
ggplot(heart, aes(x=sex, y=age) +
geom_jitter() +
geom_boxplot(aes(color = sex)) +
labs(x ="") #blank quotes removes the label
```

Change labels using `labs`

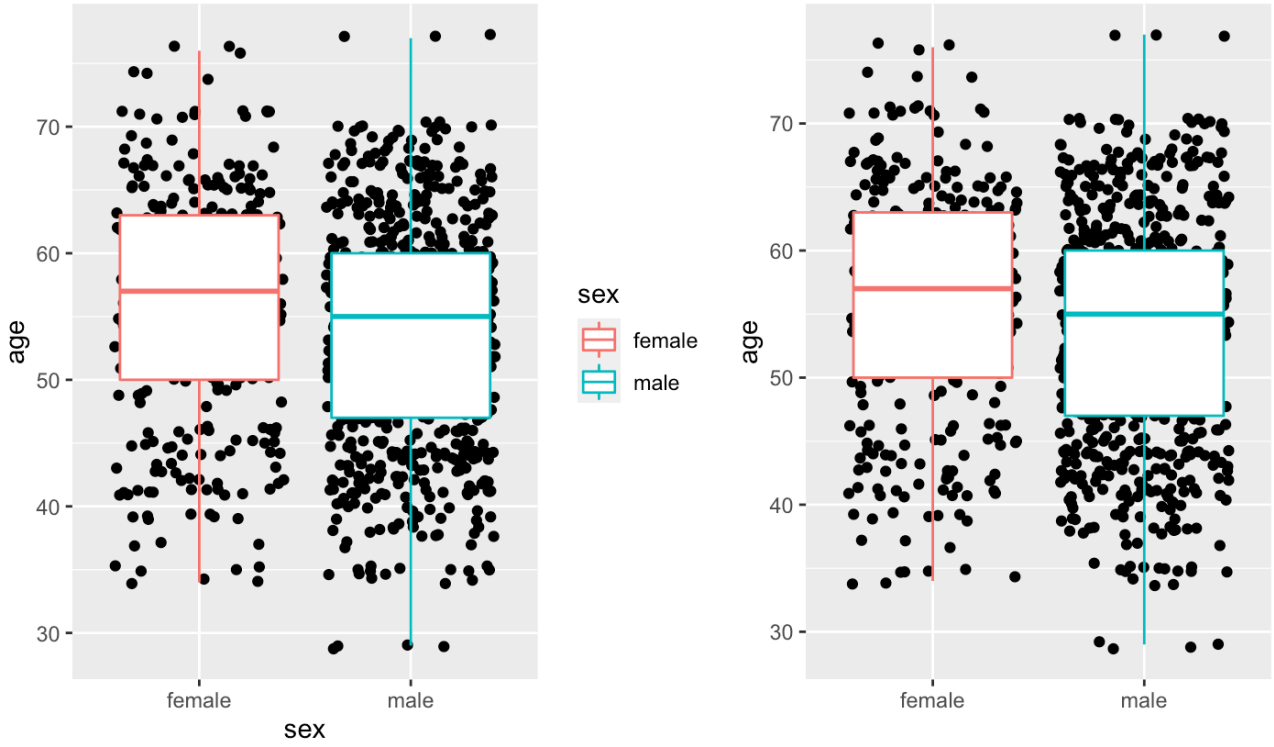
Gave us this plot:



Change legend using `guides`

```
ggplot(heart, aes(x=sex, y=age) +
geom_jitter() +
geom_boxplot(aes(color = sex)) +
labs(x ="") #blank quotes removes the label +
guides(color = "none")
```

Change legend using `guides`



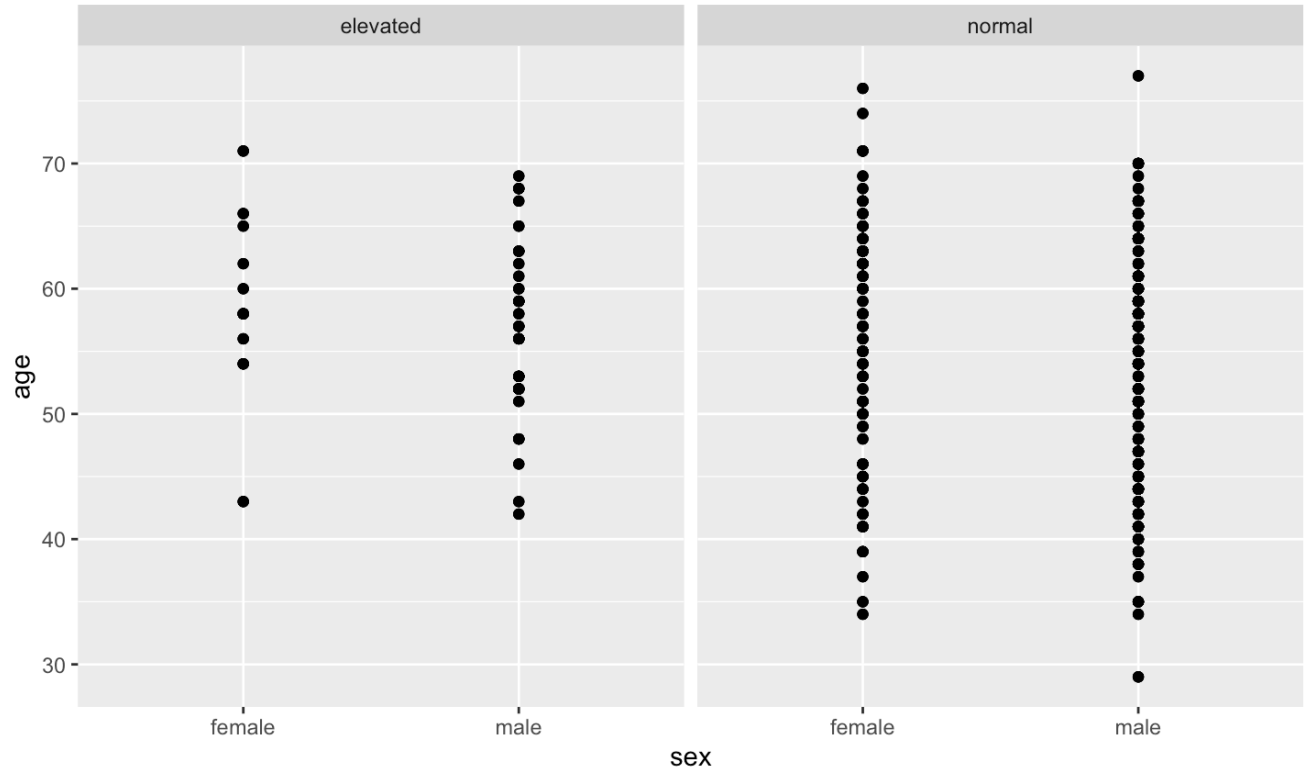
Faceting

Faceting allows you to create multiple graphs side by side in one panel. Especially useful when you want to see the data together, but not on top of each other

For example:

```
ggplot(heart) +
geom_point(aes(x=sex, y=age)) +
facet_grid(cols = vars(fbs))
```

Faceting



*This is especially useful for exploratory data analysis

Themes

You can change almost everything you see on your chart, but a lot of the things you may look to change are part of the “theme”

Here we are going to change some features about our title text:

```
ggplot(heart, aes(x=sex, y=age) +
geom_jitter() +
geom_boxplot(aes(color = sex)) +
labs(title = "My first plot") +
theme(plot.title = element_text(face = "bold", size = 12))
```

Themes

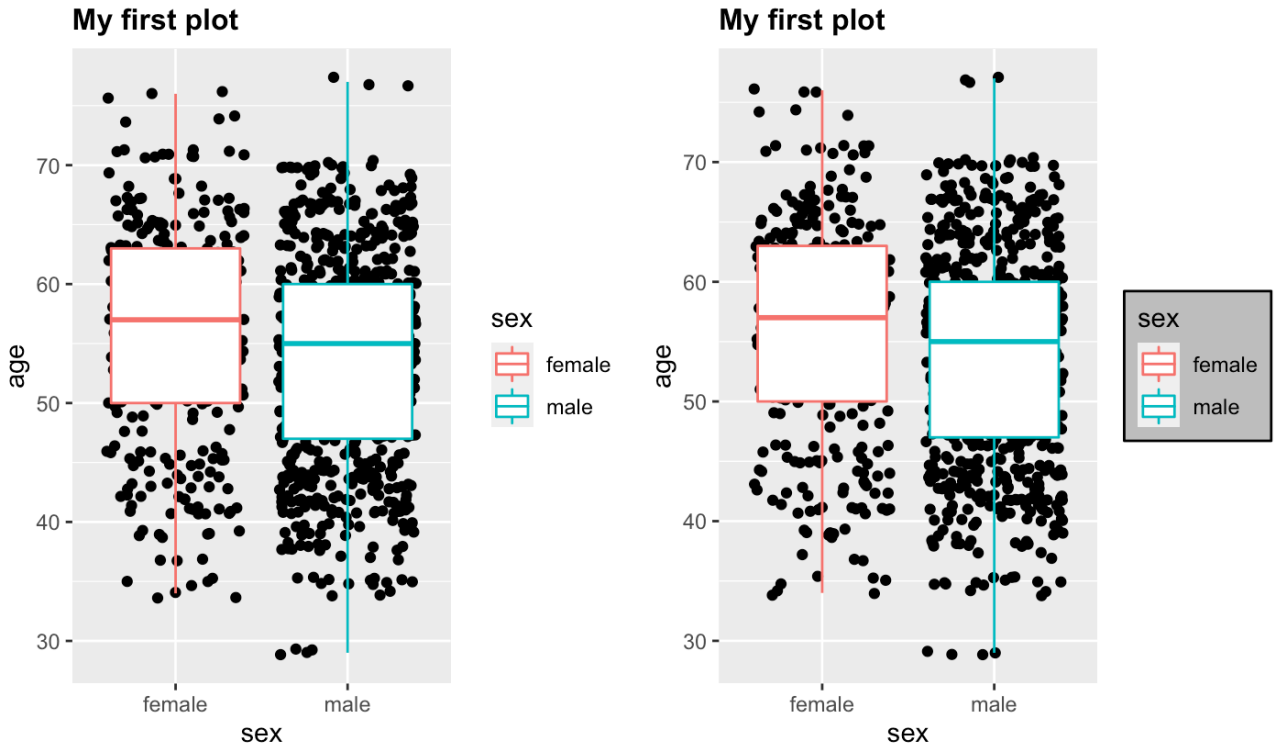

My first plot
70
60
50
40
30
age
female
male
sex
sex
female
male
70
60
50
40
30
age
female
male
sex
sex
female
male

Themes

Next, let's change the aesthetics of our legend box

