



FOUNDATION OF THE R WORKFLOW

The screenshot shows the R Studio interface with the following components:

- Top Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Tools, Help.
- File Tab:** chicagoFood.R x
- Code Editor:** R Script pane containing R code to subset a dataset and create a Leaflet map of Chicago food locations.
- Console:** Shows the same R code being run in the background.
- Data View:** Displays two datasets: 'data' (118607 obs. of 17 variables) and 'data1' (50 obs. of 17 variables).
- Plots:** A Leaflet map of Chicago with numerous blue location pins.
- Session Tab:** Shows 'Sessions (3)' and 'Radar'.

DS STATEMENT

- I have always been inspired by those who can capture the landscape with a minimum of brushstrokes

HELLO

my name is

Phil

MY BACKGROUND

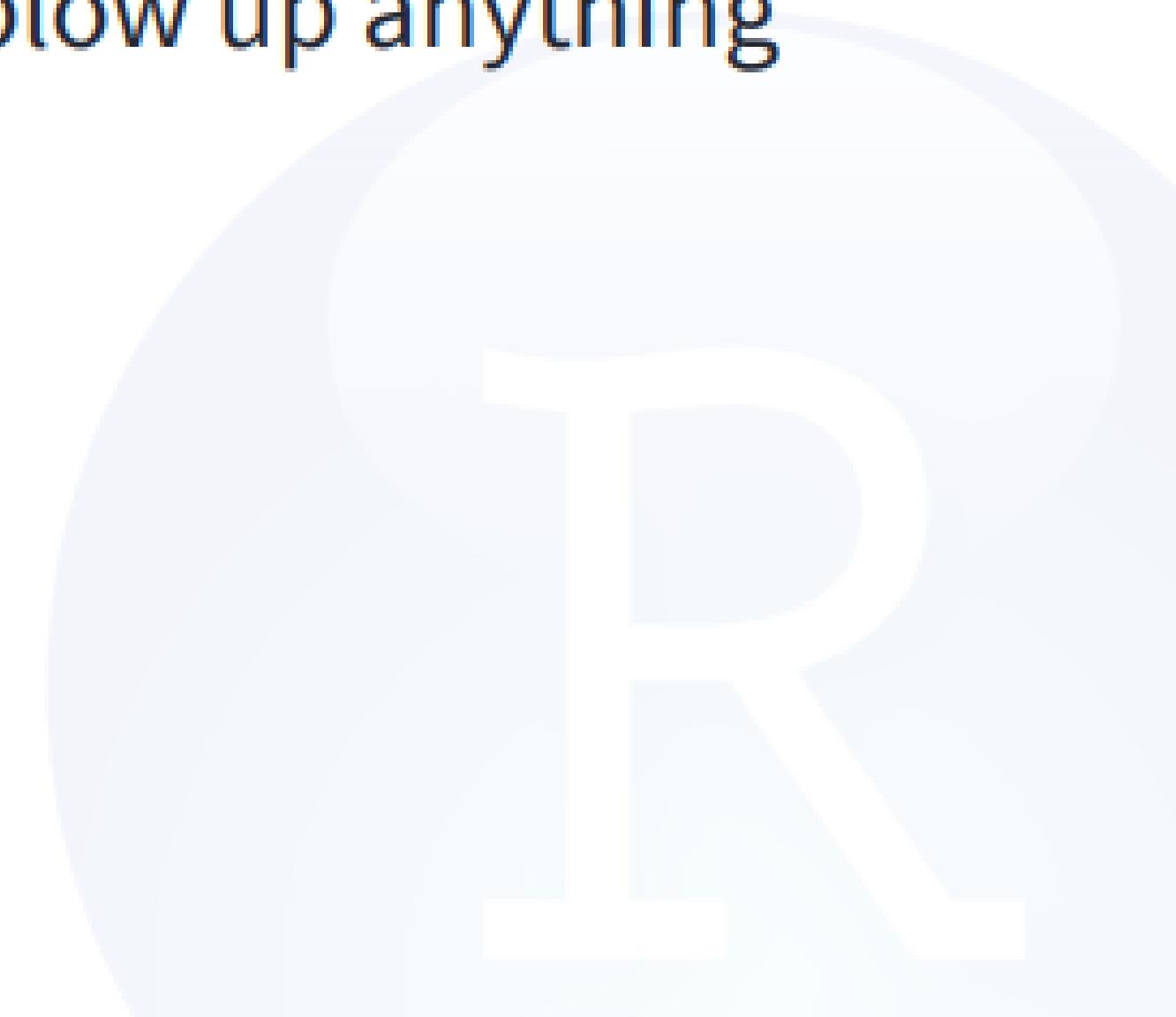
- Shiny
- CS
- Pharma
- Audience
- twitter:@rinpharma
- github:philbowsher
- Speed is the name of the game – fastest way to get you started
- Red is an Action Item for you

GOALS FOR TODAY

- Getting to Know RStudio
- Importing data
- Data Viz
- Data Step
- Procs
- ODS
- Macro Variables / Macros

THINGS YOU SHOULD KNOW

- There are no stupid questions
- Because we have limited time, please write down questions as they occur to you, and then ask them when the instructor pauses for questions
- You will not learn all of R today
- Don't be afraid to experiment and try things out; you won't blow up anything important if you make a mistake.



