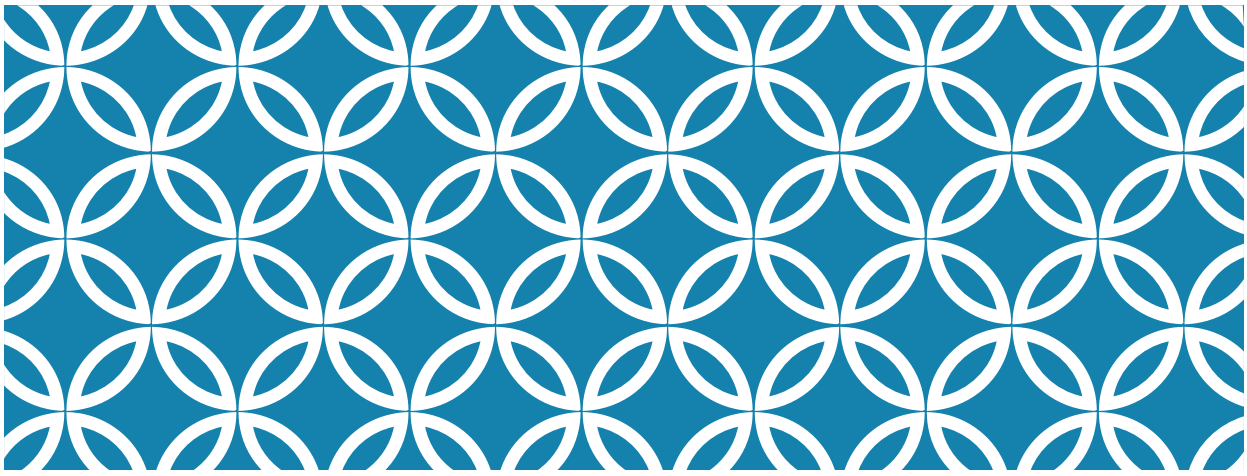
CHOP R 101

INTRO TO R FOR CLINICAL DATA

Arcus Education

Children's Hospital of Philadelphia®



CHOP R 101:
Intro to R for Clinical Data

Introduction

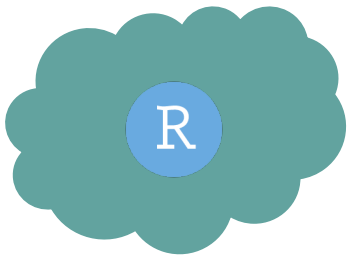
R
Programming language for data analysis

RStudio
Integrated development environment (IDE)

R Markdown
Computational document format

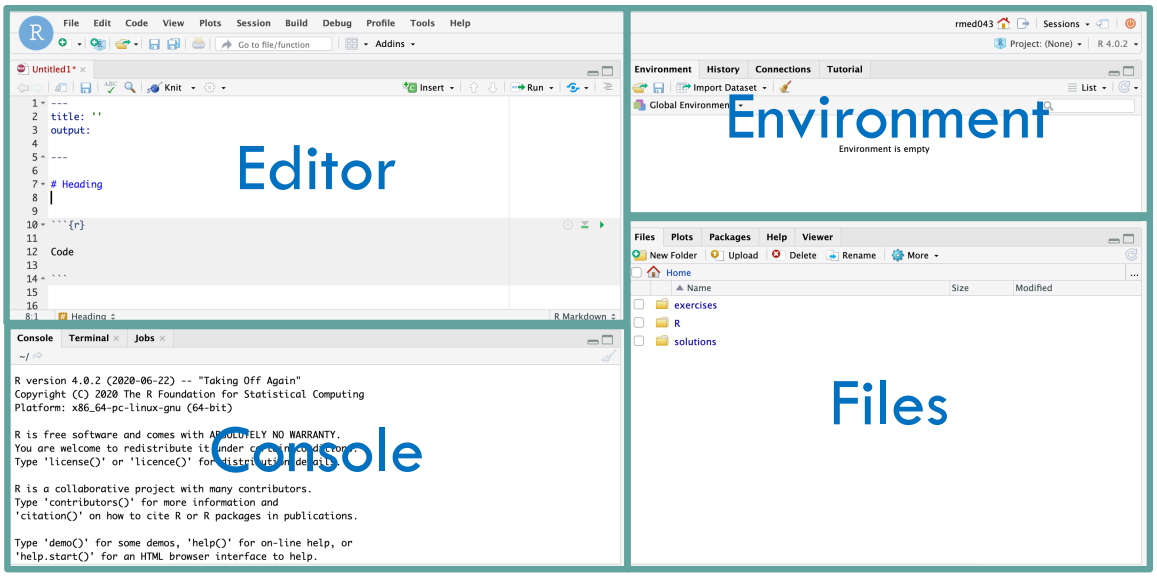
Getting Started with RStudio

RStudio: On the Web and In Your Home



RStudio Server
Hosted on a server
(in the cloud)

RStudio Desktop
Installed on your
computer



Your Turn #1

Click the link in the chat to access the RStudio training environment.

Log in using your username and password.

Click “yes” once you see the RStudio panes.

Reproducible Data Analysis and R Markdown

The Duke Cancer Scandal

- Chemo sensitivity from microarrays
- Serious errors in data analysis
- Clinical trials based on flawed models
- Papers retracted, lawsuits settled

Duke	MD Anderson
"1881_at"	"1882_g_at"
"31321_at"	"31322_at"
"31725_s_at"	"31726_at"
"32307_r_at"	"32308_r_at"
...	

Off-by-one indexing error

"Common problems are simple…

Off-by-one **indexing error**

Sensitive / resistant **label reversal**

Confounding in experimental design

Inclusion of data from **non-reported sources**

Wrong figure shown

… and simple problems are common."

Point-and-click is not reproducible

Why YOU should analyze your data reproducibly

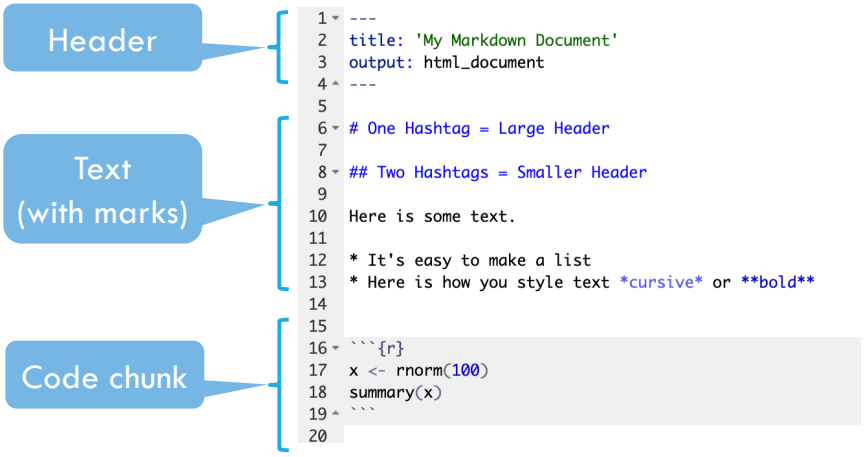
"Can we redo the analysis with this month's data?"

"Why do the data in Table 1 not seem to agree with Figure 2?"

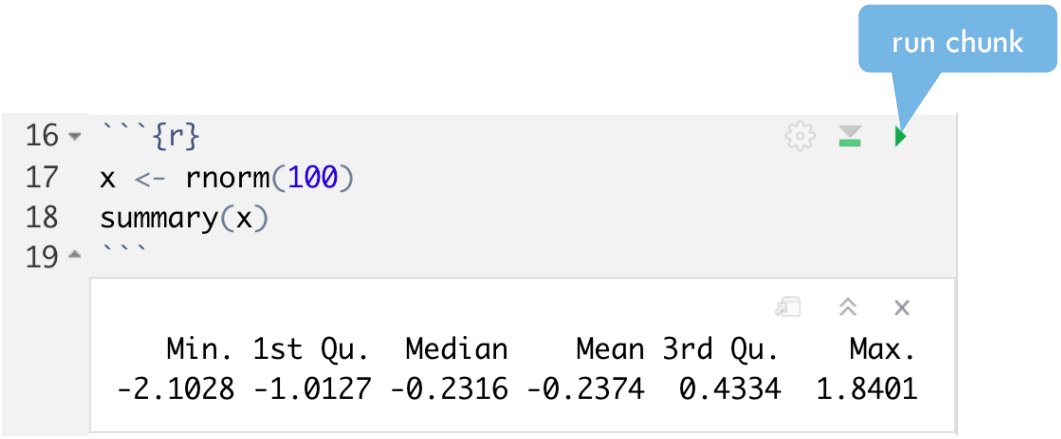
"Why did I decide to omit these six samples from my analysis?"

YOUR CLOSEST COLLABORATOR IS **YOU** FROM 6 MONTHS AGO!

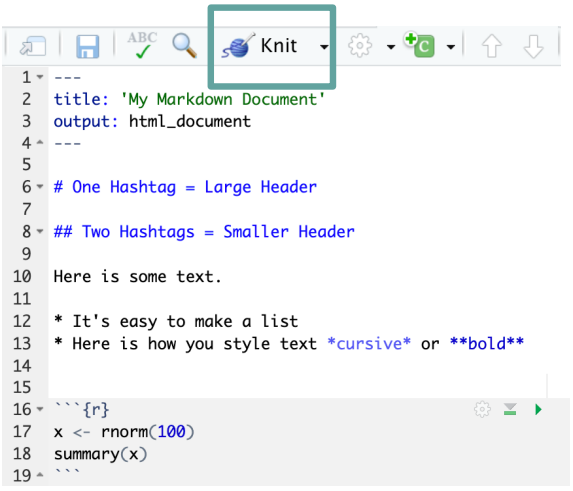
Anatomy of an R Markdown document

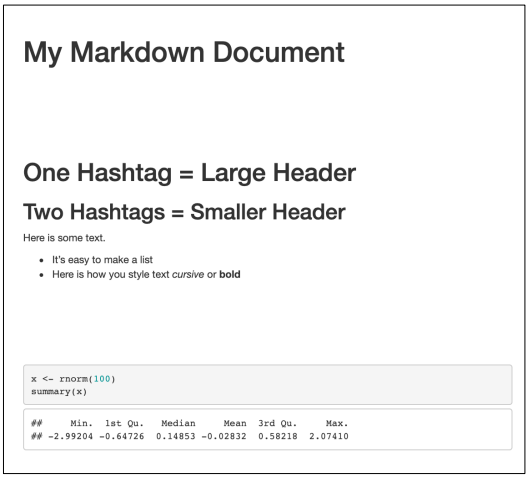


Running a single code chunk



Rendering ("Knitting") an R Markdown document





```
---
title: 'My Markdown Document'
output: html_document
---


# One Hashtag = Large Header

## Two Hashtags = Smaller Header

Here is some text.

* It's easy to make a list
* Here is how you style text *cursive* or **bold**


```{r}
x <- rnorm(100)
summary(x)
```


Including Plots

```{r, echo=FALSE}
hist(x)
```
```

My Markdown Document

One Hashtag = Large Header

Two Hashtags = Smaller Header

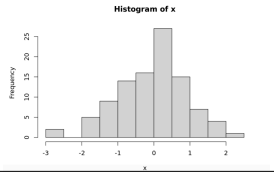
Here is some text.

- It's easy to make a list
- Here is how you style text *cursive* or **bold**

```
x <- rnorm(100)
summary(x)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.99204 -0.64726  0.14853 -0.02832  0.58218  2.07410
```

Including Plots



Your Turn #2

Go to **File** > **New File** > **R Markdown.** Click **OK.**

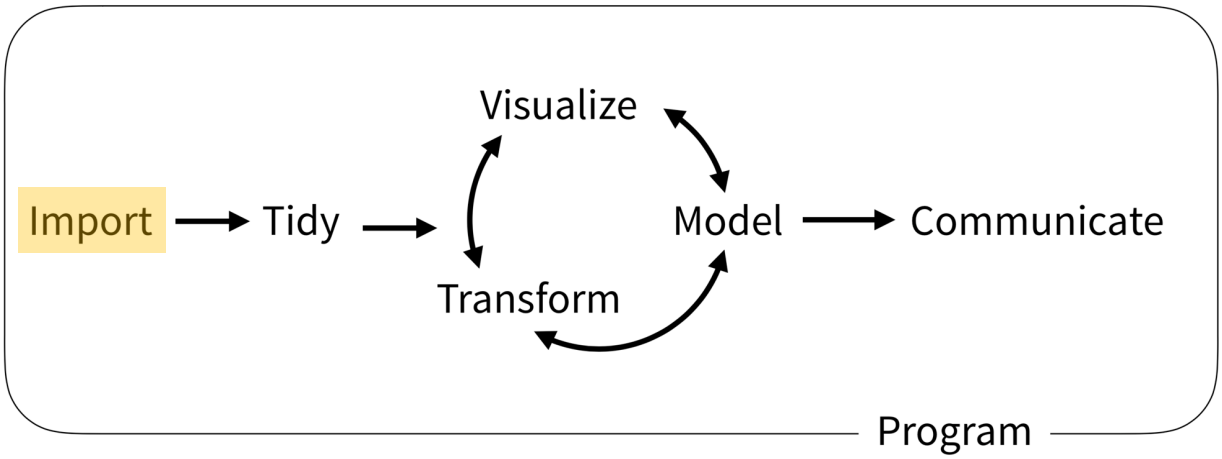
Run each chunk by clicking ▸. Note what happens.