```
ggplot(heart, aes(x=sex, y=age) +
geom_jitter() +
geom_boxplot(aes(color = sex)) +
labs(title = "My first plot") +
theme(plot.title = element_text(face = "bold", size = 12),
legend.background = element_rect(fill="gray", colour="black"))
```

Themes

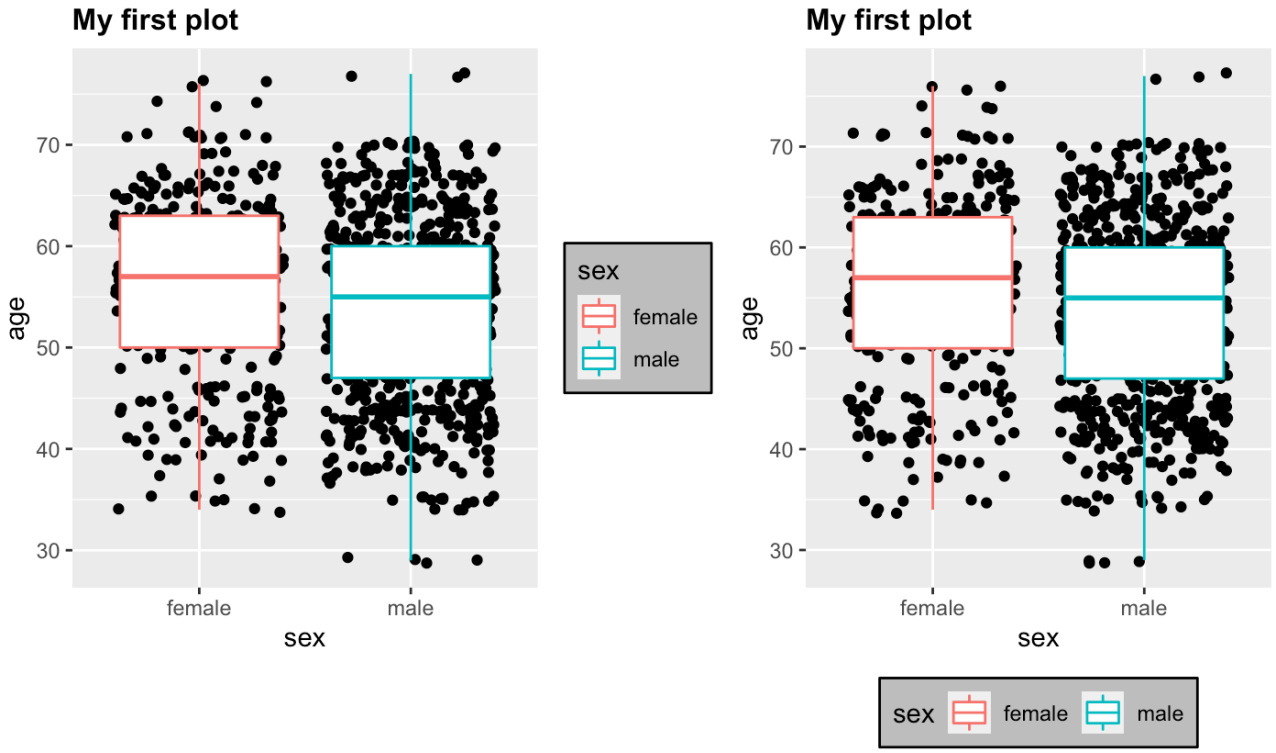

My first plot
age
70
60
50
40
30
female
male
sex
sex
female
male
My first plot
age
70
60
50
40
30
female
male
sex
sex
female
male

Themes

Finally, let's change the legend position

```
ggplot(heart, aes(x=sex, y=age) +
geom_jitter() +
geom_boxplot(aes(color = sex)) +
labs(title = "My first plot") +
theme(plot.title = element_text(face = "bold", size = 12),
legend.background = element_rect(fill="gray", colour="black"),
legend.position = "bottom"))
```

Themes

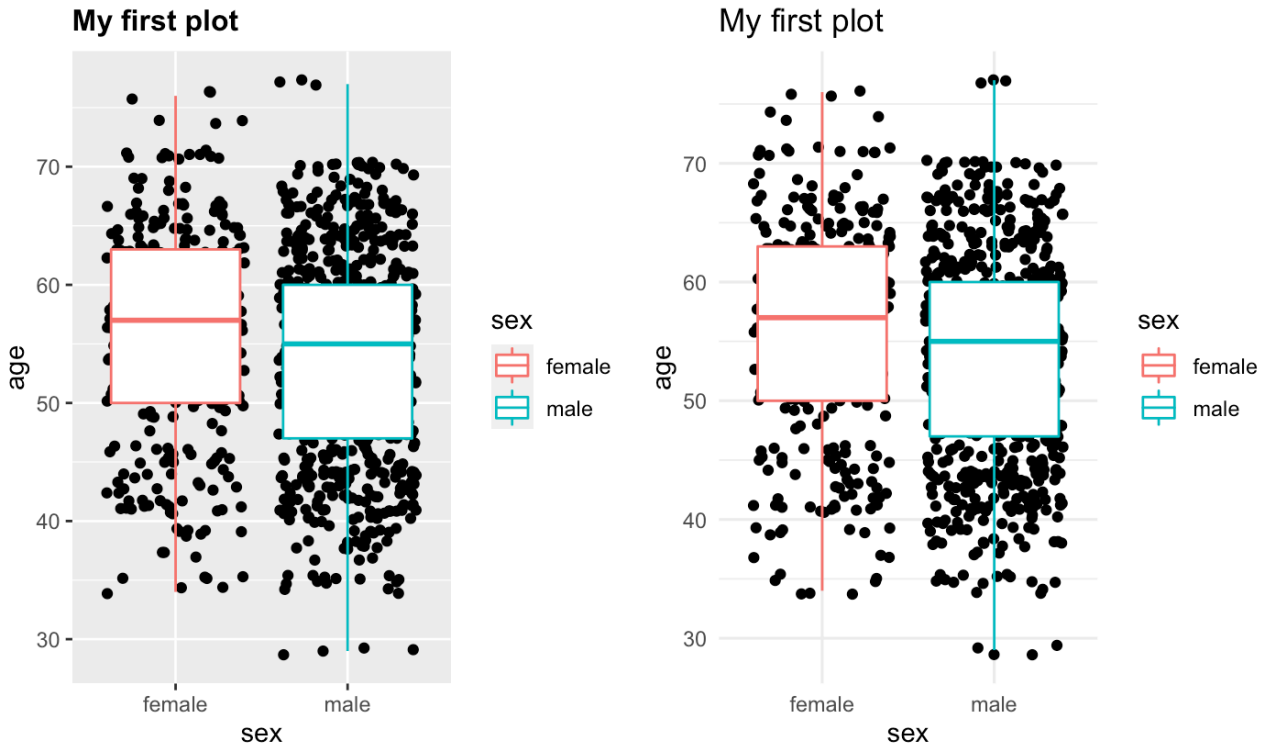

My first plot
age
70
60
50
40
30
female
male
sex
sex
female
male
My first plot
age
70
60
50
40
30
female
male
sex
sex
female
male

Themes

Pre-set themes also exist as an easy way to change the entire theme of your graph quickly. They can also be combined with custom theme settings

```
ggplot(heart, aes(x=sex, y=age) +
geom_jitter() +
geom_boxplot(aes(color = sex)) +
labs(title = "My first plot") +
theme_minimal()
```

Themes


My first plot
70
60
50
40
30
age
female
male
sex
sex
female
male
My first plot
70
60
50
40
30
age
female
male
sex
sex
female
male

Saving plots using `ggsave`

If you make a plot there are a few ways to save it, though the simplest is to use `ggsave`

```
ggsave("ggsaveexample.png", plot = last_plot())
```

You can change the type of file you save or the size.