WORKSHOP COMMUNICATION

- Zoom Chat
- Polls & Breakout Rooms
- <https://calendly.com/rstudio-phil-bowsher/30min?month=2020-10>

Your Turn

Form groups of 2-4 people. Introduce yourself to your group members. Tell them:

1. Who you are
2. What you do with data
3. How long you have been using R



Quick Survey

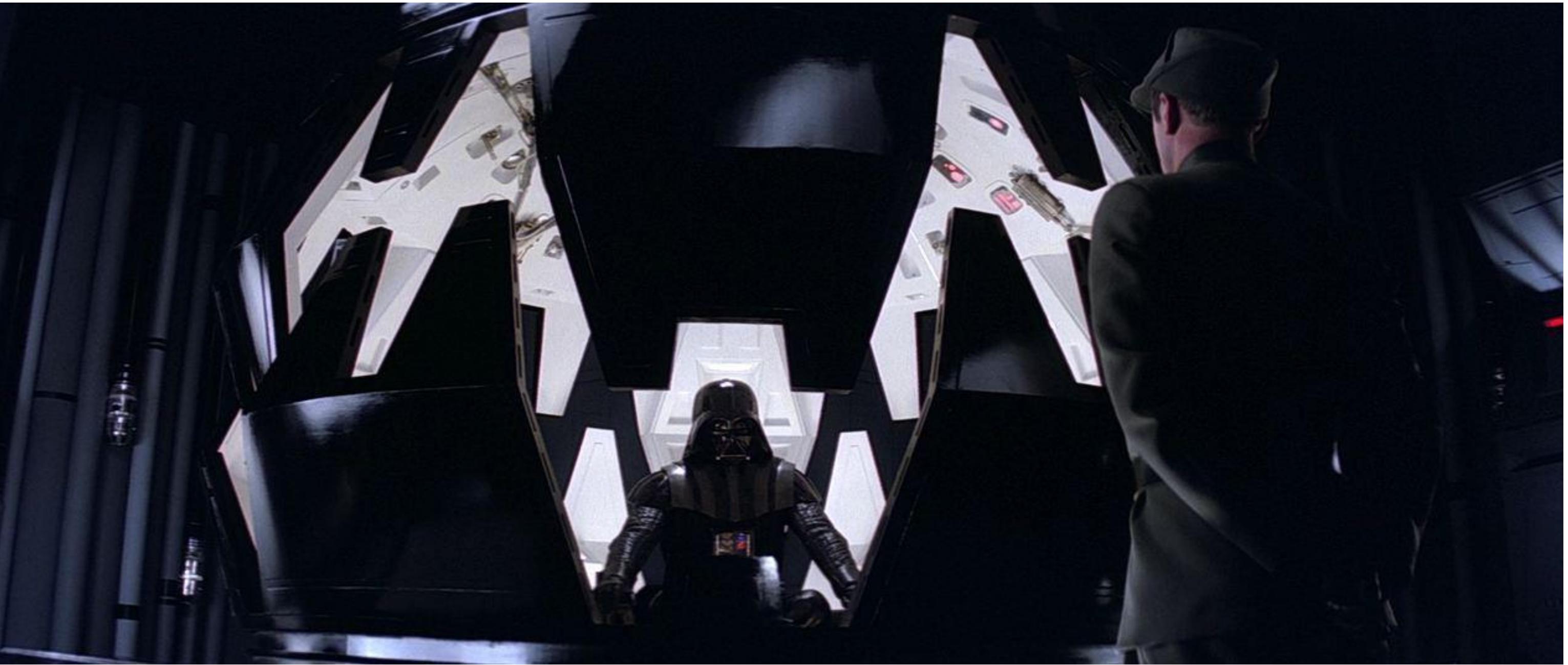
<http://rstd.io/phil-me-out>

SETUP IN RTT

- Setup
- <https://github.com/sol-eng/classroom-getting-started>
- <http://rstd.io/class>

SETUP IN RTT

- R
- Packages
- IDE
- Projects
- Sessions
- Git/Github
- RSC
- Shiny



Star Wars

Setup & Quick Intro (Panes & Buttons)



A language



Writing software



BUILT-IN DATASETS

- `data()`
- `data(ToothGrowth)`
- `?ToothGrowth`
- `ToothGrowth`
- `View(ToothGrowth)`
- `summary(ToothGrowth)`
- `plot(ToothGrowth)`

IDE – LET'S EXPLORE ...

- `# getwd()`
- `library(tidyverse)`
- `# let us explore the data set a bit`
- `names(ToothGrowth) # names of the variables`
- `dim(ToothGrowth) # dimension (number of rows and columns)`
- `str(ToothGrowth) # structure of the data set`
- `class(ToothGrowth)`
- `head(ToothGrowth, n = 5)`
- `tail(ToothGrowth, n = 5)`
- `ToothGrowth %>% write_csv('ToothGrowth.csv')`
- `ToothGrowth2 <- read_csv("ToothGrowth.csv")`

RSTUDIO.CLOUD OVERVIEW

The screenshot shows the RStudio Cloud homepage. At the top left is the R Studio Cloud logo. At the top right are links for "Log In", "Sign Up", and a menu icon. The main title "Welcome to RStudio Cloud" is displayed prominently, followed by the subtitle "alpha". Below that is the tagline "Do, share, teach and learn data science with R.". A green "Get Started" button is centered below the tagline. At the bottom of the main content area, there is a small note: "If you already have an RStudio shinyapps.io account, you can log in using your existing credentials." The background features a stylized blue and white cloud graphic.

THE MISSION

We created RStudio Cloud to make it easy for professionals, hobbyists, trainers, teachers and students to do, share, teach and learn data science using R.

To learn more about RStudio's other products, visit rstudio.com



After the workshop, go here:

https://rstudio.cloud/spaces/89287/join?access_code=V%2FvqOUv2%2FMCQF0jGlRPJnlAeyN41fegtHS0mpPB

BY THE WAY, BOOKS...

- <https://mastering-shiny.org/>
- <https://bookdown.org/yihui/rmarkdown/>
- <http://r4ds.had.co.nz> & <https://rstudio.cloud/>
- <https://r-graphics.org/>
- <http://www-bcf.usc.edu/~gareth/ISL/>
- <http://appliedpredictivemodeling.com/>
- <https://bookdown.org/yihui/rmarkdown/>
- <https://www.tidytextmining.com/>
- <https://adv-r.hadley.nz/>
- <https://plotly-r.com/>
- <https://therinspark.com/>
- <https://www.tidymodels.org/books/>

BREAK TIME

- 5-10 Min Break

YOUR TURN

- Form groups of 2-4 people
- Visual Analytics
- Have you used the Tidyverse?
- What data do you import?
- How much time do you spend cleaning data?