Knit the document (Knit). Type **test** and click **Save** to save the HTML file. Inspect the HTML document.

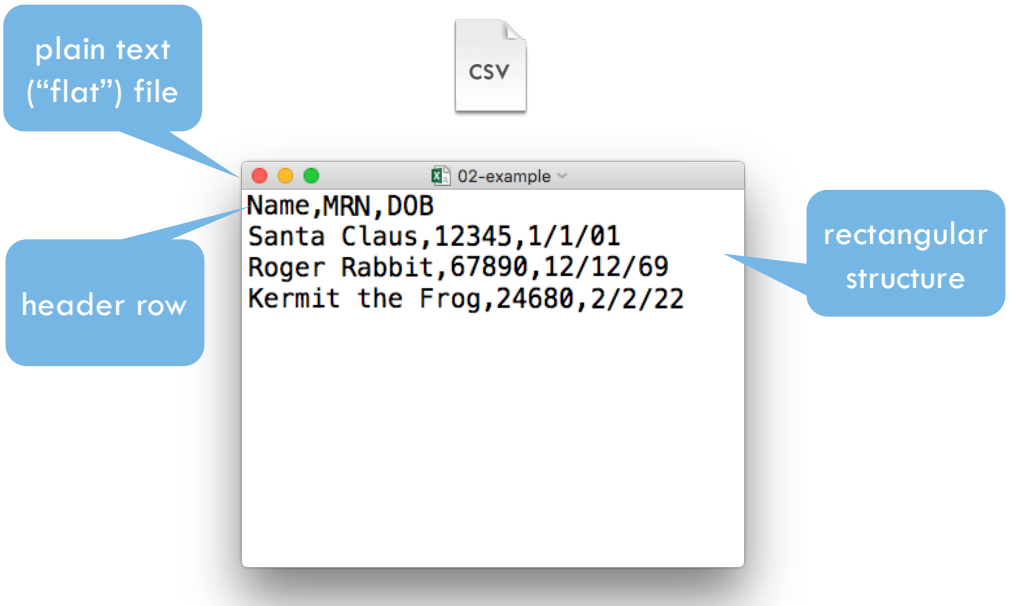
03:00

Importing Data

The Data Analysis Pipeline



From *R for Data Science* (https://r4ds.had.co.nz/introduction.html)

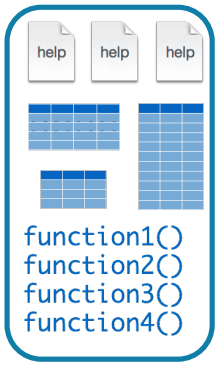


Tidyverse: R Packages for Data Science

- A consistent way to organize data
- Human readable, concise, consistent code
- Build pipelines from atomic data analysis steps

Installing and loading R packages

tidyverse




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

Downloads files to computer

1 x per computer

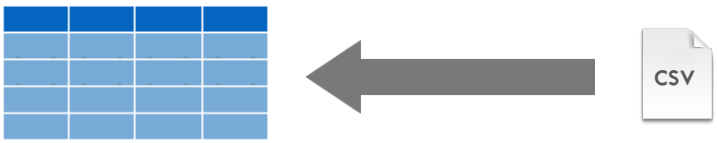
```
library("tidyverse")
```

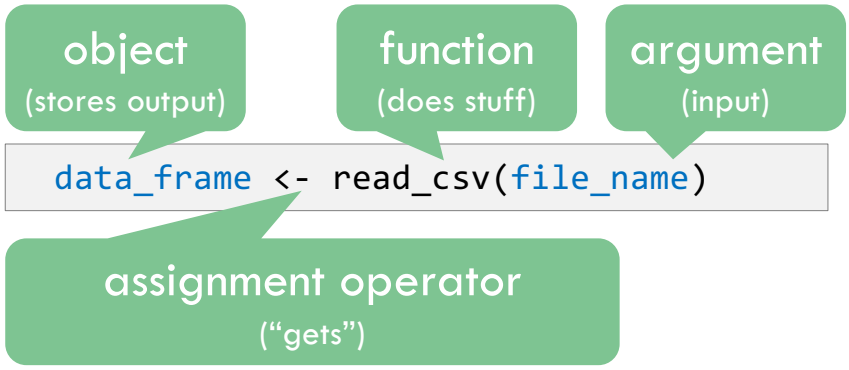
Loads package

1 x per R Session

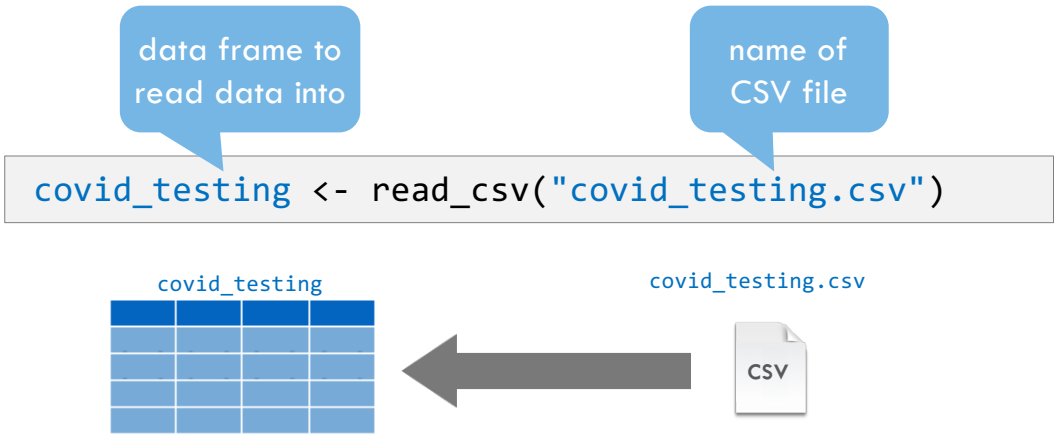
read_csv()

```
data_frame <- read_csv(file_name)
```




object
(stores output)
function
(does stuff)
argument
(input)
data_frame <- read_csv(file_name)
assignment operator
("gets")

read_csv()


data frame to read data into
name of CSV file
covid_testing <- read_csv("covid_testing.csv")
covid_testing
covid_testing.csv
CSV

Your Turn #3

In the **Files** pane, click on the folder **exercises**. Open the R Markdown file titled

01 - Introduction.Rmd

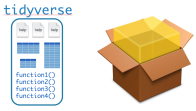
Instructions for this exercise are in the text of the R Markdown document.

Click yes when you are done.

05:00

Recap

Programming Language | **IDE (Editor)** | **Document Format**



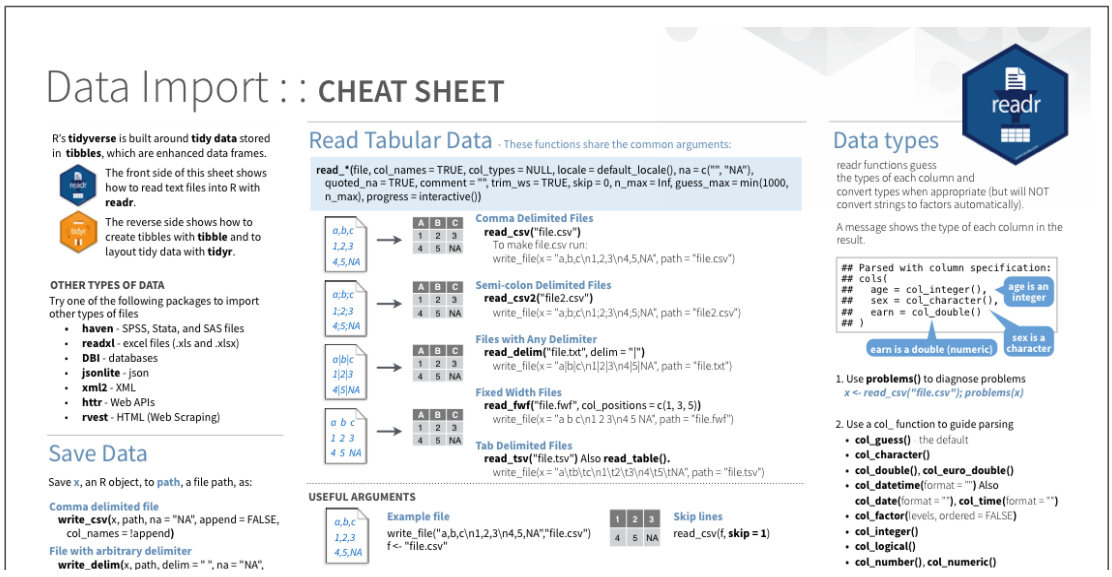
Packages extend the functionality of R. Install with `install.packages()` and load with `library()`.

Functions do stuff. They accept **Arguments** as input and return an **Output**. Capture an output in an **Object** using the **assignment operator** (`<-`).

Importing Data is the first step of data analysis. Use `read_csv()` from the `tidyverse` package to import data stored in a **CSV file.**

What else?

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



Data Import : : CHEAT SHEET

R's **tidyverse** is built around **tidy data** stored in **tibbles**, which are enhanced data frames.

The front side of this sheet shows how to read text files into R with **readr**.

The reverse side shows how to create tibbles with **tibble** and to layout tidy data with **tidyr**.

OTHER TYPES OF DATA

Try one of the following packages to import other types of files

- **haven** - SPSS, Stata, and SAS files
- **readxl** - excel files (.xls and .xlsx)
- **DBI** - databases
- **jsonlite** - json
- **xml2** - XML
- **httr** - Web APIs
- **rvest** - HTML (Web Scraping)

Save Data

Save x, an R object, to path, a file path, as:

Comma delimited file
write_csv(x, path, na = "NA", append = FALSE, col_names = !append)

File with arbitrary delimiter
write_delim(x, path, delim = " ", na = "NA",

Read Tabular Data - These functions share the common arguments:

read_*(file, col_names = TRUE, col_types = NULL, locale = default_locale(), na = c("", "NA"), quoted_na = TRUE, comment = "", trim_ws = TRUE, skip = 0, n_max = Inf, guess_max = min(1000, n_max), progress = interactive())

Comma Delimited Files
read_csv("file.csv")
To make file.csv run:
write_file(x = "a,b,c\n1,2,3\n4,5,NA", path = "file.csv")

Semi-colon Delimited Files
read_csv2("file2.csv")
write_file(x = "a;b;c\n1;2;3\n4;5;NA", path = "file2.csv")

Files with Any Delimiter
read_delim("file.txt", delim = "|")
write_file(x = "a|b|c\n1|2|3\n4|5|NA", path = "file.txt")

Fixed Width Files
read_fwf("file.fwf", col_positions = c(1, 3, 5))
write_file(x = "a b c\n1 2 3\n4 5 NA", path = "file.fwf")

Tab Delimited Files
read_tsv("file.tsv") Also **read_table()**.
write_file(x = "a\tb\tc\n1\t2\t3\n4\t5\tNA", path = "file.tsv")

USEFUL ARGUMENTS

Example file
write_file("a,b,c\n1,2,3\n4,5,NA","file.csv")
f <- "file.csv"

Skip lines
read_csv(f, **skip = 1**)

Data types

readr functions guess the types of each column and convert types when appropriate (but will NOT convert strings to factors automatically).