```
ggsave("ggsaveexample.pdf", plot = my_awesome_object, width = 6,
height = 6, units = "cm")
```

Saving plots using ggsave

Where does it save??

```
getwd()
```

Putting it all together

Go to code/
Open 07_ggplot_together.Rmd
Complete the exercise to put all these ggplot skills to work.

Any questions?

R
Importing

Importing data into R

The `readr` package (found in the `tidyverse` collection) contains a number of useful functions of the form `read_*` to import data. For example, if you have a `.csv` file, you would use the `read_csv` function

The dataset provided to you is a cleaned R-specific document. But you will never find this in 'the wild'.

Most often, you will need to find a data file (such as csv), and import it

Importing data into R

For the purpose of this class, we have generated a **simulated** dataset of pseudo-patient histories to accompany the heart dataset.

To import the phx.csv file into RStudio, run the following:

```
patient_hx <- read_csv(here::here("data", "phx.csv"))
```

Go to code/
Open 08_import_and_join.Rmd
Complete the exercise to import this new dataset.

Importing data into R

You can also use the `readr` package to import data from a URL

For example, to load a dataset from a URL, run the following

```
path <- here::here("data", "phx.csv")
url <- "https://raw.githubusercontent.com/matthewhirschey/tidybiology-plusds/master/data
/phx.csv"
patient_hx <- read_csv(url)
```

Combining datasets

There are many times when you have two or more overlapping datasets that you would like to combine

The `dplyr` package has a number of `*_join` functions for this purpose

`left_join`

Returns all rows from a, and all columns from a and b

Rows in a with no match in b will have NA values in the new columns

If there are multiple matches between a and b, all combinations of the matches are returned

`left_join` example

Take a look at the variables in each dataset - heart and patient_hx

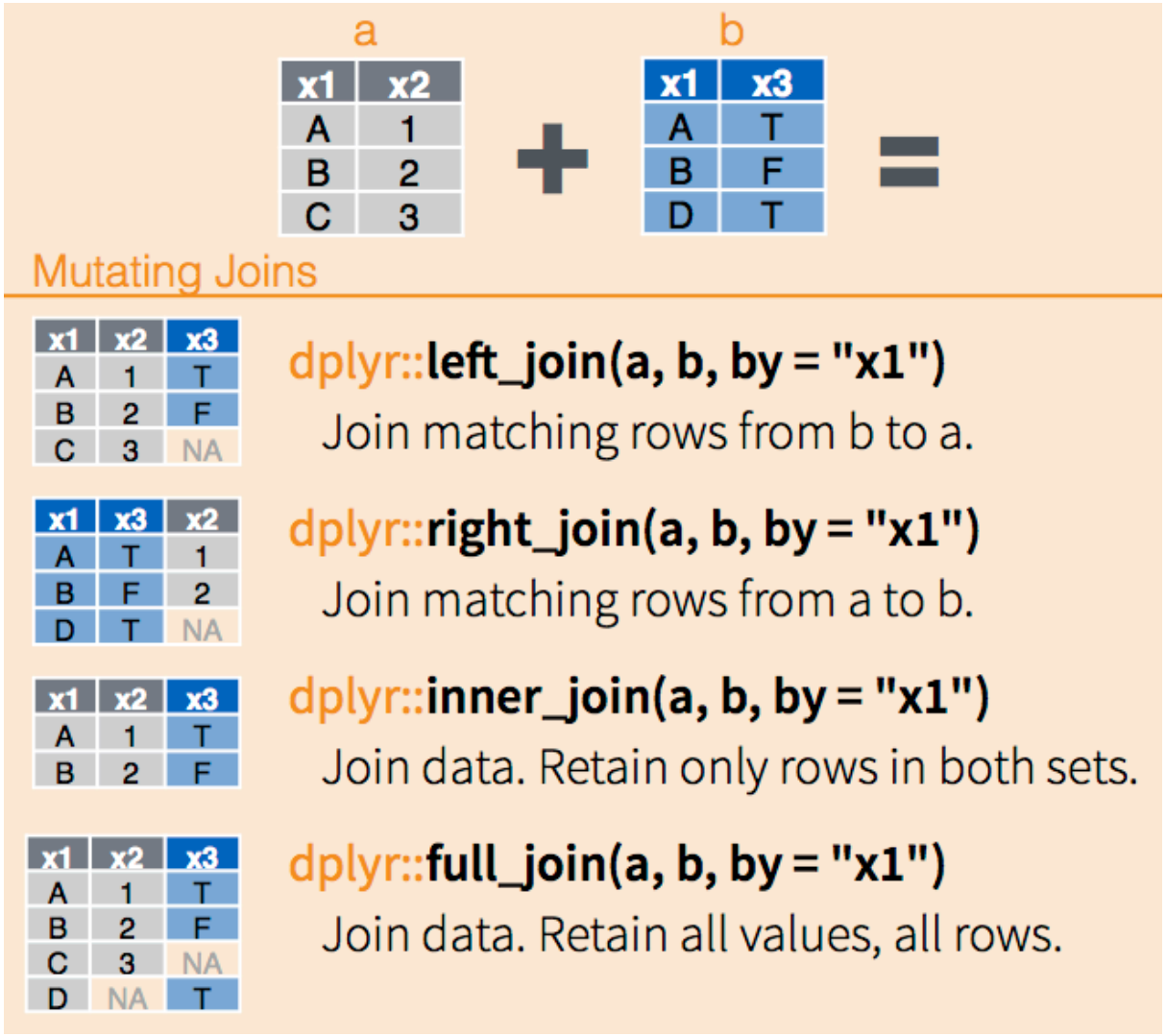
You will notice that both datasets contain common variable - patient_id. This can therefore serve as a common variable to join on. Let's join on this:

`left_join` heart with patient_hx and assign the output to a new object called `heart_join_left`

Go to code/
Open 08_import_and_join.Rmd
Complete the exercise to join the two datasets.

Now you have one dataset with additional useful information

right_join


a
x1 x2
A 1
B 2
C 3
b
x1 x3
A T
B F
D T
Mutating Joins
x1 x2 x3
A 1 T
B 2 F
C 3 NA
dplyr::left_join(a, b, by = "x1")
Join matching rows from b to a.
x1 x3 x2
A T 1
B F 2
D T NA
dplyr::right_join(a, b, by = "x1")
Join matching rows from a to b.
x1 x2 x3
A 1 T
B 2 F
dplyr::inner_join(a, b, by = "x1")
Join data. Retain only rows in both sets.
x1 x2 x3
A 1 T
B 2 F
C 3 NA
D NA T
dplyr::full_join(a, b, by = "x1")
Join data. Retain all values, all rows.

`right_join`

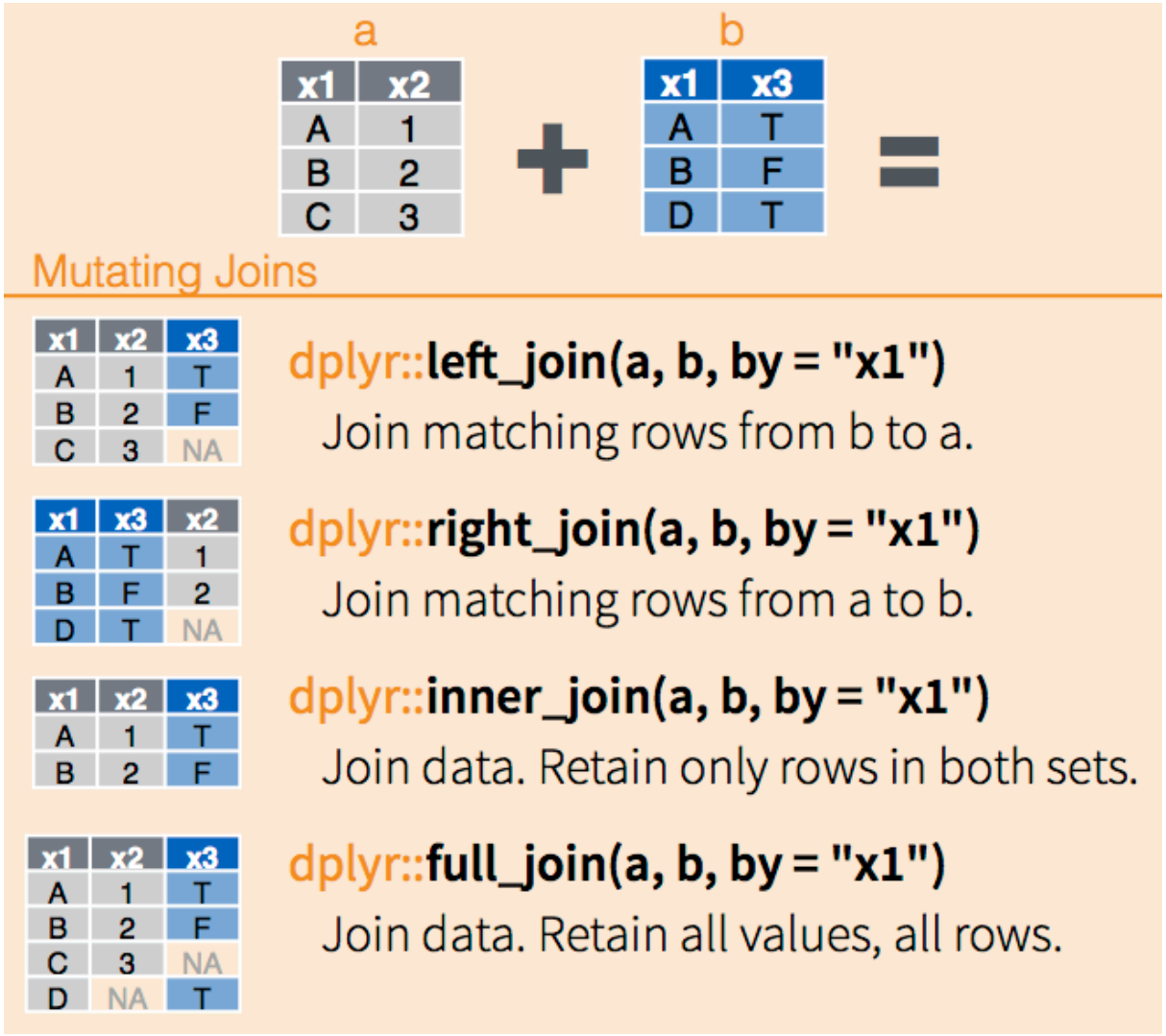
Returns all rows from b, and all columns from a and b

Rows in b with no match in a will have NA values in the new columns

If there are multiple matches between a and b, all combinations of the matches are returned