Import Data with



FIRST THINGS FIRST...DATA

Read data in

Read in data with `readr`, `haven`, `readxl`

`readr`

- `read_csv()`, `read_tsv()`, `read_delim()`

`haven`

- `read_sas()`, `read_spss()`, `read_stata()`, `read_dta()`

`readxl`

- `read_xls()`, `read_xlsx()`, `read_excel()`

Importing Data

R

readr

Simple, consistent functions for working with strings.

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





Compared to `read.table` and its derivatives,
`readr` functions are:

1. ~ 10 times faster
2. Return tibbles
3. Have more intuitive defaults. No row names, no strings as factors.



readr functions

function	reads
read_csv()	Comma separated values
read_csv2()	Semi-comma separated values
read_delim()	General delimited files
read_fwf()	Fixed width files
read_log()	Apache log files
read_table()	Space separated
read_tsv()	Tab delimited values



read_csv()

readr functions share a common syntax

```
df <- read_csv("path/to/file.csv", ...)
```

object to save
output into

path from working
directory to file

Import Text Data

File/URL:

~/Abbott Workshop/The-Little-R-Workshop-A-Primer-master/Examples/data/landdata-states.csv

Update

Data Preview:

State (character)	region (character)	Date (double)	Home.Value (double)	Structure.Cost (double)	Land.Value (double)	Land.Share..Pct. (double)	Home.Price.Index (double)	Land.Price.Index (double)	Year (double)
AK	West	2010.25	224952	160599	64352	28.6	1.481	1.552	2010
AK	West	2010.50	225511	160252	65259	28.9	1.484	1.576	2010
AK	West	2009.75	225820	163791	62029	27.5	1.486	1.494	2009
AK	West	2010.00	224994	161787	63207	28.1	1.481	1.524	2009

Previewing first 50 entries.

Import Options:

Name: <input type="text" value="landdata_states"/>	<input checked="" type="checkbox"/> First Row as Names	Delimiter: <input type="button" value="Comma"/>	Escape: <input type="button" value="None"/>
Skip: <input type="text" value="0"/>	<input checked="" type="checkbox"/> Trim Spaces	Quotes: <input type="button" value="Default"/>	Comment: <input type="button" value="Default"/>
	<input checked="" type="checkbox"/> Open Data Viewer	Locale: <input type="button" value="Configure..."/>	NA: <input type="button" value="Default"/>

Code Preview:

```
library(readr)
landdata_states <- read_csv("Abbott Workshop/The-Little
-R-Workshop-A-Primer-master/Examples/data/landdata
-states.csv")
View(landdata_states)
```

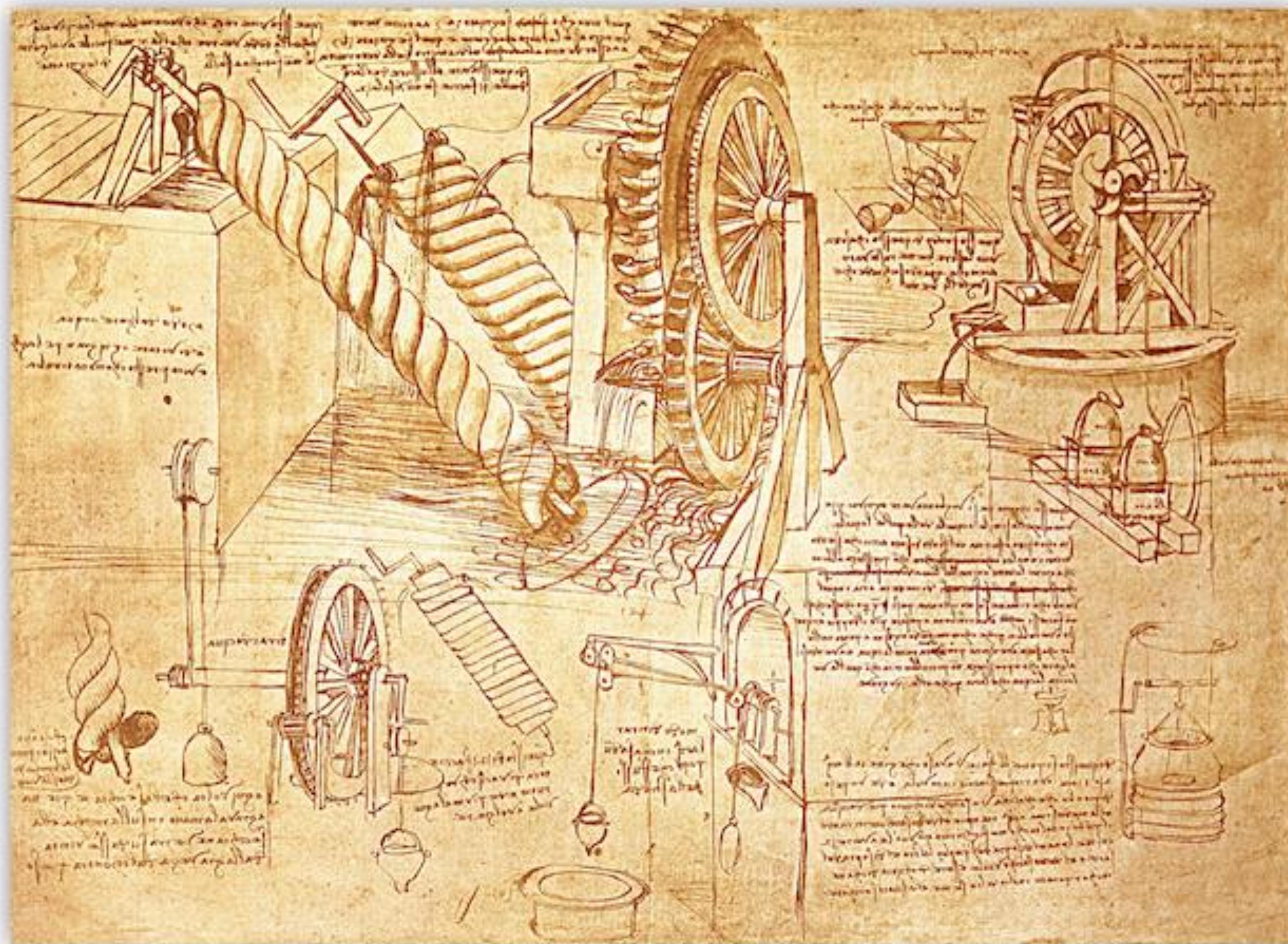
[? Reading rectangular data using readr](#)

Import

Cancel

Let's Chat about Notebooks...