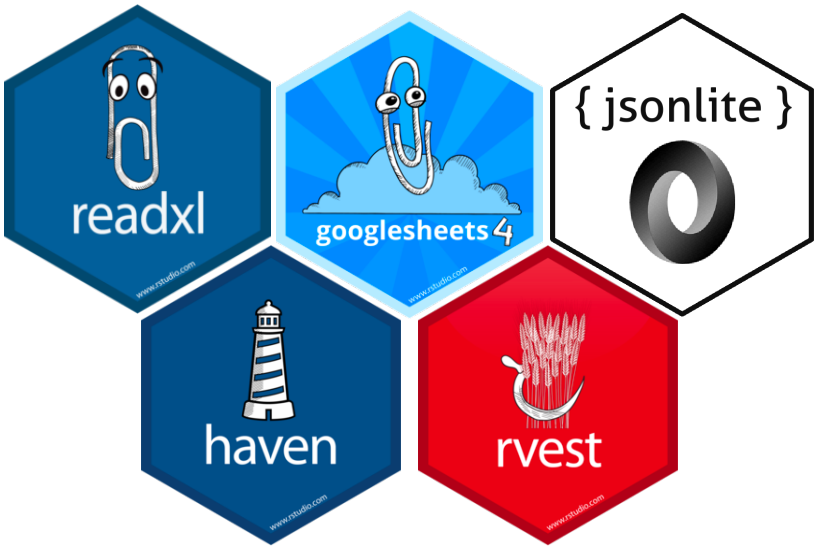
A message shows the type of each column in the result.

```
## Parsed with column specification:
## cols(
##   age = col_integer(),
##   sex = col_character(),
##   earn = col_double()
## )
```

age is an integer
earn is a double (numeric)
sex is a character

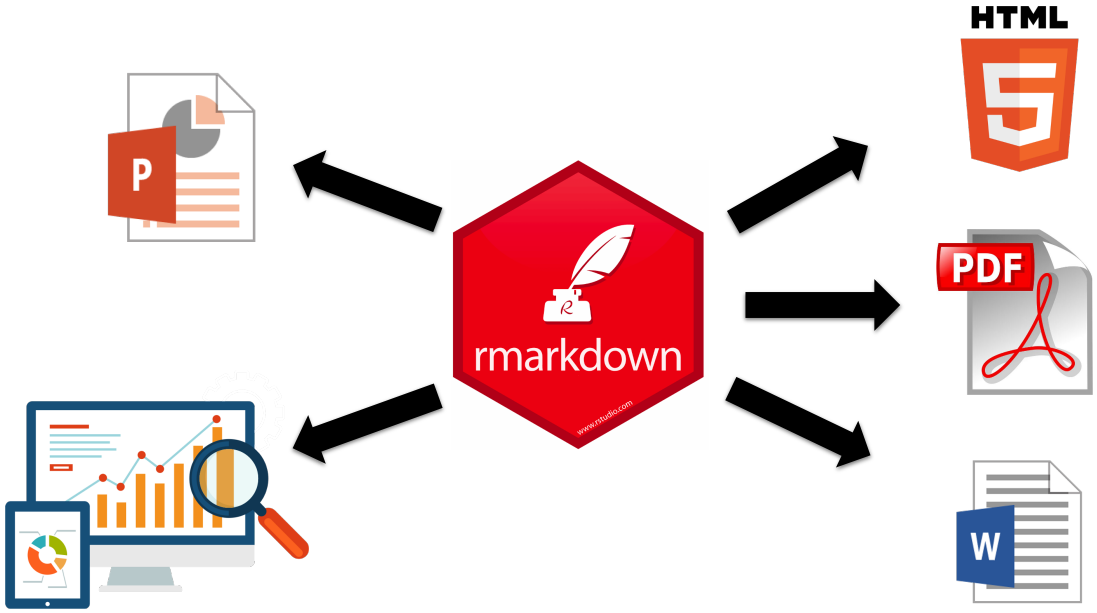
1. Use **problems()** to diagnose problems
x <- read_csv("file.csv"); problems(x)

2. Use a col_ function to guide parsing
- **col_guess()** - the default
- **col_character()**
- **col_double()**, **col_euro_double()**
- **col_datetime**(format = "") Also **col_date**(format = ""), **col_time**(format = "")
- **col_factor**(levels, ordered = FALSE)
- **col_integer()**
- **col_logical()**
- **col_number()**, **col_numeric()**



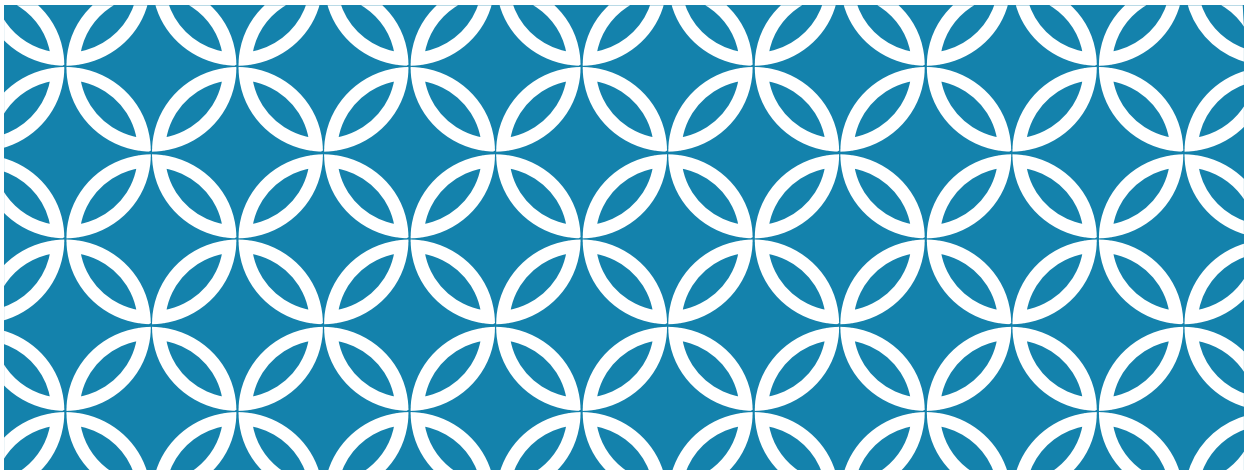
Databases

https://db.rstudio.com/databases/



R Interface to Python

```
```{python}
import pandas
covid_testing.info()
```
```



CHOP R 101
Intro to R for Clinical Data | **Visualize**

covid_testing

covid_testing

Filter

	mrn	first_name	last_name	gender	pan_day	test_id	clinic_name	result
1	5001412	jhezane	westerling	female	4	covid	inpatient ward a	negative
2	5000533	penny	targaryen	female	7	covid	clinical lab	negative
3	5009134	grunt	rivers	male	7	covid	clinical lab	negative
4	5008518	melisandre	swyft	female	8	covid	clinical lab	negative
5	5008967	rolley	karstark	male	8	covid	emergency dept	negative
6	5011048	megga	karstark	female	8	covid	oncology day hosp	negative
7	5000663	ithoke	targaryen	male	9	covid	clinical lab	negative
8	5002158	ravella	frey	female	9	covid	emergency dept	negative
9	5003794	styr	tyrell	male	9	covid	clinical lab	negative
10	5004706	wynafryd	seaworth	male	9	covid	clinical lab	negative
11	5008115	patrek	frey	male	9	covid	clinical lab	negative
12	5009309	maege	sand	female	9	covid	medical center	negative
13	5008943	myria	rivers	female	9	covid	picu	negative

Showing 1 to 14 of 15,524 entries, 17 total columns

Your Turn #1

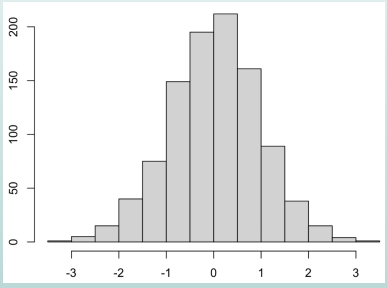
Consider the `covid_testing` data frame.

What do you think plot would look like in which:

- the x-axis represents `pan_day` (day of the pandemic), and
- the y-axis represents the number of tests that were performed on that day?

01:00

Your Turn #2



What is the name of this kind of plot?

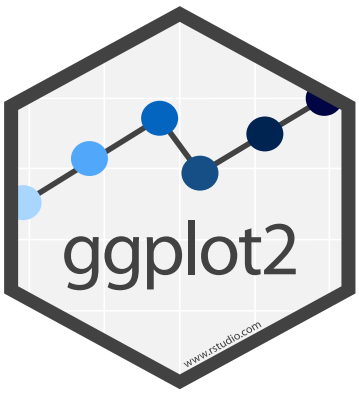
Type the answer into the chat!

Your Turn #3

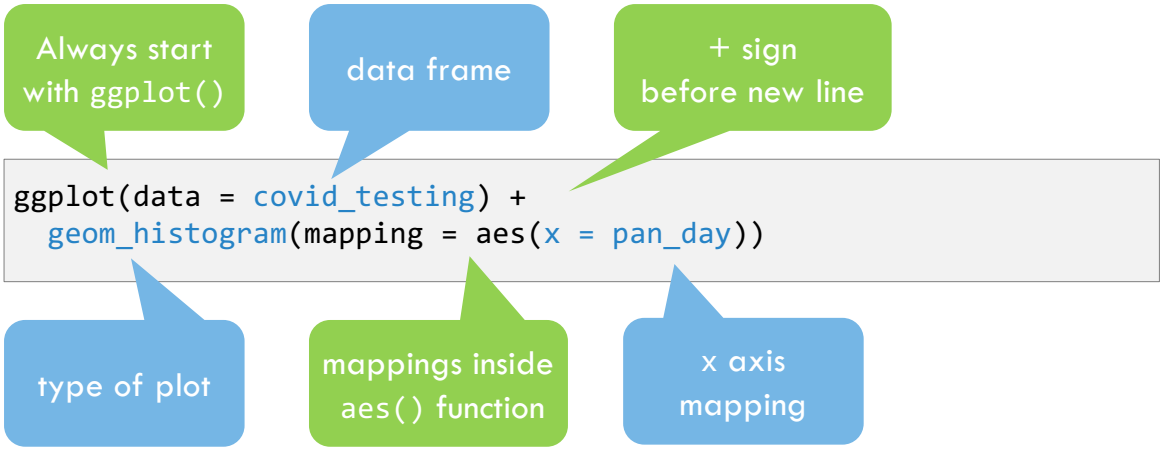
Type the following code in the RStudio console to make a graph.

Pay attention to the spelling, capitalization, and parentheses!

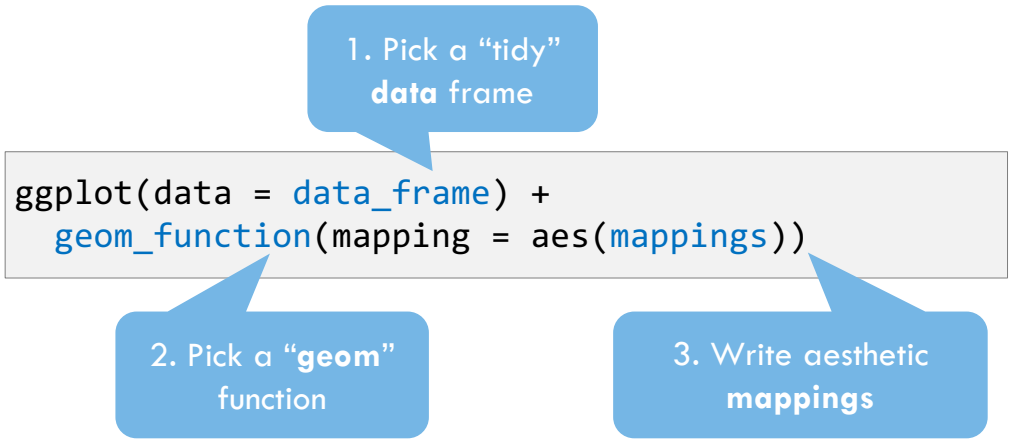
```
ggplot(data = covid_testing) +
  geom_histogram(mapping = aes(x = pan_day))
```



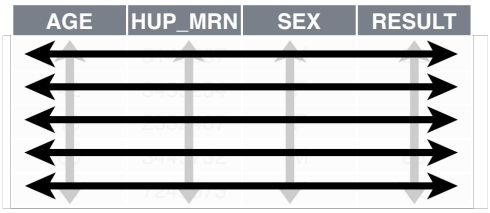
ggplot()



To make **any** kind of graph:



1. Pick a "Tidy" Data Frame



A data set is **tidy** if:

1. Each **variable** is in its own **column**
2. Each **observation** is in its own **row**
3. Each **value** is in its own **cell**

Wickham H. **Tidy Data.** *J Stat Soft* 2014.