This is conceptually equivalent to a `left_join`, but can be useful when stringing together multiple steps using %>%

inner_join

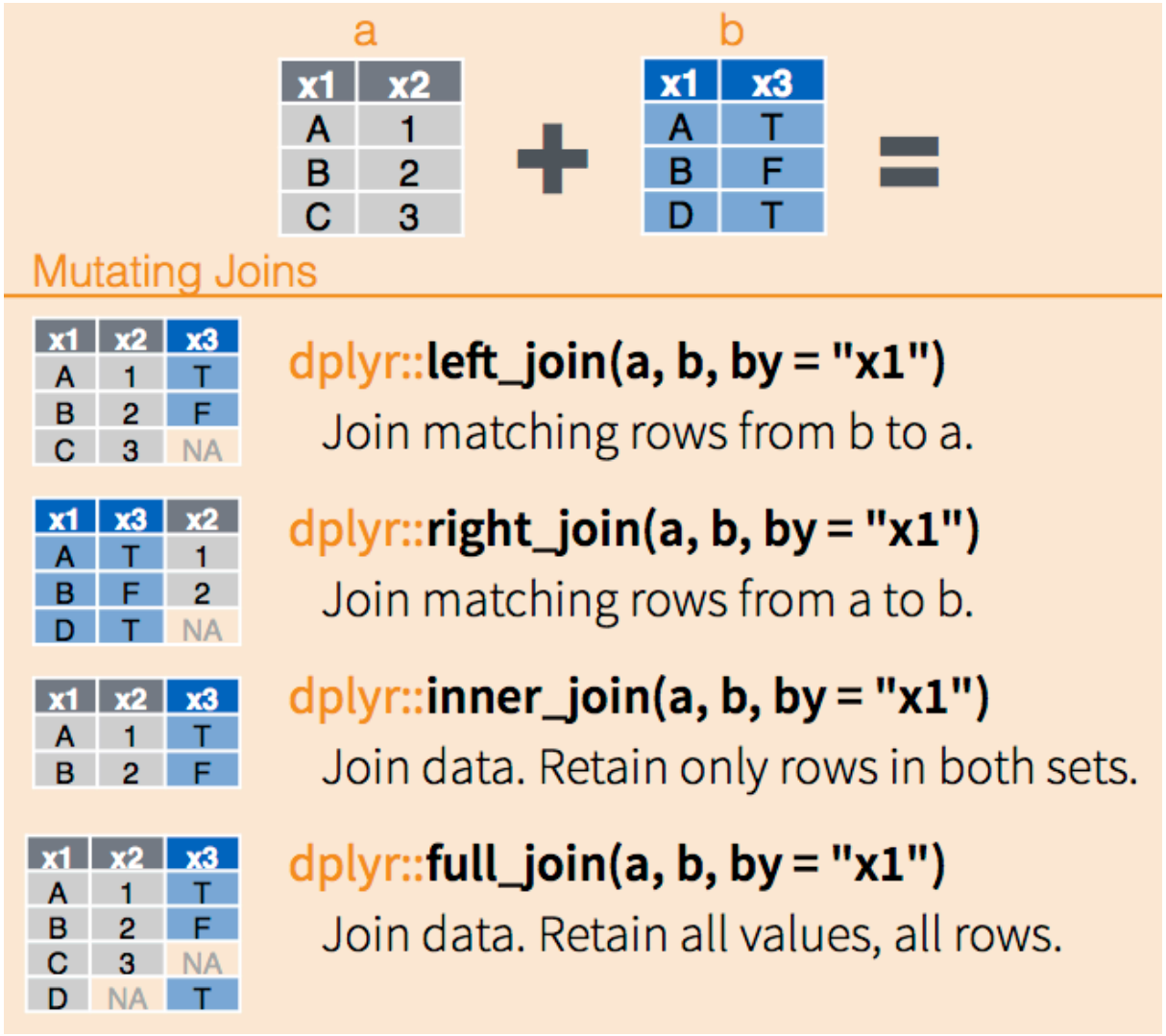

a
x1 x2
A 1
B 2
C 3
b
x1 x3
A T
B F
D T
Mutating Joins
x1 x2 x3
A 1 T
B 2 F
C 3 NA
dplyr::left_join(a, b, by = "x1")
Join matching rows from b to a.
x1 x3 x2
A T 1
B F 2
D T NA
dplyr::right_join(a, b, by = "x1")
Join matching rows from a to b.
x1 x2 x3
A 1 T
B 2 F
dplyr::inner_join(a, b, by = "x1")
Join data. Retain only rows in both sets.
x1 x2 x3
A 1 T
B 2 F
C 3 NA
D NA T
dplyr::full_join(a, b, by = "x1")
Join data. Retain all values, all rows.

`inner_join`

Returns all rows from a where there are matching values in b, and all columns from a and b

If there are multiple matches between a and b, all combination of the matches are returned

full_join


a
x1 x2
A 1
B 2
C 3
b
x1 x3
A T
B F
D T
Mutating Joins
x1 x2 x3
A 1 T
B 2 F
C 3 NA
dplyr::left_join(a, b, by = "x1")
Join matching rows from b to a.
x1 x3 x2
A T 1
B F 2
D T NA
dplyr::right_join(a, b, by = "x1")
Join matching rows from a to b.
x1 x2 x3
A 1 T
B 2 F
dplyr::inner_join(a, b, by = "x1")
Join data. Retain only rows in both sets.
x1 x2 x3
A 1 T
B 2 F
C 3 NA
D NA T
dplyr::full_join(a, b, by = "x1")
Join data. Retain all values, all rows.

`full_join`

Returns all rows and all columns from both a and b

Where there are no matching values, returns NA for the one missing

Any questions?

R Strings

Strings hold text

A string is what we store text within. It can be either:
- A single word: `"awesome"`
- A sentence: `"this class is awesome"`
- A combination: `c("blue", "is my favorite", "color")`

Any of these can be stored as object, which we call strings.

Code | Start Over | Run Code

```
1
2
3
```

Dealing with strings

Dealing with character strings is a bit different than dealing with numbers in R?

- Sort?
- Select?
- Change?

Fortunately, the `tidyverse` as a package called `stringr` for dealing with them.

Stringr

<iframe src = "https://stringr.tidyverse.org (https://stringr.tidyverse.org)>

Counting string

`str_count()` is a function we can use to count the number of rows that match a particular pattern.

The output with either be 1 (match), or 0 (no match)

Example
In this code:
- string we want to evaluate is `heart_joined$health_status`
- pattern we want to count `"High Cholesterol"`

```
str_count(heart_joined$health_status, "High Cholesterol")
```

Counting

```
str_count(heart_joined$health_status, "High Cholesterol")
```

Summarizing our counts

A bunch of 0 and 1 are not incredibly useful.

But since R is good at adding, we can simply wrap the previous expression in `sum()`

Try it below:

Code | Start Over | Run Code

```
1
2
3
```

Matching subsets of strings

We previously matched the entire string “High Cholesterol”
But we can use the same function to detect patterns within longer strings.

Let’s look for how many patients take a statin of any kind using
`str_count(heart_joined$medication_hx, "statin")`

What about subsets of strings?