Leonardo da Vinci...Page from the Codex Atlanticus shows notes and images about water wheels and Archimedean Screws



Notebooks

- Number 3: Notebooks are for doing science
- Number 2: R Notebooks have great features
- Number 1: R Notebooks make it easy to create and share reports

<https://rviews.rstudio.com/2017/03/15/why-i-love-r-notebooks/>
<http://r4ds.had.co.nz/r-markdown-workflow.html>

Notebooks

The screenshot shows the RStudio Source Editor with an R Markdown file named "nb-demo.Rmd". The code chunk at line 10 contains the command `summary(iris)`, which is executed and its output is displayed in a code evaluation panel. The output shows statistical summaries for Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, and Species. The code chunk at line 16 contains the command `qplot(Sepal.Length, Petal.Length, data = iris, color = Species, size = Petal.Width)`, which is run and its output is displayed in a plot evaluation panel. The plot is a scatter plot of Sepal.Length (x-axis) vs Petal.Length (y-axis). Data points are colored by Species (setosa in red, versicolor in green, virginica in blue) and sized by Petal.Width. A legend on the right side of the plot panel shows the mapping between petal width values (0.5, 1.0, 1.5, 2.0, 2.5) and bubble sizes.

```
9
10 ````{r}
11 summary(iris)
12 ````

Sepal.Length Sepal.Width Petal.Length Petal.Width Species
Min. :4.300 Min. :2.000 Min. :1.000 Min. :0.100 setosa :50
1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.300 versicolor:50
Median :5.800 Median :3.000 Median :4.350 Median :1.300 virginica :50
Mean :5.843 Mean :3.057 Mean :3.758 Mean :1.199
3rd Qu.:6.400 3rd Qu.:3.300 3rd Qu.:5.100 3rd Qu.:1.800
Max. :7.900 Max. :4.400 Max. :6.900 Max. :2.500

13
14 ````{r}
15 library(ggplot2)
16 qplot(Sepal.Length, Petal.Length, data = iris, color = Species, size =
Petal.Width)
17 ````
```

Combine in a single document:

- Narrative
- Code
- Output

Then Render to HTML

Setup

The setup chunk is always run once before anything else
and is a great place to load packages

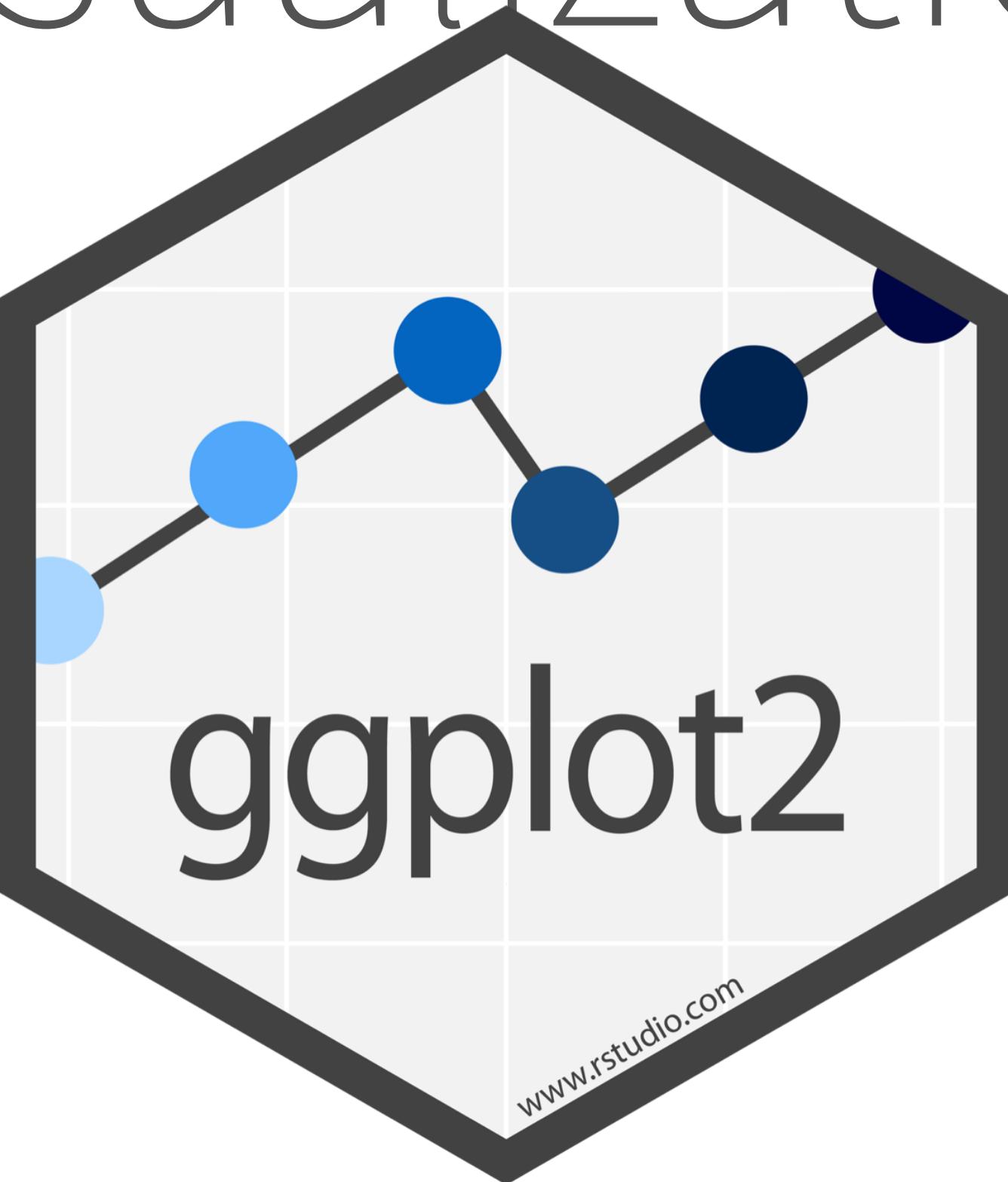
The screenshot shows an RStudio interface. The code editor pane contains R code with numbered lines from 1 to 37. Lines 1 through 18 are part of a setup chunk:

```
1 - ...
2 - title: "Data Visualization"
3 - output: html_notebook
4 - ...
5 -
6 - ```{r setup}
7 - library(tidyverse)
8 - ```
9 -
10 - ```{r}
11 - mpg
12 - ...
13 -
14 -
15 - # Your Turn 1
16 -
17 - Run the code on the slide to make a graph. Pay strict attention to
    spelling, capitalization, and parentheses!
18 -
19 - ```{r}
20 -
21 - ...
22 -
23 - # Your Turn 2
24 -
25 - Add "color", "size", "alpha", and "shape" aesthetics to your graph.
    Experiment.
26 -
27 - ```{r}
28 - ggplot(data = mpg) +
29 -   geom_point(mapping = aes(x = displ, y = hwy))
30 - ...
31 -
32 - # Your Turn 3
33 -
34 - Replace this scatterplot with one that draws boxplots. Use the cheatsheet.
    Try your best guess.
35 -
36 - ```{r}
37 - ggplot(mpg) + geom_point(aes(class, hwy))
38 - ...
39 -
```