2. Pick a "Geom" Function


`geom_histogram()`


`geom_dotplot()`


`geom_bar()`

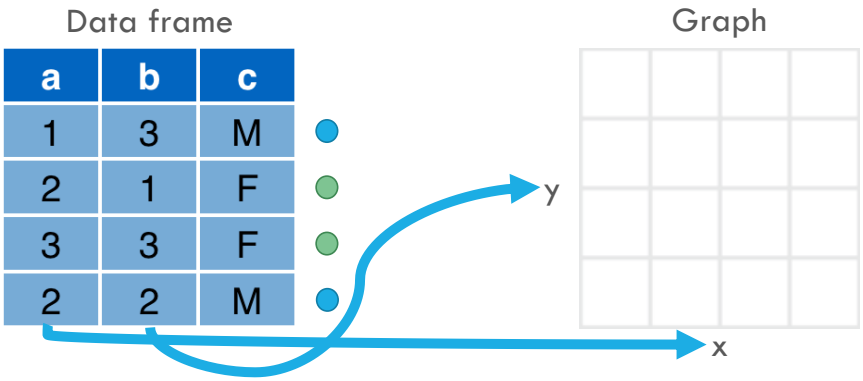

`geom_boxplot()`


`geom_point()`

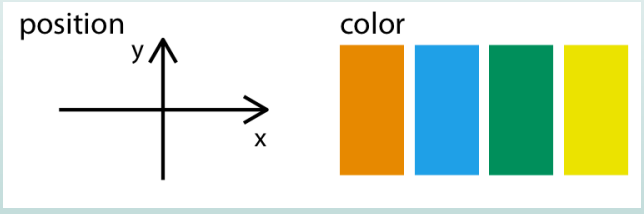

`geom_line()`

3. Write Aesthetic Mappings

```
aes(x = a, y = b, color = c)
```



Your Turn #4



In addition to x/y position and color, what other aesthetic mappings can you think of?

Type your answers in the chat!

Your Turn #5

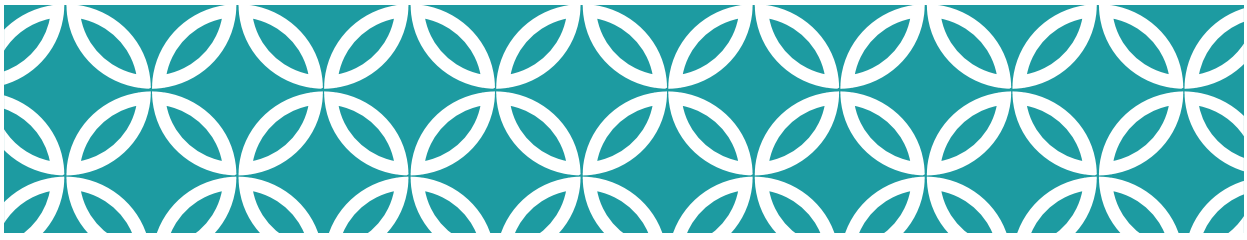
Open `02 - Visualize.Rmd`. Work through the exercises of the section titled "Your Turn 5".

Stop when it says "Stop Here".

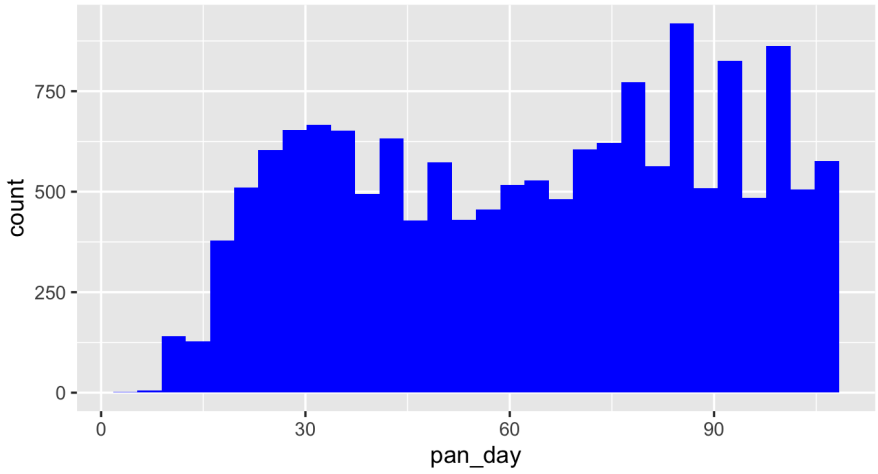
Click "yes" when you're done!

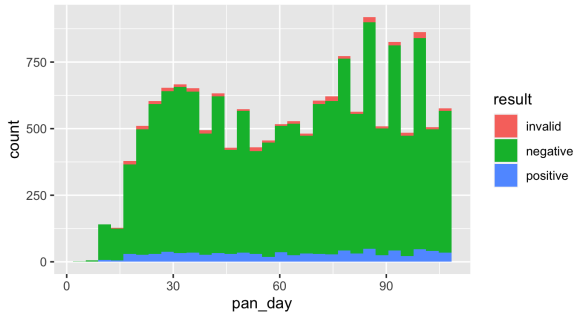
05:00

Setting vs Mapping Aesthetics



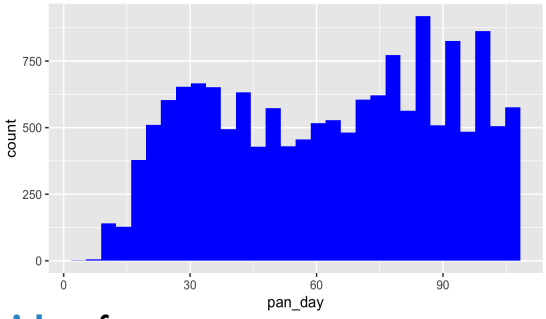
How would you make this plot?





Inside of `aes()`:
map an aesthetic
to a **variable**

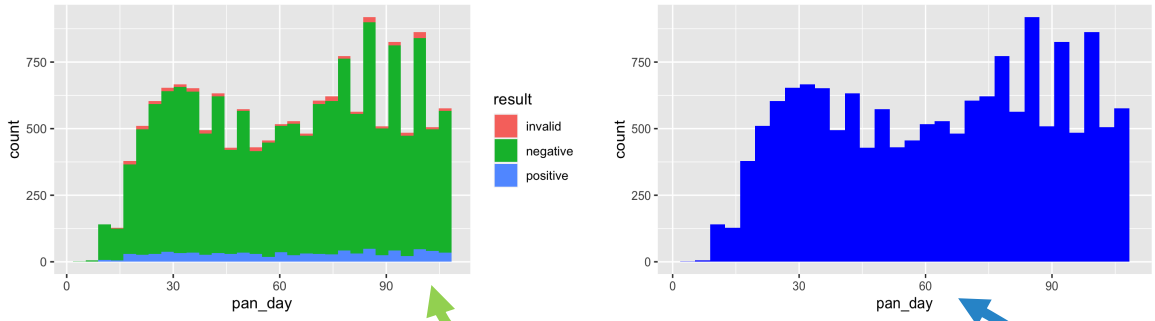
```
ggplot(data = covid_testing) +
  geom_histogram(mapping = aes(x = pan_day, fill = result))
```



Outside of `aes()`:
set an aesthetic to a **value**

color name in "quotes"

```
ggplot(data = covid_testing) +
  geom_histogram(mapping = aes(x = pan_day), fill = "blue")
```

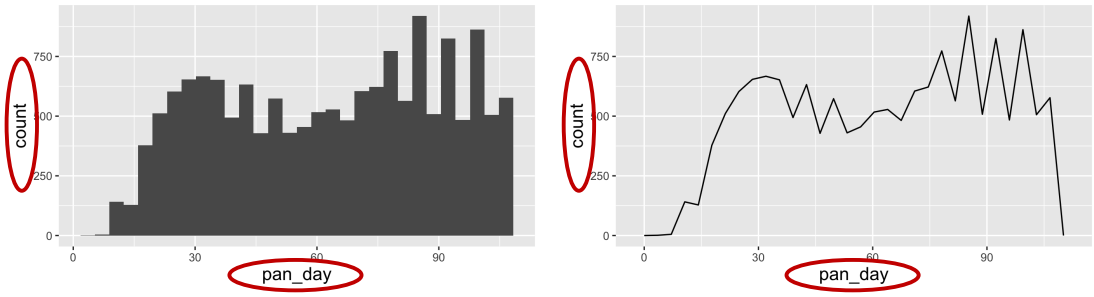



```
ggplot(data = covid_testing) +
  geom_histogram(mapping = aes(x = pan_day, fill = result))
```

```
ggplot(data = covid_testing) +
  geom_histogram(mapping = aes(x = pan_day), fill = "blue")
```

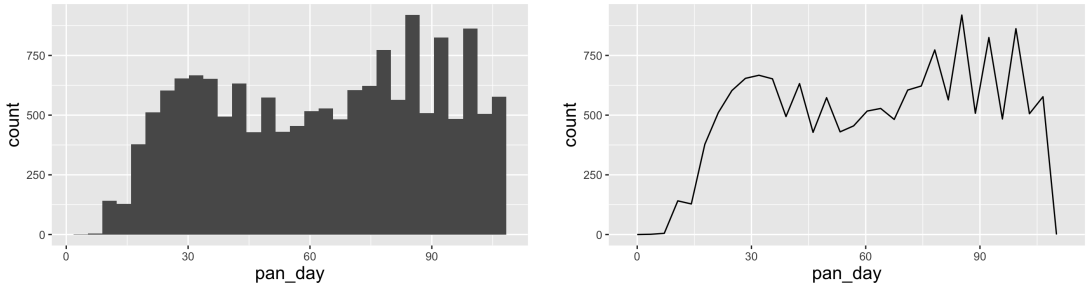
Geom Functions

How are these plots similar?



Same: x axis, y axis, data

How are these plots different?



Different **geometric object** ("geom") used to represent the data

Your Turn #6

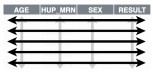
Return to `02 - Visualize.Rmd`. Work through the exercises of the section titled "Your Turn 6."

Click "yes" when you're done!

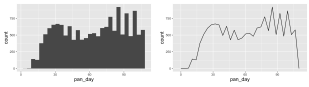
05:00

Recap

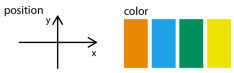
ggplot2 is a package that provides a **grammar of graphics**. You can create **any type of plot** using a simple template to which you provide:



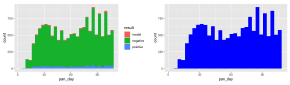
1. A **tidy data frame**, in which each **variable** is in its own **column**, each **observation** is in its own **row**, each **value** is in its own **cell**;



2. A **geom function**, which tells R what kind of plot to make; and

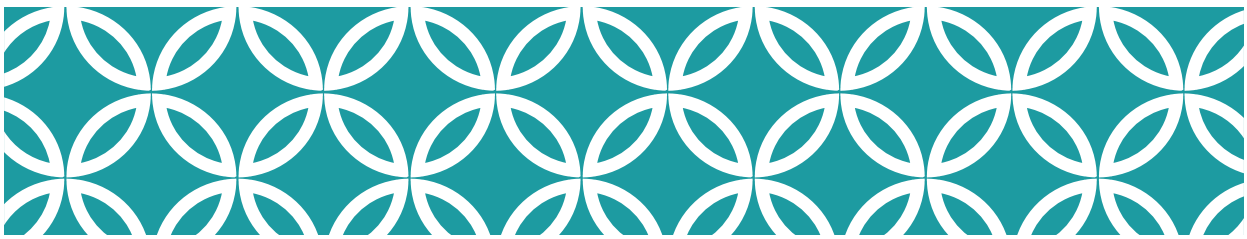


3. **Aesthetic mappings**, which tell R how to represent data as graphical markings on the plot.