```
str_count(heart_joined$medication_hx, "statin")
```

What does the ouput mean?

Code | Start Over | Run Code

```
1
2
3
```

A note about string patterns

When using a stringr function, you may get an output saying a string pattern doesn't exist. If you know for sure it does,
double check capitalization
.

The string must match exactly, or it will not be found!

stringr Exercise

How many people having an `"auntie" or "aunt"` in their health history?

Go to code/
Open 09_stringr.Rmd
Complete the exercise.

Using Regular Expressions

That solution worked in this case, but was not very elegant, and might not work for all cases (what if there was a ‘great aunt’ in the list?)

Or here is a more specific case for this data set.

How many patients have a father with a history of disease? But we don’t want to include grandfathers in the results.

We can use something called **Regular Expressions**, aka **Regex**, to solve this

Using Regular Expressions

Think of regex as a separate language, with it's own code, syntax, and rules.

Regex rules allow complex matching patterns for strings, to ensure matching *exactly* the content desired

It is far too complex to cover in its entirely here, but here is one specific example.

GOAL: identify all of the patients that have a father with a history of disease, but excluding grandfathers in the results.

Regular Expression Example

Example
We want to start with recognizing `father`.

But then we want to make sure that we capture both `Father` and `father`. To accept either case f in the first spot we add `(F|f)`, so now our regex looks like `(F|f)ather`

Lastly, we want this pattern to appear at the beginning of the word, so we add the regex `^` symbol.

Our completed regex looks like:

```
str_count(heart_joined$family_history, "^(F|f)ather")
```

Code | Start Over | Run Code

```
1
2
3
```

Regex Exercise

Go to code/
Open 09_stringr.Rmd
Complete the exercise to count mothers.

Regex resources

- The stringr cheatsheet shown above is a great resource
- Here the tidyverse website with a link to its vignette

Regular expressions

Source: `vignettes/regular-expressions.Rmd` (https://github.com/tidyverse/stringr/blob/master/vignettes/regular-expressions.Rmd)

Regular expressions are a concise and flexible tool for describing patterns in strings. This vignette describes the key features of stringr's regular expressions, as implemented by stringi (https://github.com/gagolews/stringi). It is not a tutorial, so if you're unfamiliar regular expressions, I'd recommend starting at http://r4ds.had.co.nz/strings.html (http://r4ds.had.co.nz/strings.html). If you want to master the details, I'd recommend reading the classic *Mastering Regular Expressions* (https://amzn.com/0596528124) by Jeffrey E. F. Friedl.

Regular expressions are the default pattern engine in stringr. That means when you use a pattern matching

Detecting strings

In addition to counting, we can use another function `str_detect()` to logically evaluate a character string.

Because this logically evaluates an expression, the output is either `TRUE or FALSE`

Practially, `str_detect` is used to detect the presence or absence of a pattern in a string

Logic Evaluation

Example
Find the patients with diabetes using the following code

```
str_detect(heart_joined$health_status, "Diabetic")
```

Code | Start Over | Run Code

```
1
2
3
```

Modifiying strings with `str_replace()`

In the health_status column we have:
-`"Diabetic"`
-`"High Cholesterol"`
-`"Normal blood sugar and cholesterol"`

But let's say we want to simplify healthy individuals to `"Normal"`

```
str_replace(heart_joined$health_status, "Normal blood sugar and cholesterol", "Normal")
```

Code | Start Over | Run Code

```
1
2
3
```

Modifiying strings with `str_replace()`

We use this same code to modify the `health_status` column by assigning it to the same variable

```
heart_joined$health_status <-
str_replace(heart_joined$health_status, "Normal blood sugar and
cholesterol", "normal")
```

```
##  [1] "normal"                      "Diabetic"
##  [3] "normal"                      "normal"
##  [5] "Diabetic & High Cholesterol" "High Cholesterol"
##  [7] "High Cholesterol"            "High Cholesterol"
##  [9] "High Cholesterol"            "High Cholesterol"
```

Using `stringr` with `dplyr`

We can use `stringr` functions in tandem with `dplyr` functions.

Example
We want to make a logical variable (`TRUE`/`FALSE`) that tells us if a patient has a normal health history using

```
heart_joined2 <- heart_joined %>% mutate(healthy =
str_detect(health_status, "normal"))
```

```
##  [1]  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

Regex Exercise

Go to code/
Open 09_stringr.Rmd
Complete the exercise to count mothers.

Any questions?

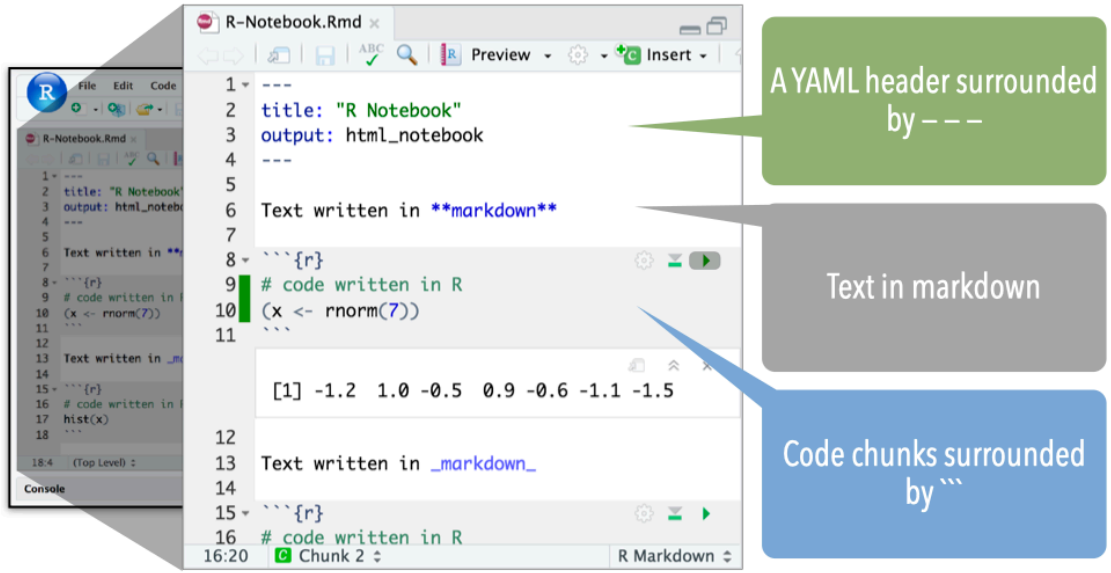
R Markdown

Scientific Reproducibly and Provenance with

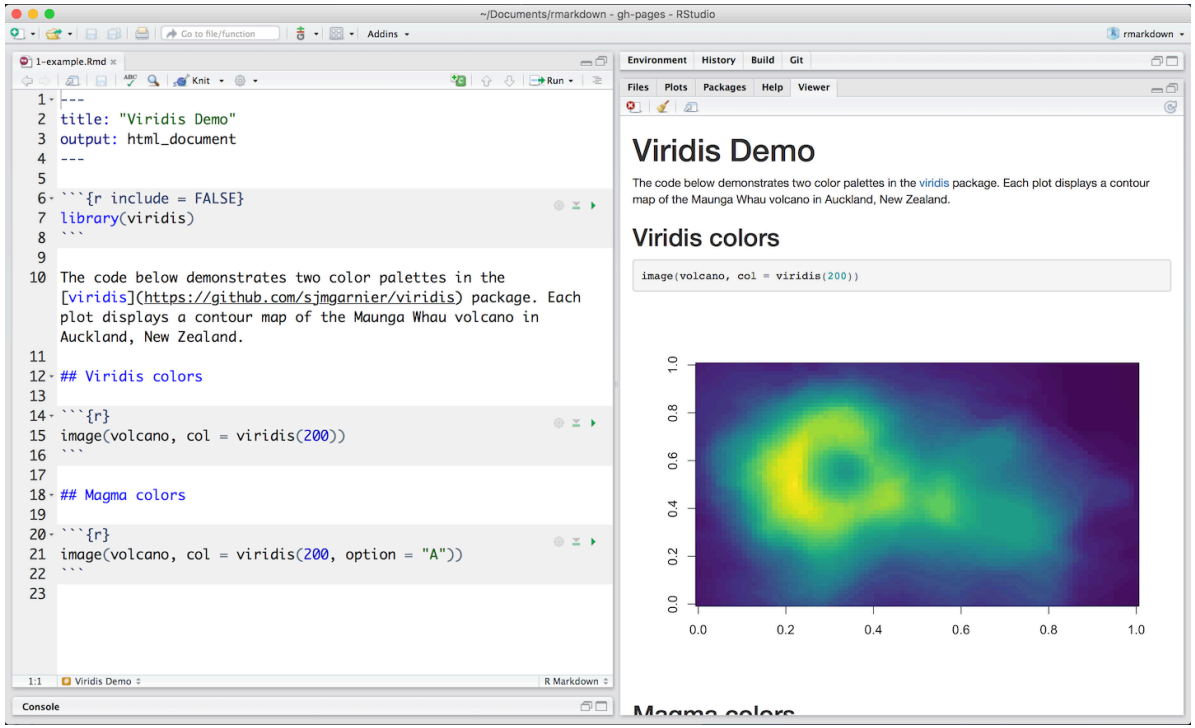
www.rstudio.com (www.rstudio.com)

R Markdown

Plain text file with 3 types of content



Use the 'knit' button to render a report



How it works

When you run render, R Markdown feeds the .Rmd file to knitr ⧉, which executes all of the code chunks and creates a new markdown (.md) document which includes the code and it's output.