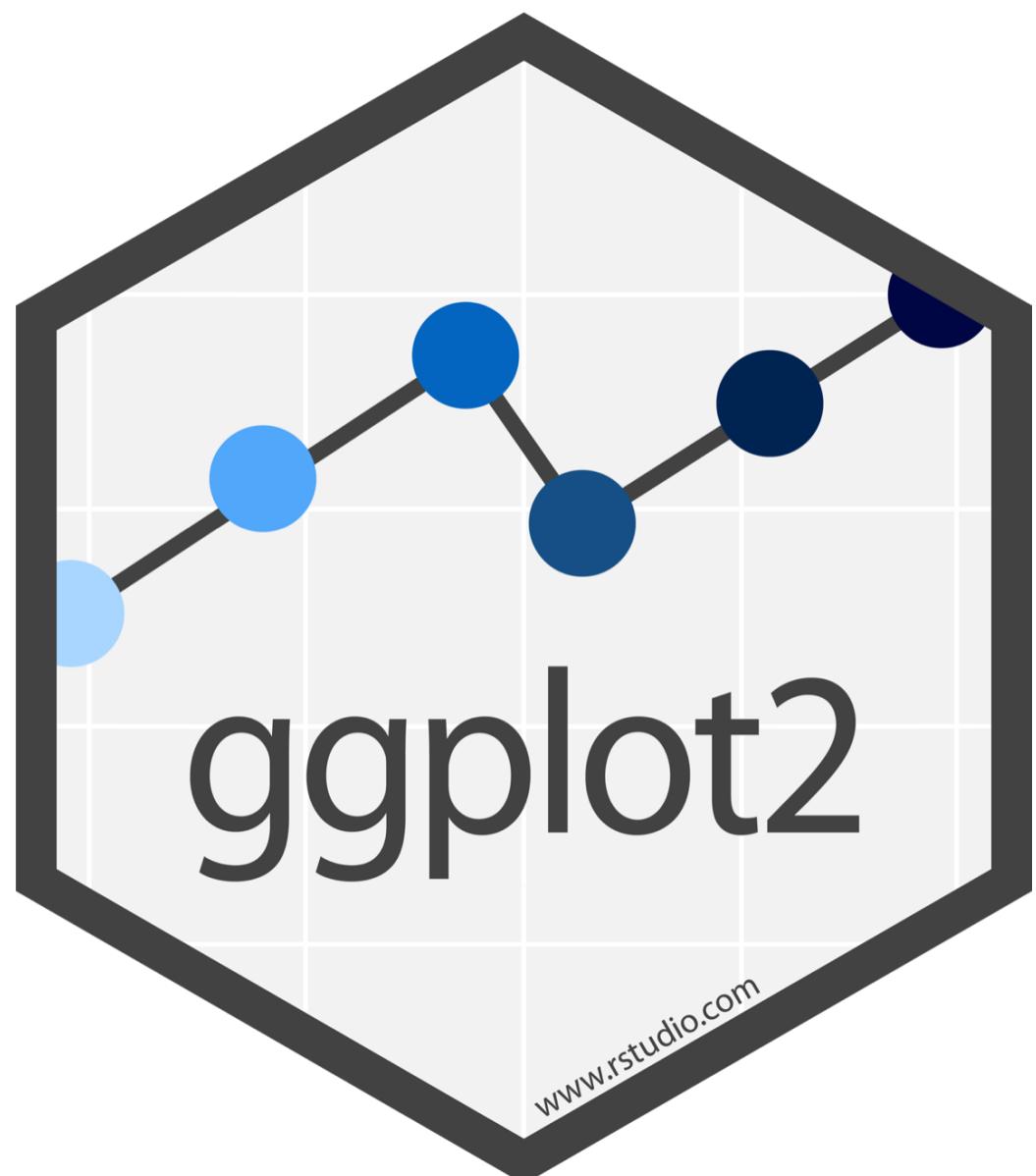
The preview pane displays the output of the R code. It shows a large, bold, blue text block containing the command `library(tidyverse)`. Above this text, there is a decorative graphic consisting of three black arrows pointing upwards and to the right, forming a triangular shape.