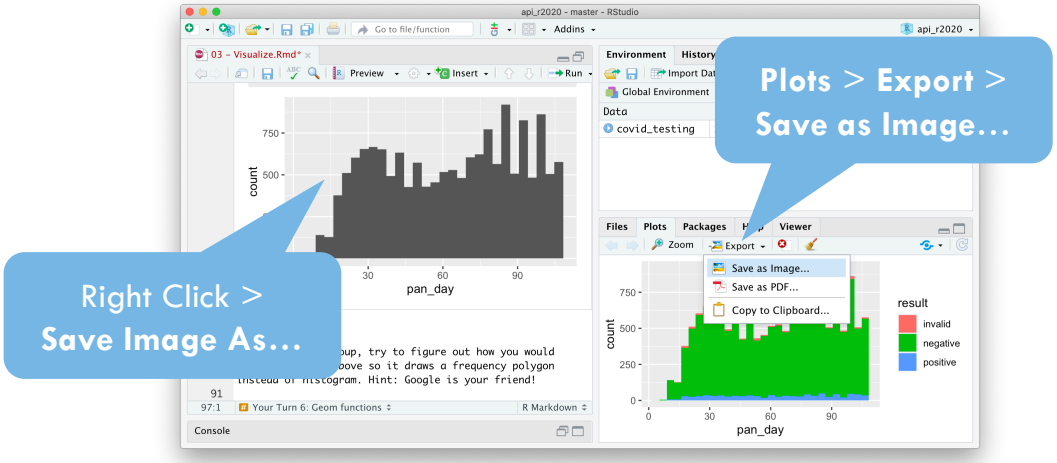


Aesthetics can be **mapped** to a variable or **set** to a constant value.

What Else?

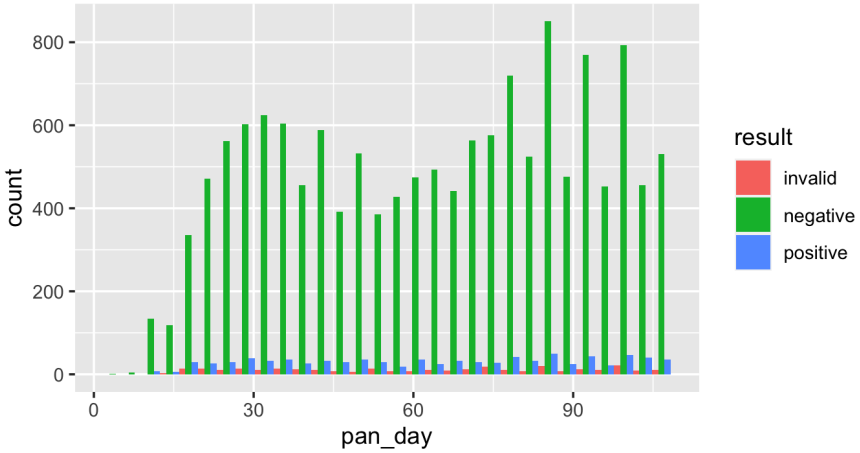


Manually saving plots



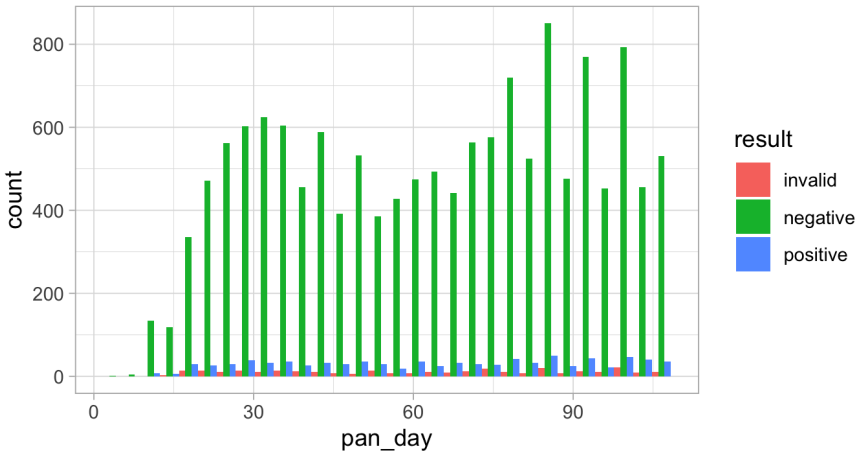
Position adjustments

How overlapping objects are arranged



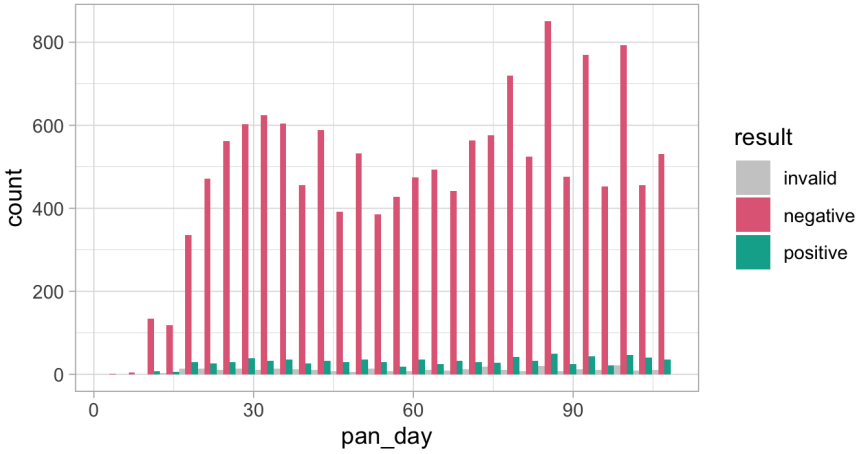
Themes

Visual appearance of non-data elements



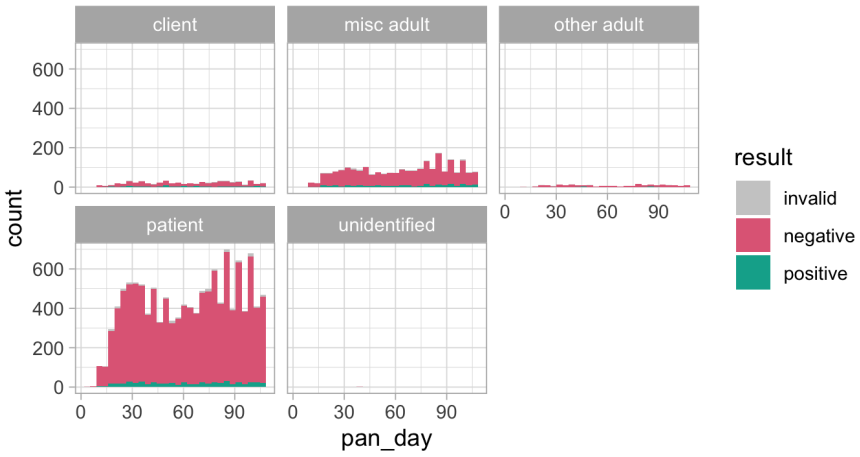
Scales

Customize color scales and other mappings

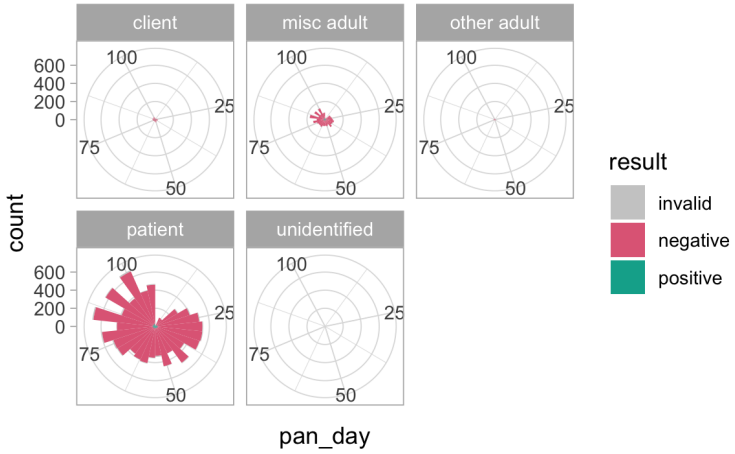


Facets

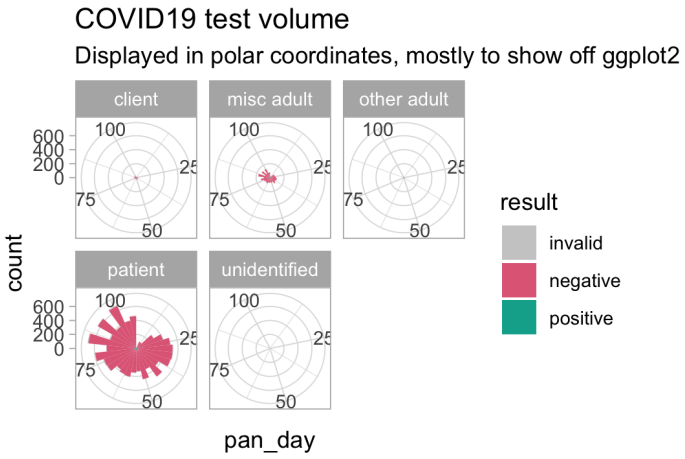
Subplots that display subsets of the data



Coordinate systems



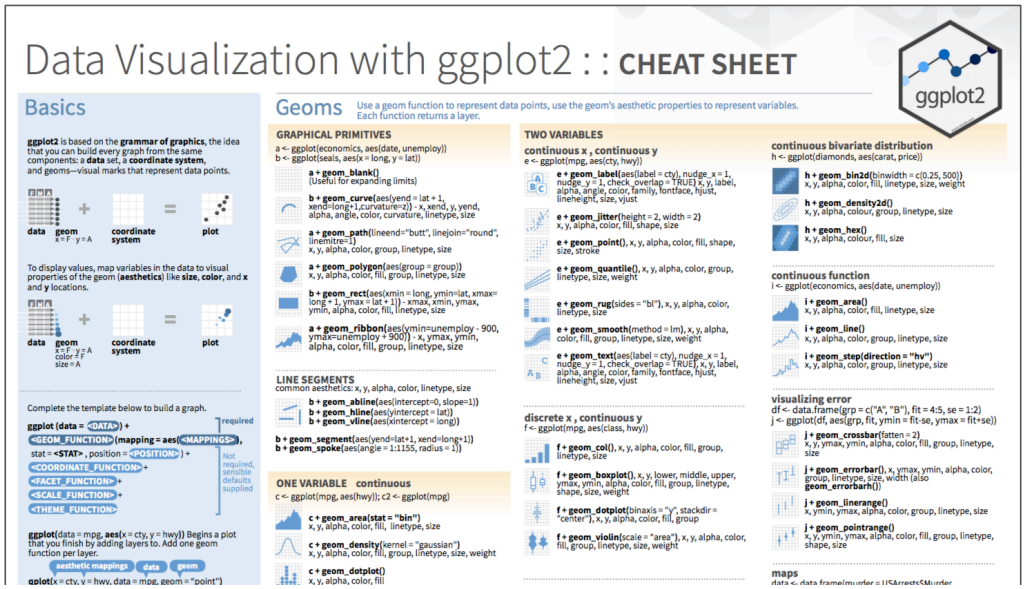
Titles and captions




```
ggplot(data = data_frame) +
  geom_function(mapping = aes(mappings)) +
  theme_function +
  scale_function +
  facet_function +
  coordinate_function +
  ...
```

Required

Optional



Data Visualization with ggplot2 : : **CHEAT SHEET**

ggplot2

Basics

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.

data + geom (x = F, y = A) = coordinate system + plot

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

data + geom (x = F, y = A, color = F, size = A) = coordinate system + plot

Complete the template below to build a graph.

required

```
ggplot (data = <DATA>) +
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
stat = <STAT>, position = <POSITION>) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

Not required, sensible defaults supplied

ggplot(data = mpg, **aes**(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings | data | geom

```
ggplot(x = cty, y = hwy, data = mpg, geom = "point")
```

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```

a + geom_blank()
(Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1, xend=long+1,curvature=z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round", linemitre=1)
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(group = group))
x, y, alpha, color, fill, group, linetype, size

b + geom_rect(aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept=0, slope=1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

b + geom_segment(aes(yend=lat+1, xend=long+1))
b + geom_spoke(aes(angle = 1:1155, radius = 1))

ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
```