The markdown file generated by `knitr` is then processed by `pandoc` which is responsible for creating the finished format.

This may sound complicated, but R Markdown makes it extremely simple by encapsulating all of the above processing into a single render function.

Key `knitr` points

- `Knitr` runs the document in a fresh R session, which means you need to load the libraries that the document uses **in the document**
- Objects made in one code chunk will be available to code in later code chunks, but not before
- For example, first create heart and then using `dplyr::left_join` you create heart_joined, heart will be available later on in the document to do this. **However, you cannot use heart_joined in a code chunk before you make it, even if it is available in your environmnet**
- To keep this straight, just think (and code) in sequential chunks

Rmarkdown – The definitive Guide

https://bookdown.org/yihui/rmarkdown/
(https://bookdown.org/yihui/rmarkdown/)

R Markdown: The Definitive Guide

Yihui Xie, J. J. Allaire, Garrett Grolemund

2019-12-02

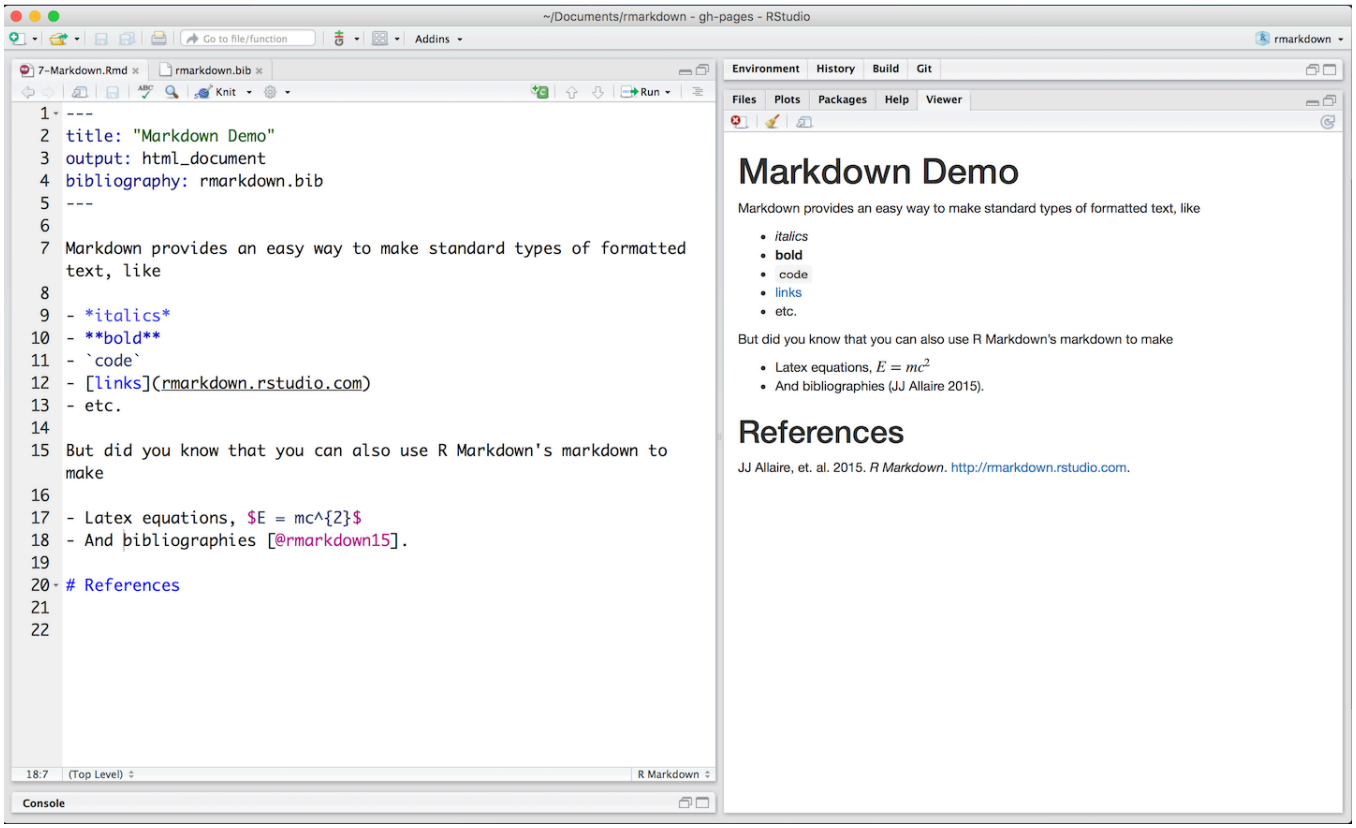
Preface

Note: This book has been published by Chapman & Hall/CRC. The online version of this book is free to read here (thanks to Chapman & Hall/CRC), and licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Any questions?

Rmarkdown Text

Markdown is a simplifed language to format text



Markdown can ‘markup’ text to be

- **bold**
- *italics*
- `code`
- unformatted text
- # Medium headers (3)
- Small headers (6)
- Bullets (like this list)
- Links, such as to the github repository storing class material (https://github.com/matthewhirschey/)
- …and many more

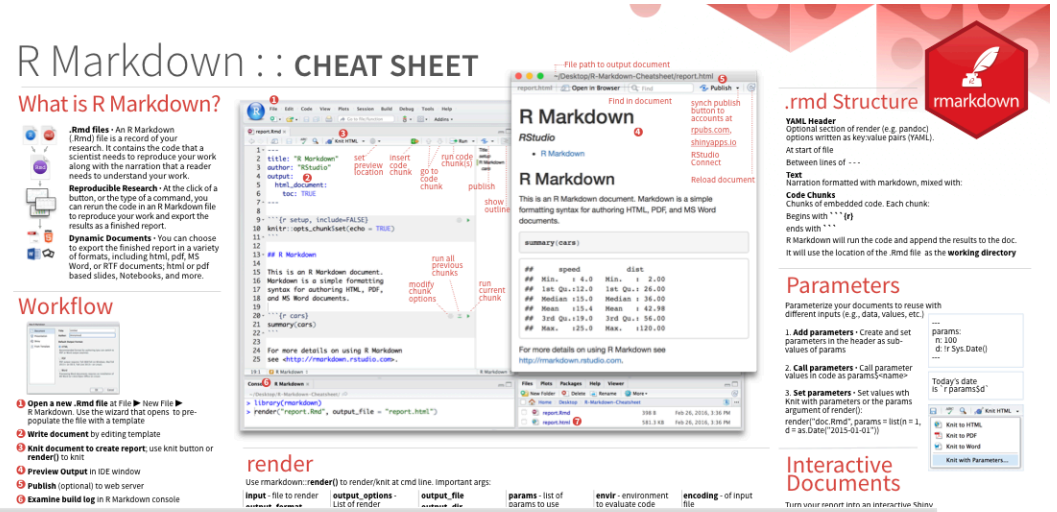
Rmarkdown cheatsheet & quick reference

`R Markdown (index.html)`

from (https://www.rstudio.com/) Get Started (lesson-1.html) Gallery (gallery.html) Formats (formats.html) Articles (articles.html)

Cheatsheets

Use the cheatsheets that are built into the RStudio IDE to jog your memory about R Markdown.



Any questions?

Rmarkdown Code

Rmarkdown Code

Insert a chunk of R code with

````
```{r}
# some code
```
````

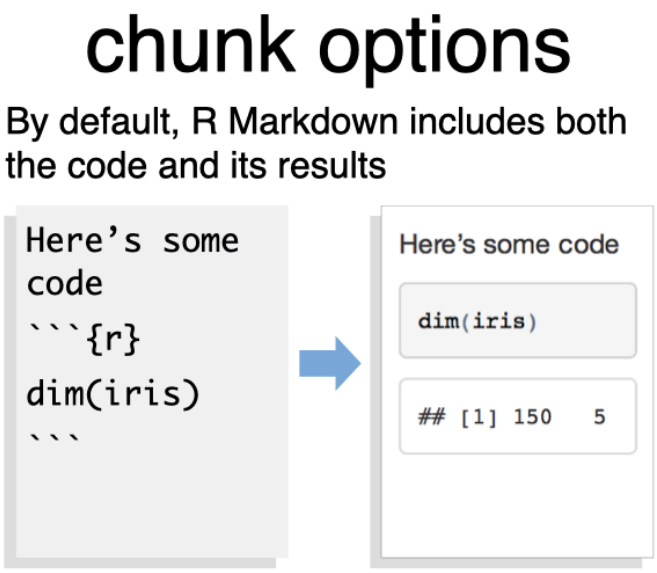
When you render the report, R Markdown will run the code and include its results. R Markdown will also remove the ```` ```{r} ```` and ```` ``` ````.