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

TWO VARIABLES

continuous x , continuous y

```
e <- ggplot(mpg, aes(cty, hwy))
```

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE) x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_jitter(height = 2, width = 2)
x, y, alpha, color, fill, shape, size

e + geom_point(), x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"), x, y, alpha, color, linetype, size

e + geom_smooth(method = lm), x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE), x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y

```
f <- ggplot(mpg, aes(class, hwy))
```

f + geom_col(), x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = "center"), x, y, alpha, color, fill, group

f + geom_violin(scale = "area"), x, y, alpha, color, fill, group, linetype, size, weight

continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))
```

h + geom_bin2d(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

h + geom_density2d()
x, y, alpha, colour, group, linetype, size

h + geom_hex()
x, y, alpha, colour, fill, size

continuous function

```
i <- ggplot(economics, aes(date, unemploy))
```

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))
```

j + geom_crossbar(fatten = 2)
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

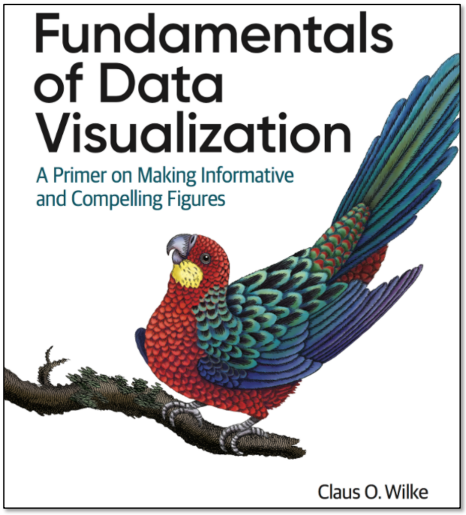
j + geom_errorbar(), x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)

j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

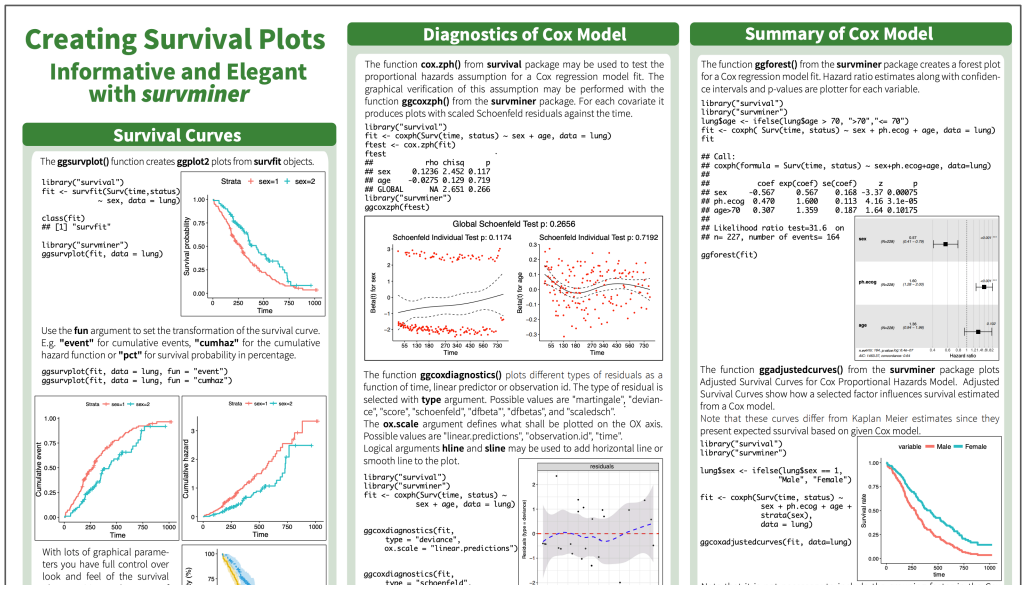
j + geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

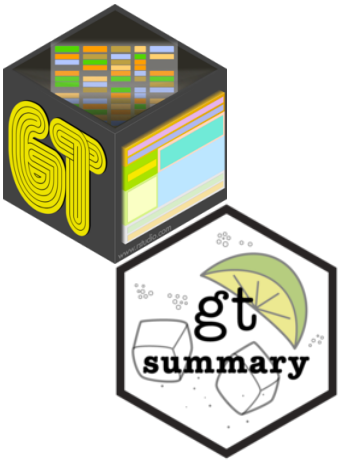
```
data <- data.frame(murder = USArrests$Murder
```



https://serialmentor.com/dataviz



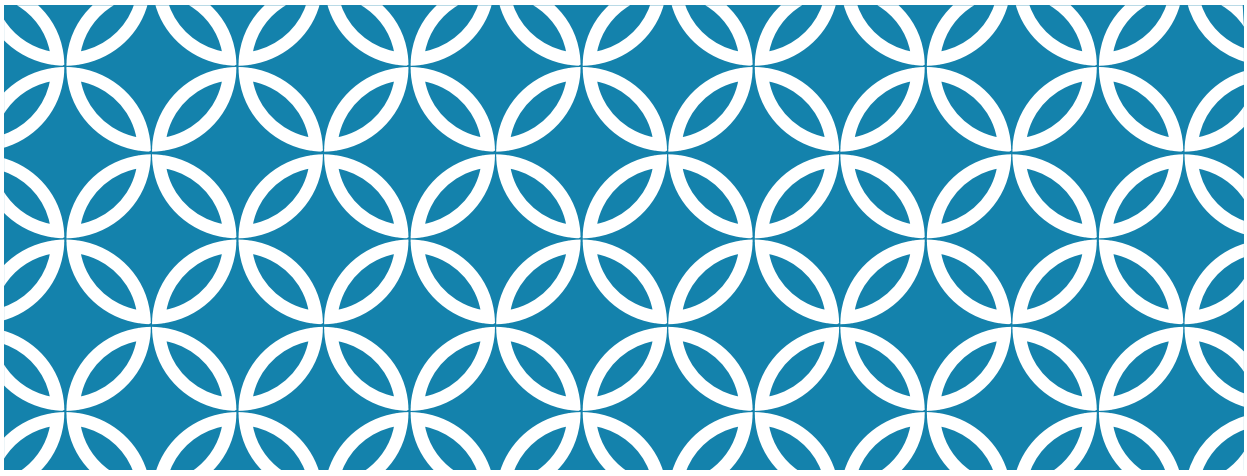
A grammar for *tables*



Variable	N	Drug A, N = 98[1]	Drug B, N = 102[1]	p-value[2]
Age	189	46 (37, 59)	48 (39, 56)	0.7
Grade	200			0.9
I		35 (36%)	33 (32%)	
II		32 (33%)	36 (35%)	
III		31 (32%)	33 (32%)	
Tumor Response	193	28 (29%)	33 (34%)	0.6

[1] Statistics presented: median (IQR); n (%)

[2] Statistical tests performed: Wilcoxon rank-sum test; chi-square test of independence



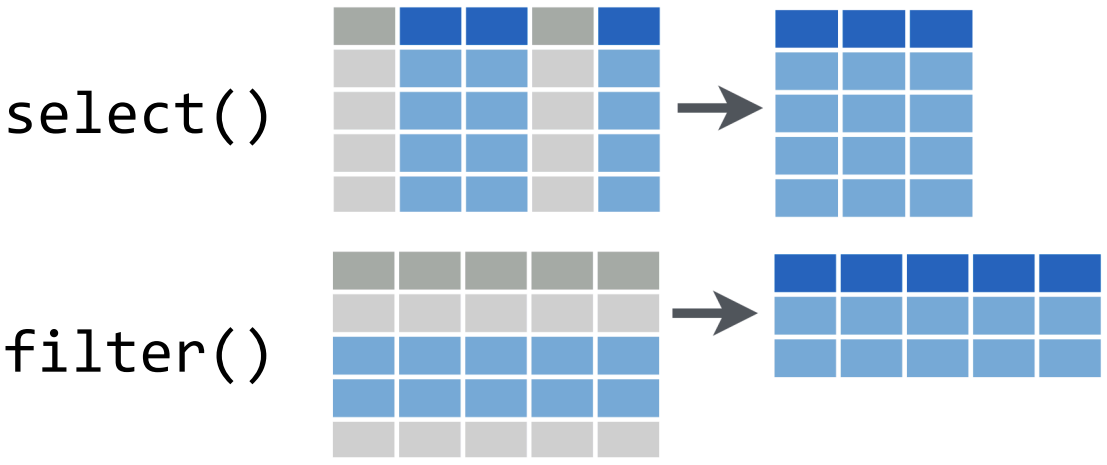
CHOP R 101:
Intro to R for Clinical Data

Transform

a grammar for transforming data frames

Subsetting data

Subsetting columns vs rows



select()

Extract columns by name.

```
select(data_frame, ...)
```

names of columns to extract

select()

Extract columns by name.

```
select(covid_testing, mrn, last_name)
```

covid_testing

mrn	first_name	last_name	gender
5000876	sarella	stark	female
5006017	alester	stark	male
5001412	jhezane	westerling	female
5000533	penny	targaryen	female

mrn	last_name
5000876	stark
5006017	stark
5001412	westerling
5000533	targaryen

Your Turn #1

Which of the following will select the `first_name` column from the `covid_testing` data frame and capture the result in a data frame named `newdata`?

A) `newdata = select(first_name, covid_testing)`

B) `newdata <- select(covid_testing, first_name)`

C) `select(newdata, covid_testing, first_name)`

D) `newdata <- select(covid_testing, First_Name)`

E) Both B and D

Type your response in the chat!

filter()

Extract rows that meet logical criteria.

```
filter(data_frame, condition)
```

logical test
(return each row for which the test is TRUE)

filter()

= sets
== compares

Extract rows that meet logical criteria.

```
filter(covid_testing, mrn == 5000083)
```

covid_testing

	mrn	first_name	last_name
FALSE	5000876	sarella	stark
FALSE	5006017	alester	stark
FALSE	5001412	jhezane	westerling
TRUE	5000083	lollys	clegane

mrn	first_name	last_name
5000083	lollys	clegane

Logical Operators

`x < y`	less than
`x > y`	greater than
`x == y`	equal to
`x <= y`	less than or equal to
`x >= y`	greater than or equal to
`x != y`	not equal to
`is.na(x)`	a missing value

Your Turn #2

Write a **`filter()`** statement that returns a data frame containing only the rows from **`covid_testing`** in which the **`last_name`** column is NOT equal to **`"stark"`**.

(You don't have to capture the returned data frame)

Type your response in the chat!

Your Turn #3

Which of these would successfully filter the `covid_testing` data frame to only tests with positive results?

A) `filter(covid_testing, result == positive)`

B) `filter(covid_testing, result = "positive")`

C) `filter(covid_testing, result == "positive")`

D) `filter(covid_testing, positive == "result")`

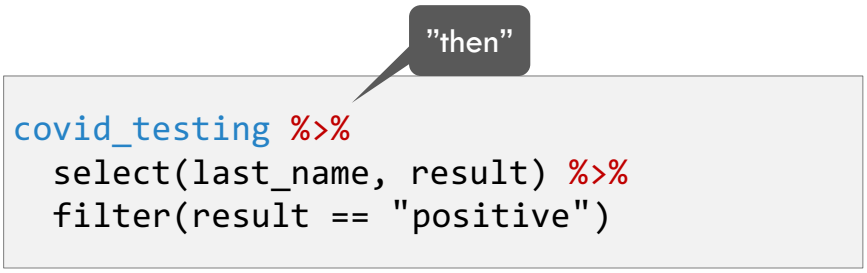
The pipe operator %>%

The Pipe Operator %>%

```
covid_testing %>% filter(_____, pan_day <= 10)
```

Passes the **object on the left** as **first argument** to the **function on the right.**

```
filter(covid_testing, pan_day <= 10)
covid_testing %>% filter(pan_day <= 10)
```



Your Turn #4

Rewrite the following statement with a pipe:

```
select(mydata, first_name, last_name)
```

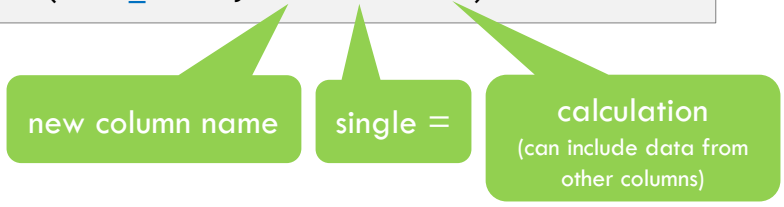
Type the answer in the chat!

Create new columns

mutate()

Create new calculated columns.

```
mutate(data_frame, name = value)
```



mutate()

Create new calculated columns.

```
mutate(covid_testing,
       col_rec_tat_mins = col_rec_tat * 60)
```

mrn	col_rec_tat
5000876	29.5
5006017	3.6
5001412	1.4
5000533	2.3

mrn	col_rec_tat	col_rec_tat_mins
5000876	29.5	1770
5006017	3.6	216
5001412	1.4	84
5000533	2.3	138

Your Turn #5

Open `03 - Transform.Rmd` and work through the exercises.

Click "yes" when you are finished.

Recap

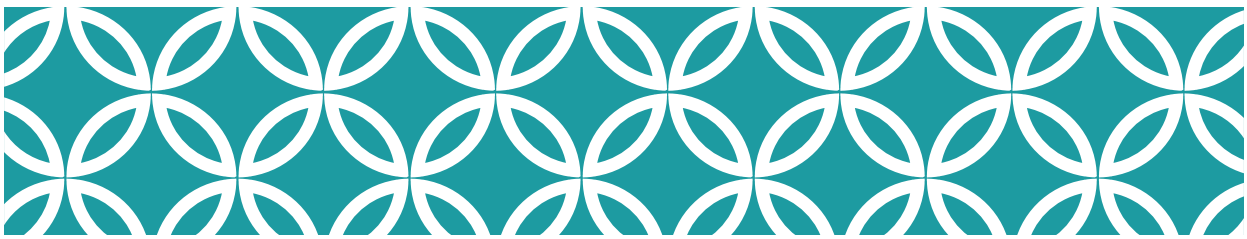
`select()` subsets columns by name

`filter()` subsets rows by a logical condition

`mutate()` creates new calculated columns

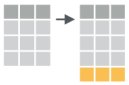
Use the **pipe operator** %>% to combine dplyr functions into a pipeline

What else?



More dplyr functions

```
arrange(data, ...)
```

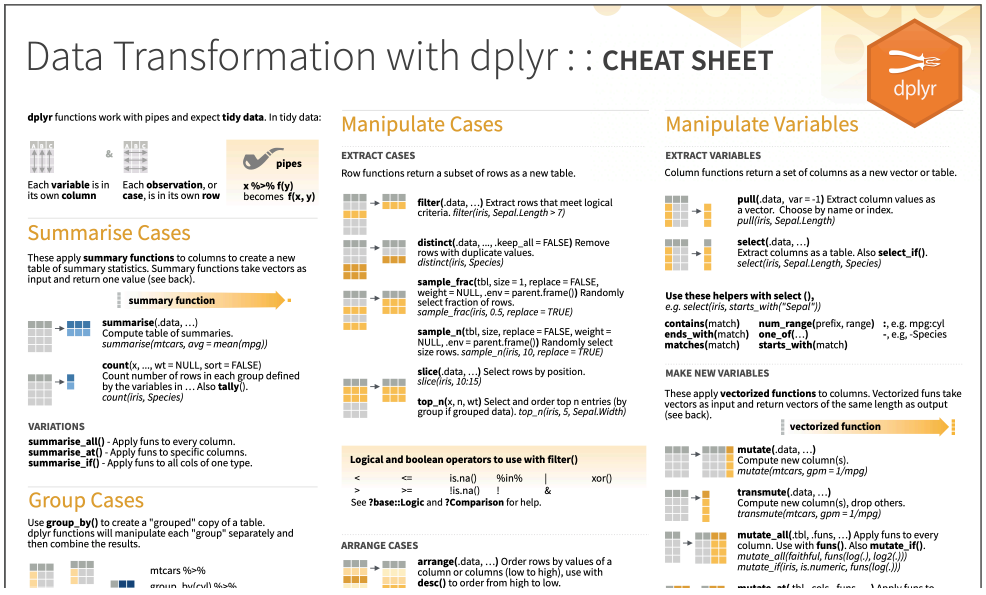


```
add_row(data, ...)
```

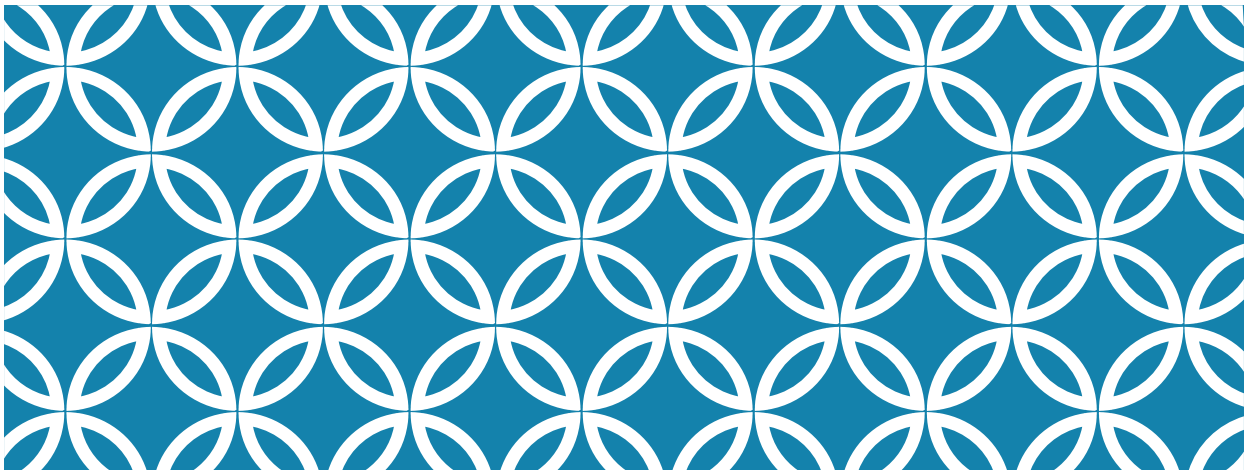
```
data %>%
  group_by(...) %>%
  summarize(...)
```


```
x %>%
  bind_cols(y)
```


```
x %>%
  left_join(y)
```



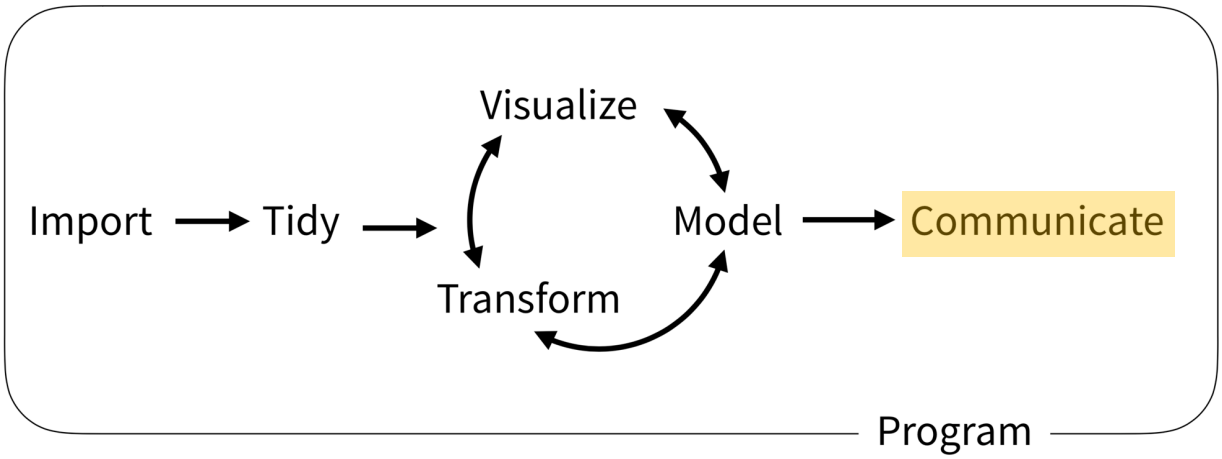

tidyr
tidyr.tidyverse.org
lubridate
www.rstudio.com
stringr
www.rstudio.com
purrr
www.rstudio.com
dbplyr
dbplyr.tidyverse.org



CHOP R 101:
Intro to R for Clinical Data

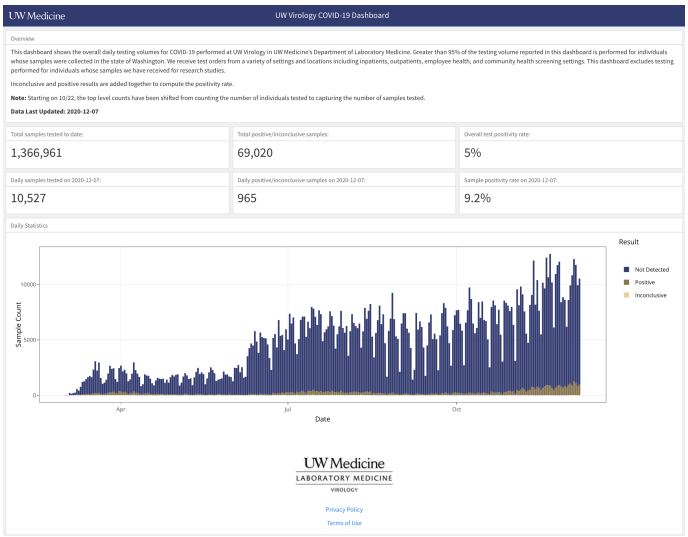
Dashboard

The Data Analysis Pipeline



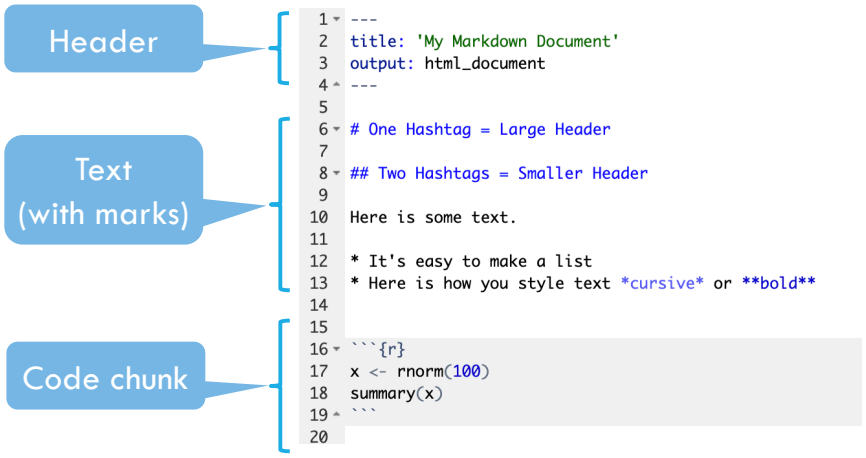
From *R for Data Science* (https://r4ds.had.co.nz/introduction.html)

Dashboards visualize key data on demand



Turn an R Markdown document into an interactive dashboard

Recap: Anatomy of an R Markdown Document




```
---
title: 'My Markdown Document'
output: html_document
---

# One Hashtag = Large Header

## Two Hashtags = Smaller Header

Here is some text.

* It's easy to make a list
* Here is how you style text *cursive* or **bold**


```{r}
x <- rnorm(100)
summary(x)
```

```

```
---
output:
  flexdashboard::flex_dashboard:
    orientation: columns

Column

Chart 1

```{r}
```

Column

Chart 2

```{r}
```

Chart C

```{r}
```

```

```
---
output:
  flexdashboard::flex_dashboard:
    orientation: rows

Row

Chart 1

```{r}
```

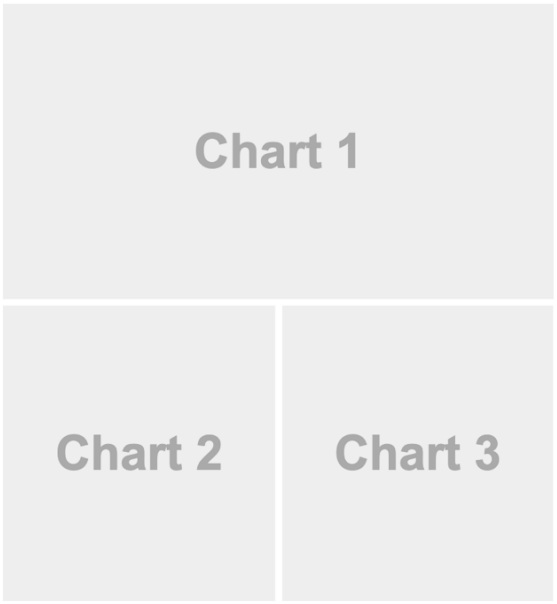
Row

Chart 2

```{r}
```

Chart C

```{r}
```
```



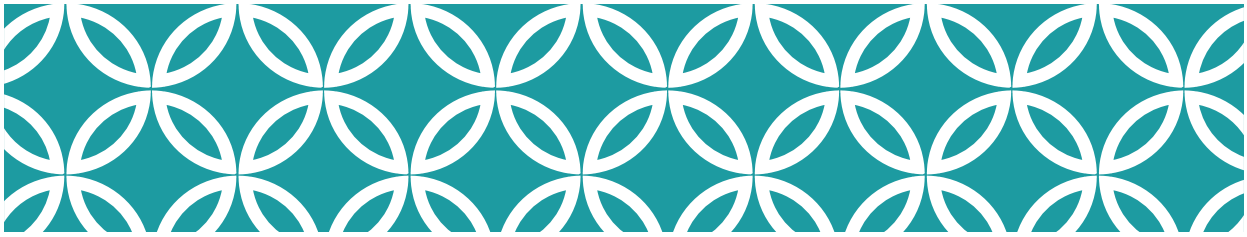
Your Turn #1

Open `04 - Dashboard.Rmd` to work with a draft COVID19 flexdashboard. Do the following:

1. Knit the document to see what the dashboard looks like.
2. Change the **Test Volumes Over Time** plot so that it visualizes the fraction of positive tests on a given day. (Hint: map the `fill` aesthetic to `result`). Knit the document again and note the change.
3. Change the **layout** from **columns** to a **row** orientation. Knit the document again and note the change.