Rmarkdown Code Shortcut

Insert a chunk of R code with

```
```{r}
# some code
```
```

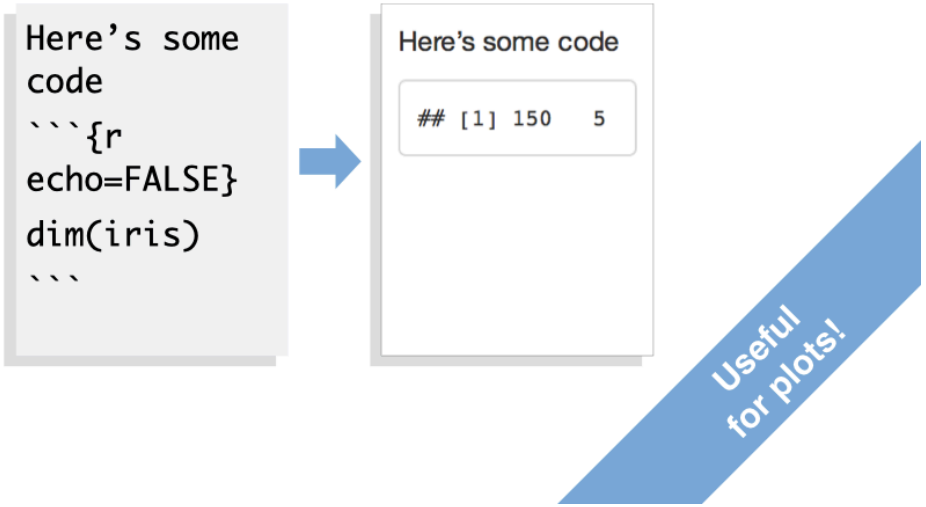
Rmarkdown Code Chunk Options



Rmarkdown Code Chunk Options

echo

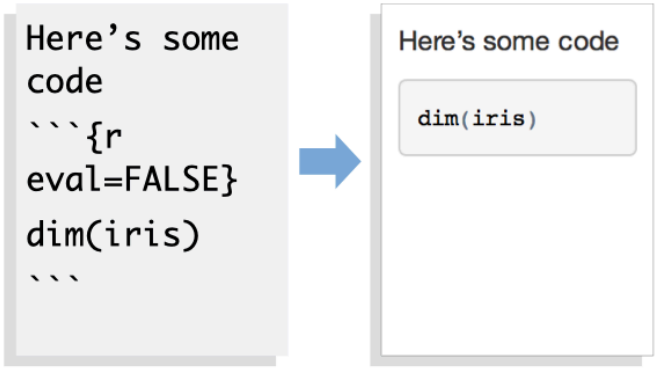
Add options in the brackets after r.
echo = FALSE hides the code.



Rmarkdown Code Chunk Options

eval

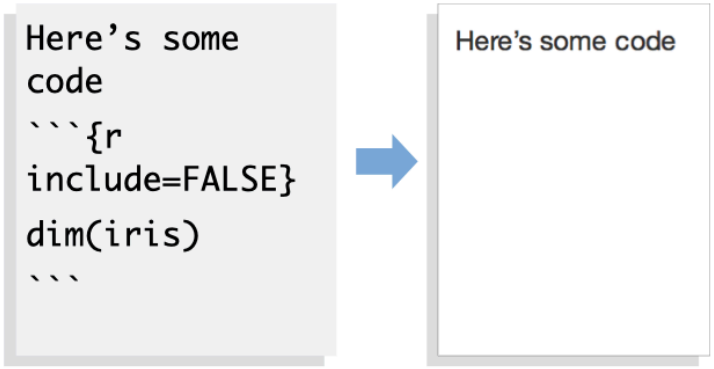
eval = FALSE prevents the code from being run. As a result, no results will be displayed with the code.



Rmarkdown Code Chunk Options

include

include = FALSE runs the code, but prevents both the code and the results from appearing (e.g. to setup).

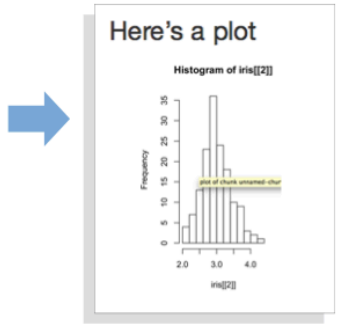


Rmarkdown Code Chunk Options

fig.height, fig.width

Specify the dimension of plots (in inches) with fig.width and fig.height. Separate multiple arguments with commas.

````
Here's a plot
```{r echo=FALSE, fig.width=3, fig.height=5}
hist(iris[[2]])
```
````



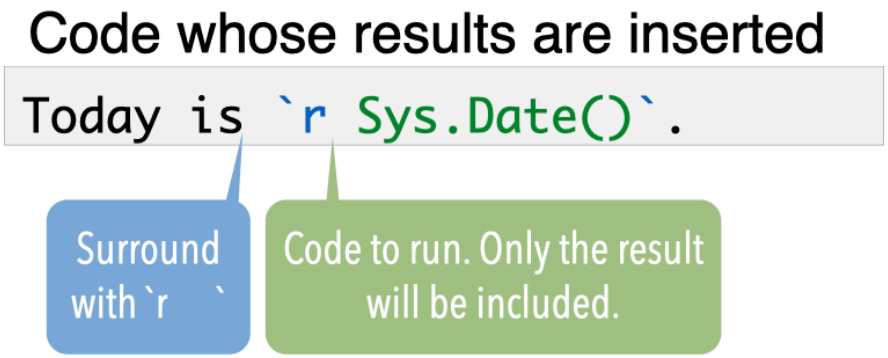
Rmarkdown Code Chunk Options

Place code in a sentence with `` `r <code>` ``. R Markdown will replace the code with its results.

```
Today is
`r
Sys.Date()`
.
```

Today is 2015-04-16.

Rmarkdown Code Chunk Options



Clean-up the Rmarkdown exercise

1. Go to code/
2. Open 10_rmarkdown.Rmd
3. Inspect the code starting with the set-up chunk. Below it, you'll see code that imports data, stores objects, and creates a graph.
4. Knit the document and see the results; pretty messy, huh?
5. Use what you've learn to execute the code, but not include the results of the set-up chunk and data import chunk.
6. Replace every **BOLD** with inline R code
7. Remove the {calculations} chunk so it is not included with the output
8. Change the {graph} chunk so that only the output of the plot is shown, but not the code.
9. Re-Knit the document
10. **Beautiful!**

Any questions?

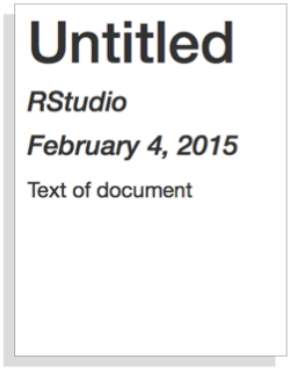
Rmarkdown YAML

Rmarkdown header is called 'YAML'

YAML

A section of `key:value` pairs
separated by dashed lines ———

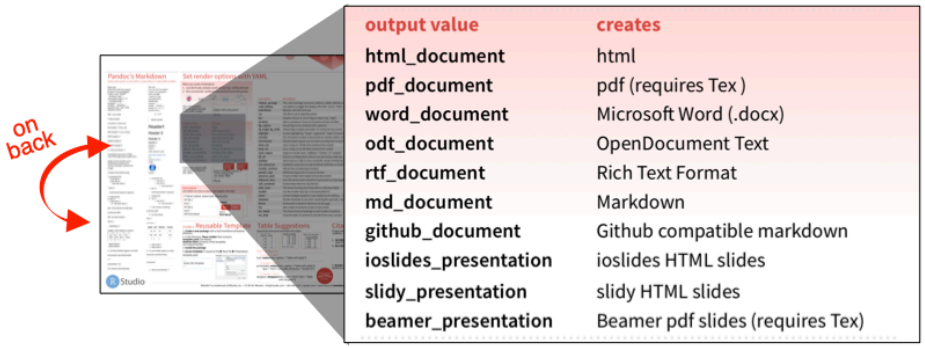
```
---
title: "Untitled"
author: "RStudio"
date: "February 4, 2015"
output: html_document
---
Text of document
```