Click "yes" when you are finished!

Making plots interactive



Plotly makes ggplots interactive

```
ggplot(data = data_frame) +
  geom_function(mapping = aes(mappings))
```

```
my_plot <- ggplot(data = data_frame) +
  geom_function(mapping = aes(mappings))
ggplotly(my_plot)
```

Your Turn #2

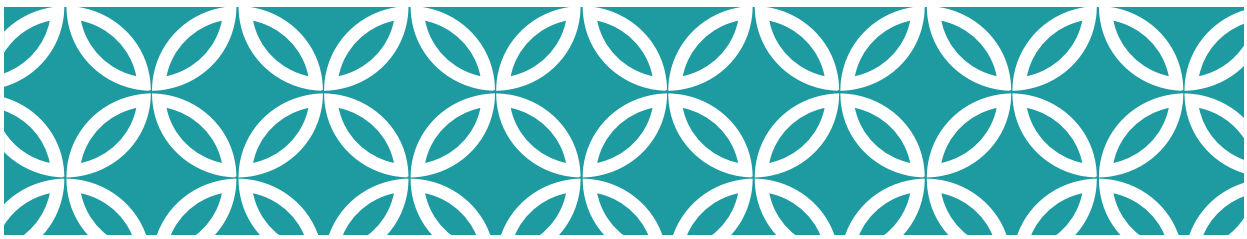
Return to `04 - Dashboard.Rmd`.

Make the **Test Volumes Over Time** plot interactive.

Knit the document again to note the change. Hover over the interactive plots. Note what happens when you click on the various buttons that appear above the plot.

Click "yes" when you are finished.

Interactive tables



DT makes data frames interactive

```
covid_testing
```

```
covid_testing %>%
  datatable()
```

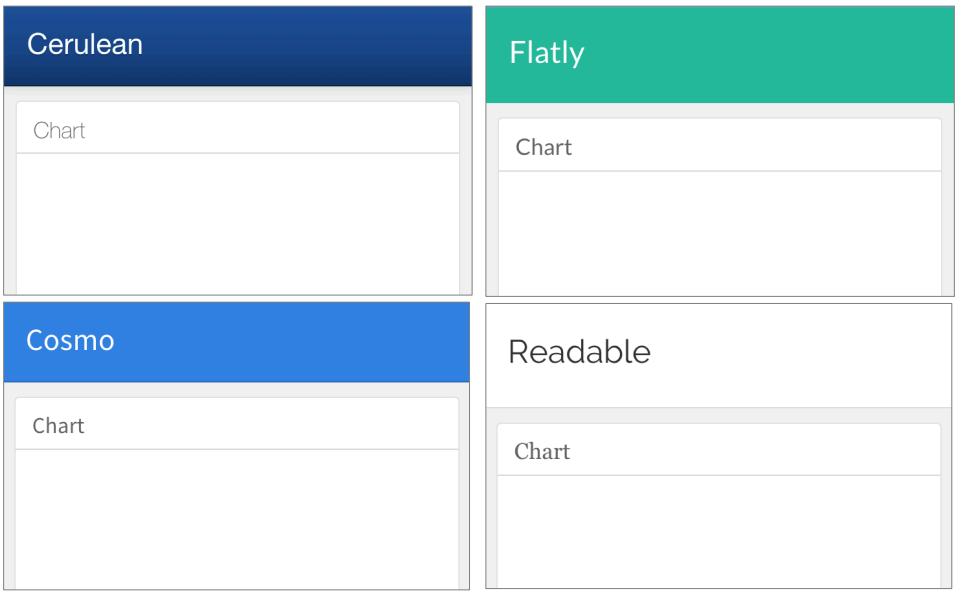
Your Turn #3

Return to `04 - Dashboard.Rmd`. Let's try to display a list of positive results in the **Details of Positive Test Results** area of the plot.

1. Note that this area currently shows the contents of the `covid_testing` data frame but doesn't actually filter for positive results. Add to the pipeline a `filter()` statement that selects only those rows where the `result` value is `"positive"` (Hint: `"positive"` must be in quotes). Knit the dashboard and note the change.
2. In order to use the `datatable()` function, we first need to load the `DT` package. Add a line to the `setup` chunk (all the way on top!) to load the `DT` package.
3. Now make the table interactive! Knit the dashboard and note the change.

Click "yes" when you are finished!

Add styling



Adding a theme

Header

```
---
title: "My Dashboard"
output:
  flexdashboard::flex_dashboard:
     orientation: rows
    theme: cerulean
---
```

https://rmarkdown.rstudio.com/flexdashboard/using.html#themes

COVID19 Dashboard Hackathon!

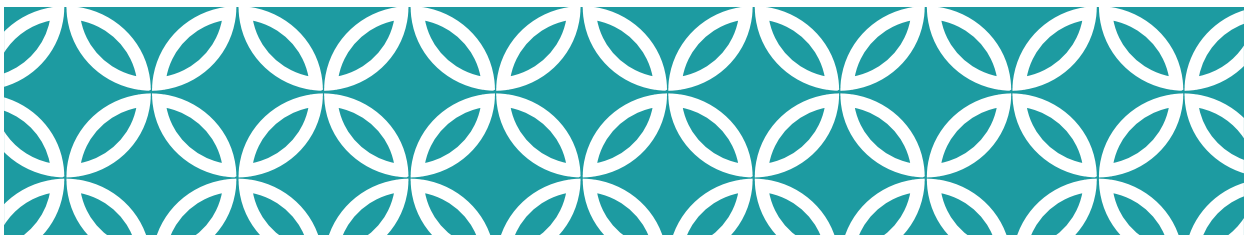
Recap

Turn R Markdown documents into interactive dashboards with `flexdashboard`

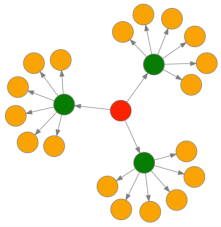
Turn ggplot objects into interactive plots with `plotly`

Turn data frames into interactive tables with `DT`

What else?

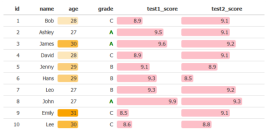


More interactive charts

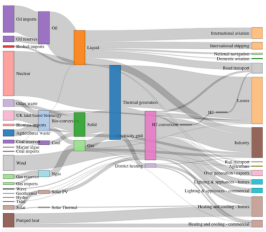


DiagrammeR Star 1281

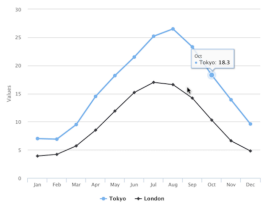
leaflet Star 578



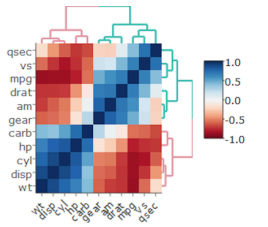
formattable Star 517



networkD3 Star 516



highcharter Star 507



heatmaply Star 230

https://gallery.htmlwidgets.org/

A framework for developing interactive web applications in R

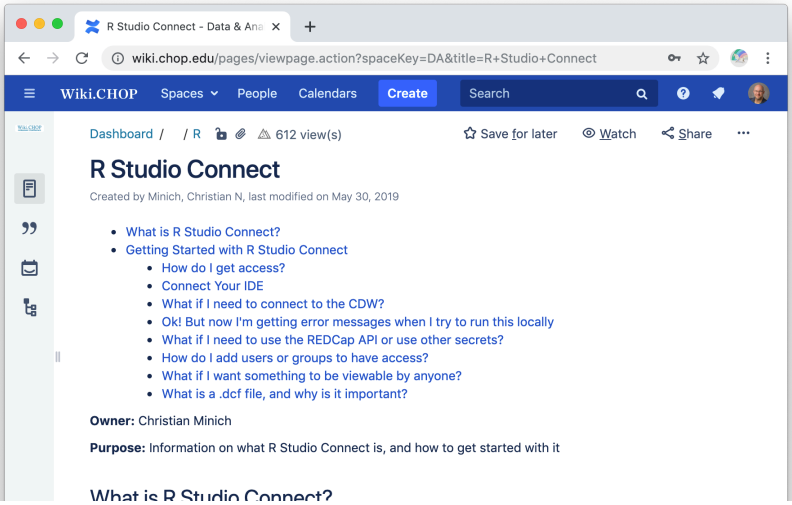
Deploying flexdashboards and Shiny apps

Shinyapps.io (free)

Shiny Server (free)
RStudio Connect ($)

CHOP RStudio Connect

wiki.chop.edu/display/DA/R+Studio+Connect



What next?

Install R and RStudio on your computer!

After the course

If you would like to practice with the materials from the course, go to the GitHub repository, and click the green button labeled "Code" to download the repository as a .ZIP file.

To install all the packages we used in the training environment, open RStudio and run the following command in the Console:

```
install.packages(c(
  "tidyverse",
  "rmarkdown",
  "shiny",
  "flexdashboard",
  "plotly",
  "DT"
))
```

https://skadauke.github.io/intro-to-r-for-clinicians-chop/

“Come” to a CHOP R User Group Meeting!

CHOP R User Group Meeting 🗣️

Thursday, Dec 10 at 2 PM

Meredith Lee and Cass Wilkinson Saldaña: *How to solve small, everyday work problems with R and R Markdown*

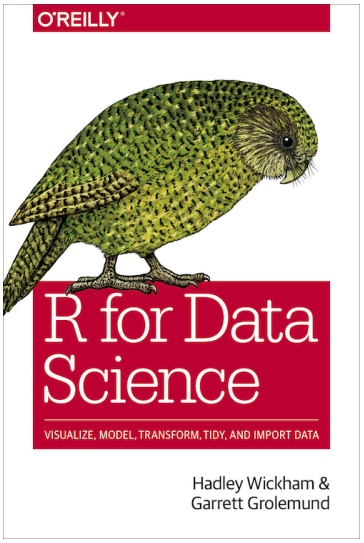
https://bluejeans.com/849314871

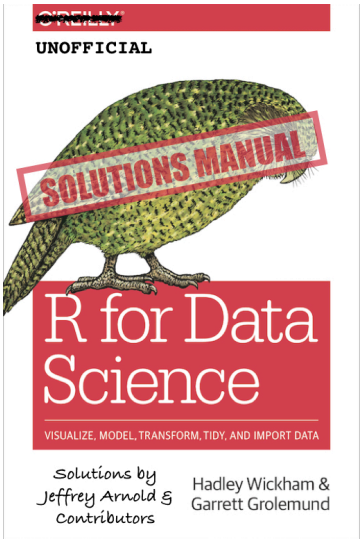
Do a Mini Project in R!

The only way to learn R is to actually spend some time writing R code.

- Identify a work problem that can be solved with a Mini R Markdown document
- See tomorrow’s CHOP R User Group meeting for sample Mini projects
- Kick-off meeting next week (time TBD)
- Get lots of support via CHOP R User Group and Arcus Education
- Optional: Give a Mini presentation at an upcoming CHOP R User Group meeting

https://bit.ly/chopr101-miniproject





https://r4ds.had.co.nz

https://jrnold.github.io/r4ds-exercise-solutions/