Rmarkdown output is defined here

output

The output: field sets the format of

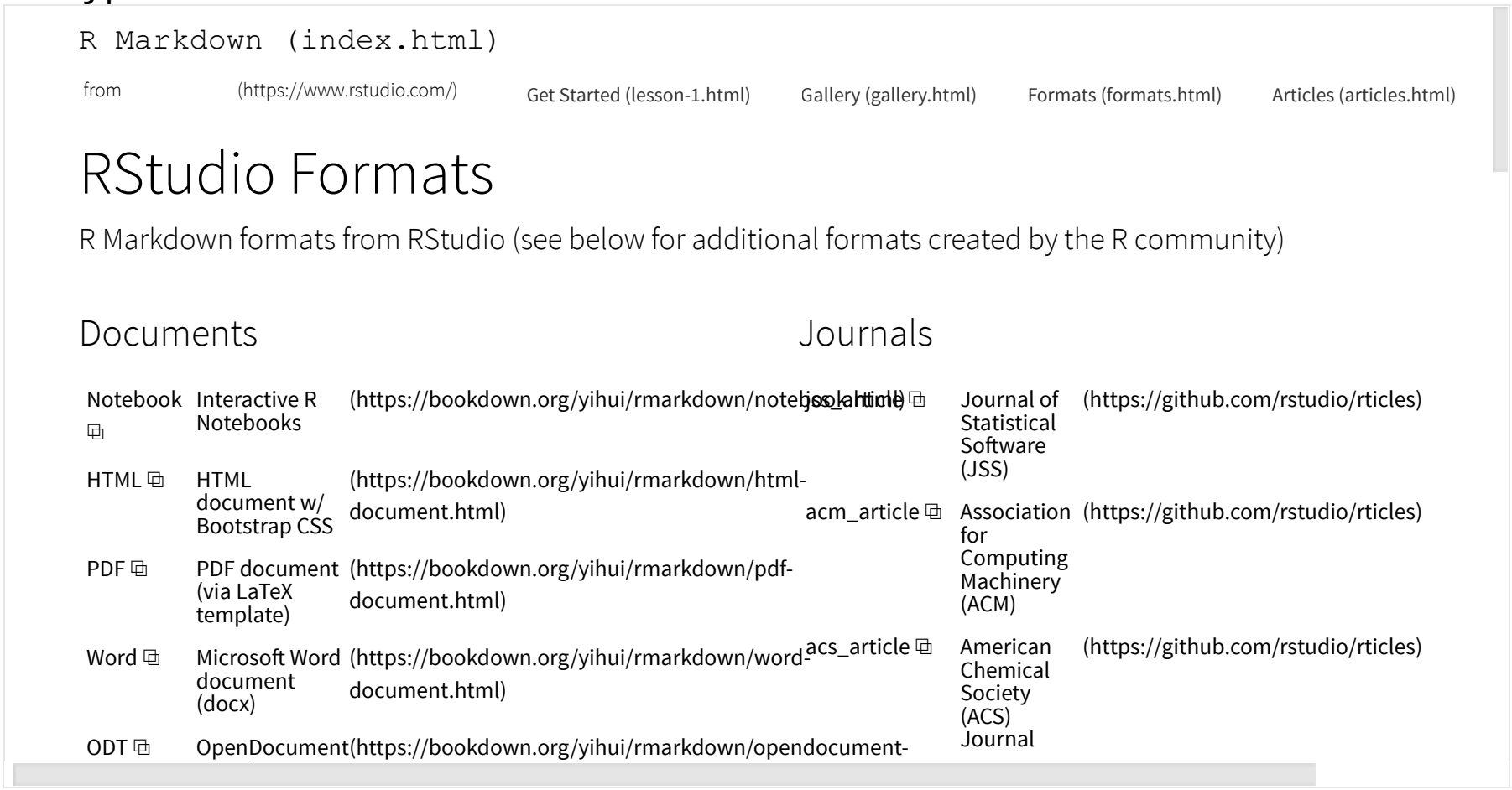


output value	creates
html_document	html
pdf_document	pdf (requires Tex)
word_document	Microsoft Word (.docx)
odt_document	OpenDocument Text
rtf_document	Rich Text Format
md_document	Markdown
github_document	Github compatible markdown
ioslides_presentation	ioslides HTML slides
slidy_presentation	slidy HTML slides
beamer_presentation	Beamer pdf slides (requires Tex)

More at rmarkdown.rstudio.com/formats.html

Rmarkdown formats

Recall that Rmarkdown documents can be rendered into several different output file types



R Markdown (index.html)

from (https://www.rstudio.com/) Get Started (lesson-1.html) Gallery (gallery.html) Formats (formats.html) Articles (articles.html)

RStudio Formats

R Markdown formats from RStudio (see below for additional formats created by the R community)

Documents

Notebook ⧉	Interactive R Notebooks	(https://bookdown.org/yihui/rmarkdown/notebook.html)
HTML ⧉	HTML document w/ Bootstrap CSS	(https://bookdown.org/yihui/rmarkdown/html-document.html)
PDF ⧉	PDF document (via LaTeX template)	(https://bookdown.org/yihui/rmarkdown/pdf-document.html)
Word ⧉	Microsoft Word document (docx)	(https://bookdown.org/yihui/rmarkdown/word-document.html)
ODT ⧉	OpenDocument	(https://bookdown.org/yihui/rmarkdown/opendocument-

Journals

jss_article ⧉	Journal of Statistical Software (JSS)	(https://github.com/rstudio/rticles)
acm_article ⧉	Association for Computing Machinery (ACM)	(https://github.com/rstudio/rticles)
acs_article ⧉	American Chemical Society (ACS) Journal	(https://github.com/rstudio/rticles)

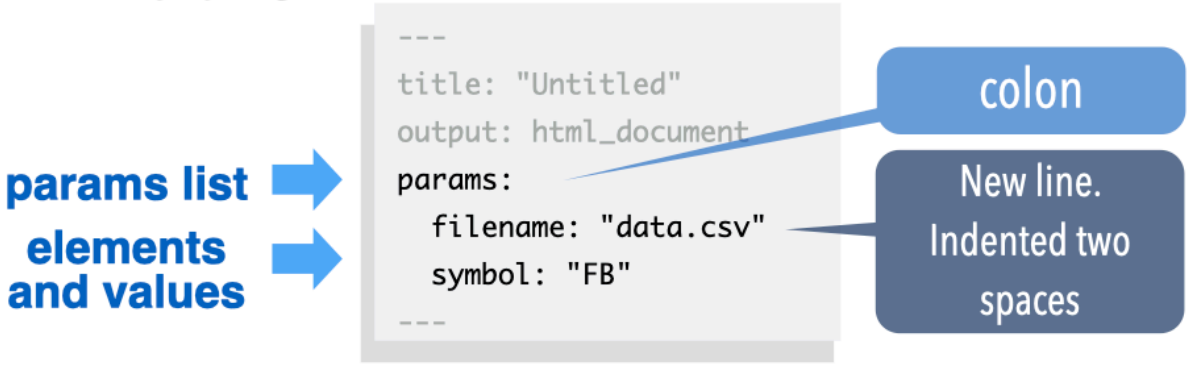
Rmarkdown Parameters

Parameters of a document are defined in the YAML header, and can pre-populate an Rmarkdown document. To see this in action,

1. Open 11_rmarkdown_params.Rmd.
2. Click the dropdown menu next to Knit and use Knit with Parameters to render the document.
3. What happens if you type in a different gender or different age?

Parameters

A list of values that you can call in R code chunks



Using Parameters

Call parameter values as elements of the params list, **params$num**

````
---
params:
  num: 42
---
The value of the
parameter is
`r params$num`, e.g.

```{r}
params$num
```
````

The value of the parameter is 42, e.g.

```
params$num
```

```
## [1] 42
```

Take another look

Re-inspect 11_rmarkdown_params.Rmd.

Where were the parameters called in the code?

Concluding Thoughts

Data science enables

- Data science enables data-driven information gathering and hypothesis generation
  – Scientific Research
  – Reviews
- Data science enables the ability to ask new types of questions
- Process-centric, not necessarily question-centric
- Making things computable makes them actionable at zero marginal cost.
- Workflows save time, achieve reproducibility

Resources

Cheatsheets (https://rstudio.com/resources/cheatsheets)

RStudio Cheatsheets

The cheatsheets below make it easy to use some of our favorite packages. From time to time, we will add new cheatsheets. If you'd like us to drop you an email when we do, click the button below.

SUBSCRIBE TO CHEATSHEET UPDATES

CONTRIBUTED CHEATSHEETS

Resources

Stackoverflow (https://stackoverflow.com/questions/tagged/r)

Resources

R for Data Science (https://r4ds.had.co.nz)

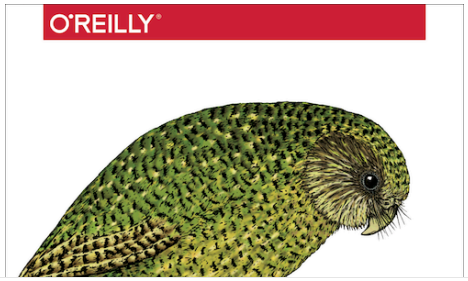
R for Data Science

Garrett Grolemund

Hadley Wickham

Welcome

This is the website for **“R for Data Science”**. This book will teach you how to do data science with R: You’ll learn how to get your data into R, get it into the most useful structure, transform it, visualise it and model it. In this book, you will find a practicum of skills for data science. Just as a chemist learns how to clean test tubes and stock a lab, you’ll learn how to clean data and draw



Resources

Tidytuesday (https://github.com/rfordatascience/tidytuesday)

Acknowledgements

Teaching Assistants
- Allie Mills, Ph.D.
- Akshay Bareja, D.Phil.

Inspiration, ideas, packages, code
- R4DS (Garrett Grolemund and Hadley Wickham)
- Mine Çetinkaya-Rundel (datasciencebox.org)
- Chester Ismay and Albert Y. Kim (Modern Dive)
- Garrett Grolemund (Remastering the Tidyverse)
- Tidyverse devs and community
- Rstudio

Any questions?

Thank you