chunk labels are optional,
the setup label is special

Data Visualization with

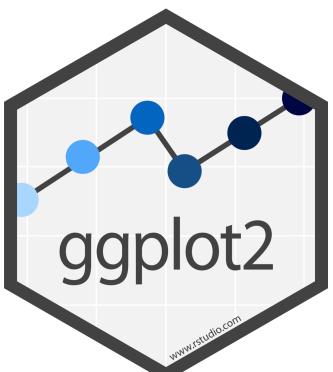


ggplot2



A package that visualizes data.

ggplot2 implements the *grammar of graphics*, a system for building visualizations that is built around **cases** and **variables**.



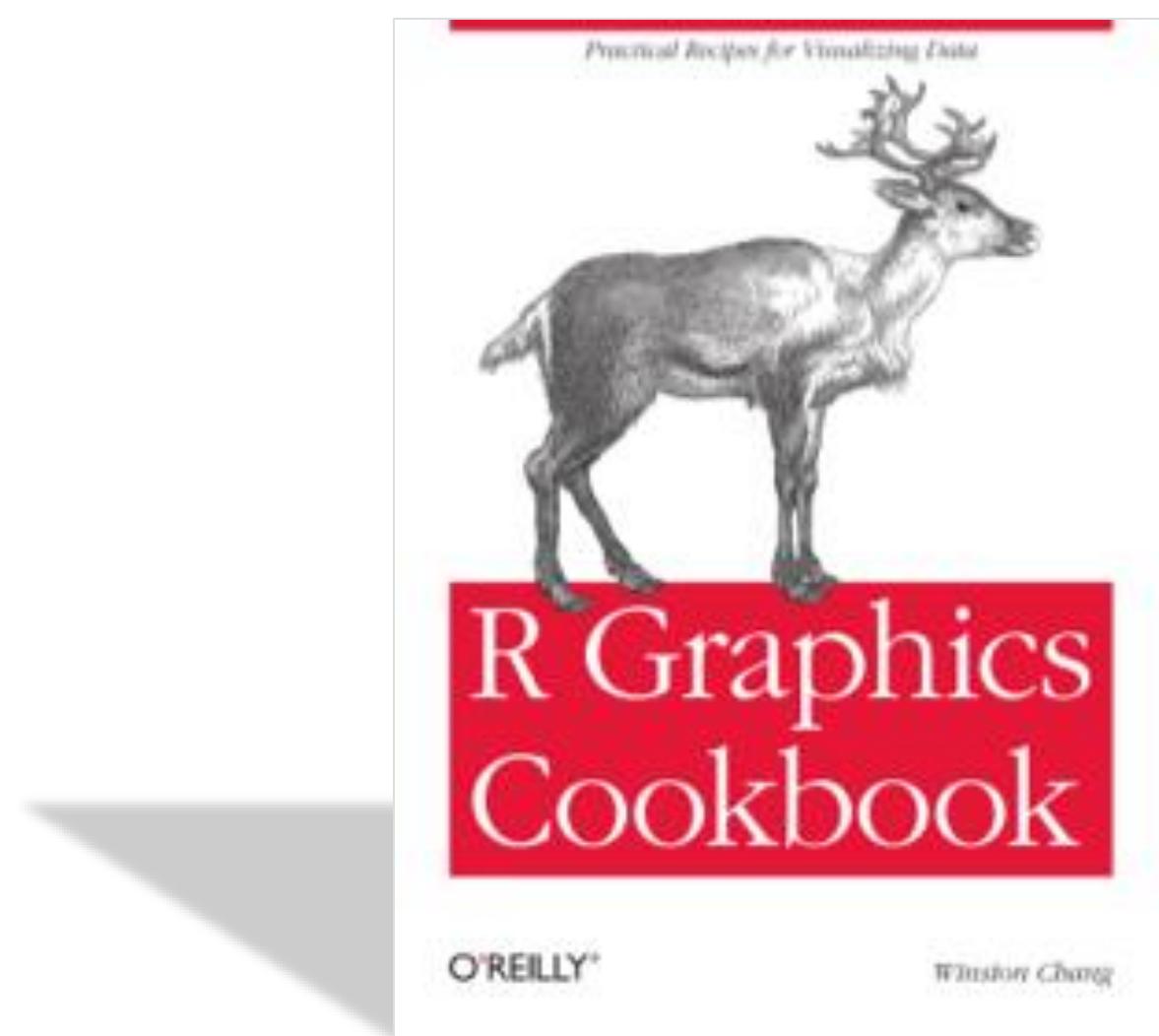
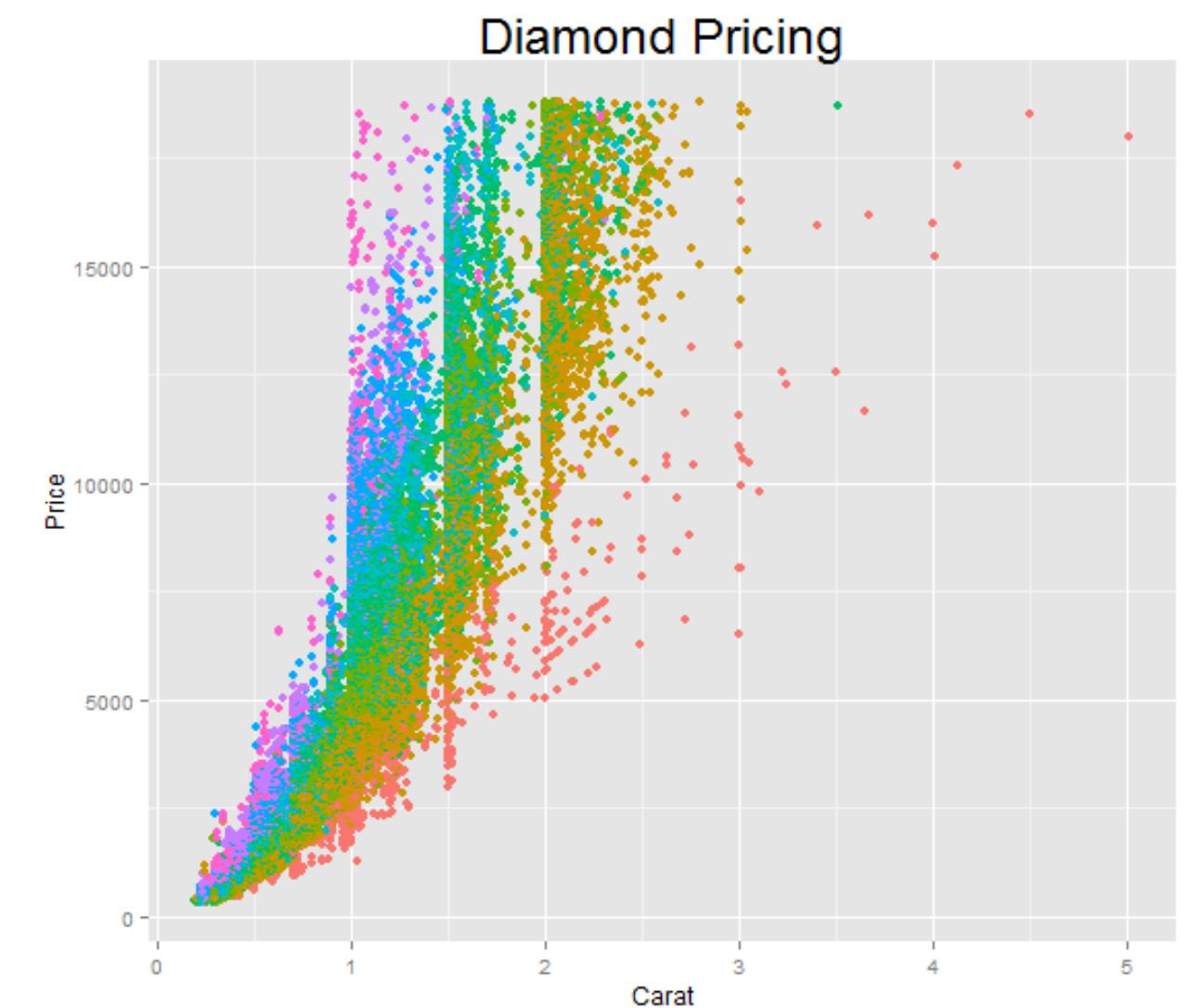
R Graphics:

Four Main Graphical Systems in R:

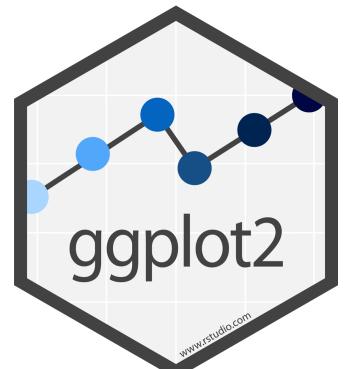
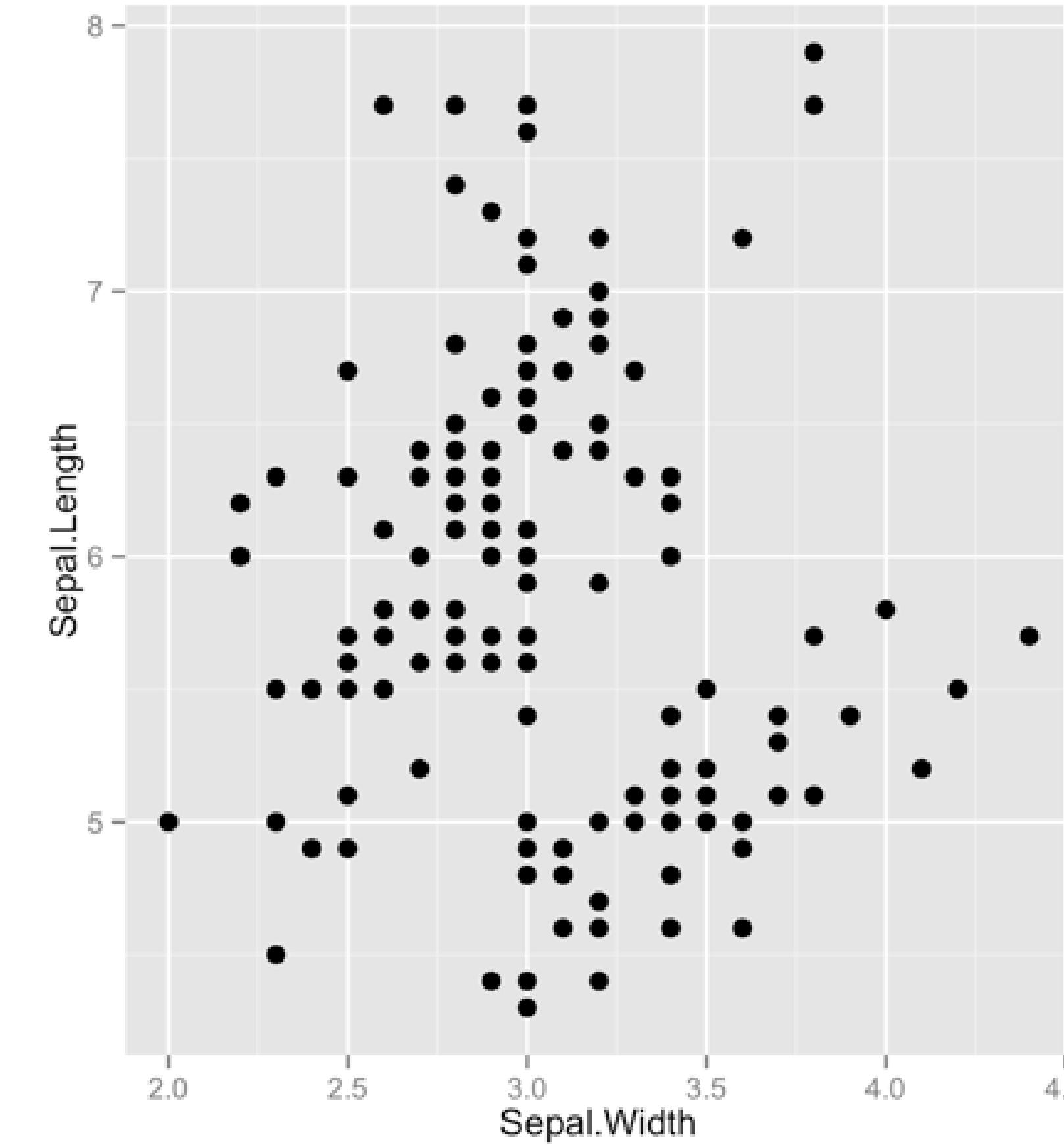
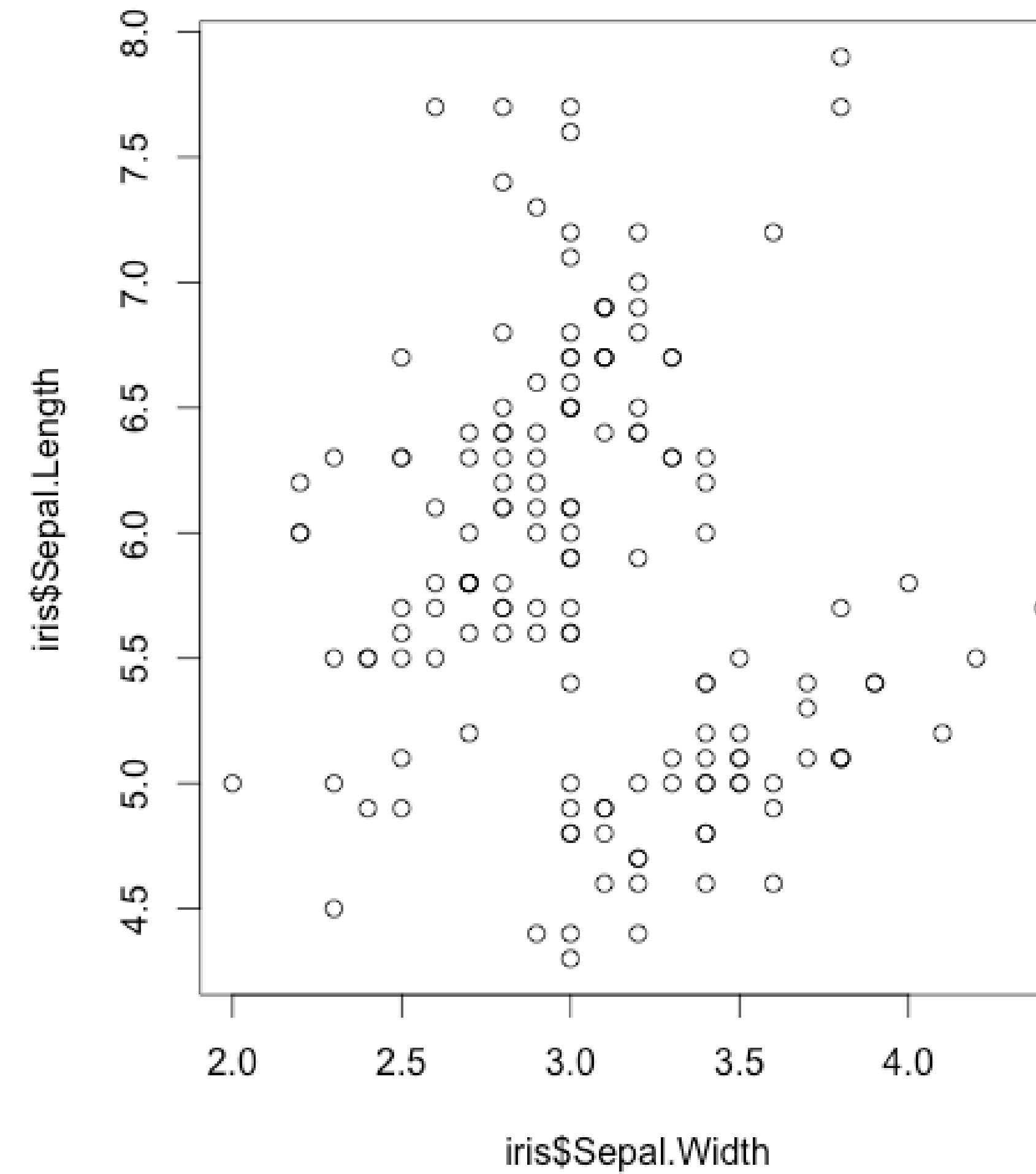
- R's Base Graphics
- Grid Graphics System
- The lattice Package
- The ggplot2 Package – Created by Hadley Wickham

Why ggplot2?

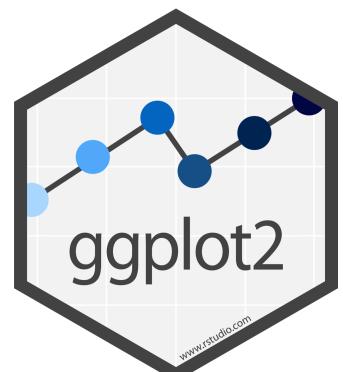
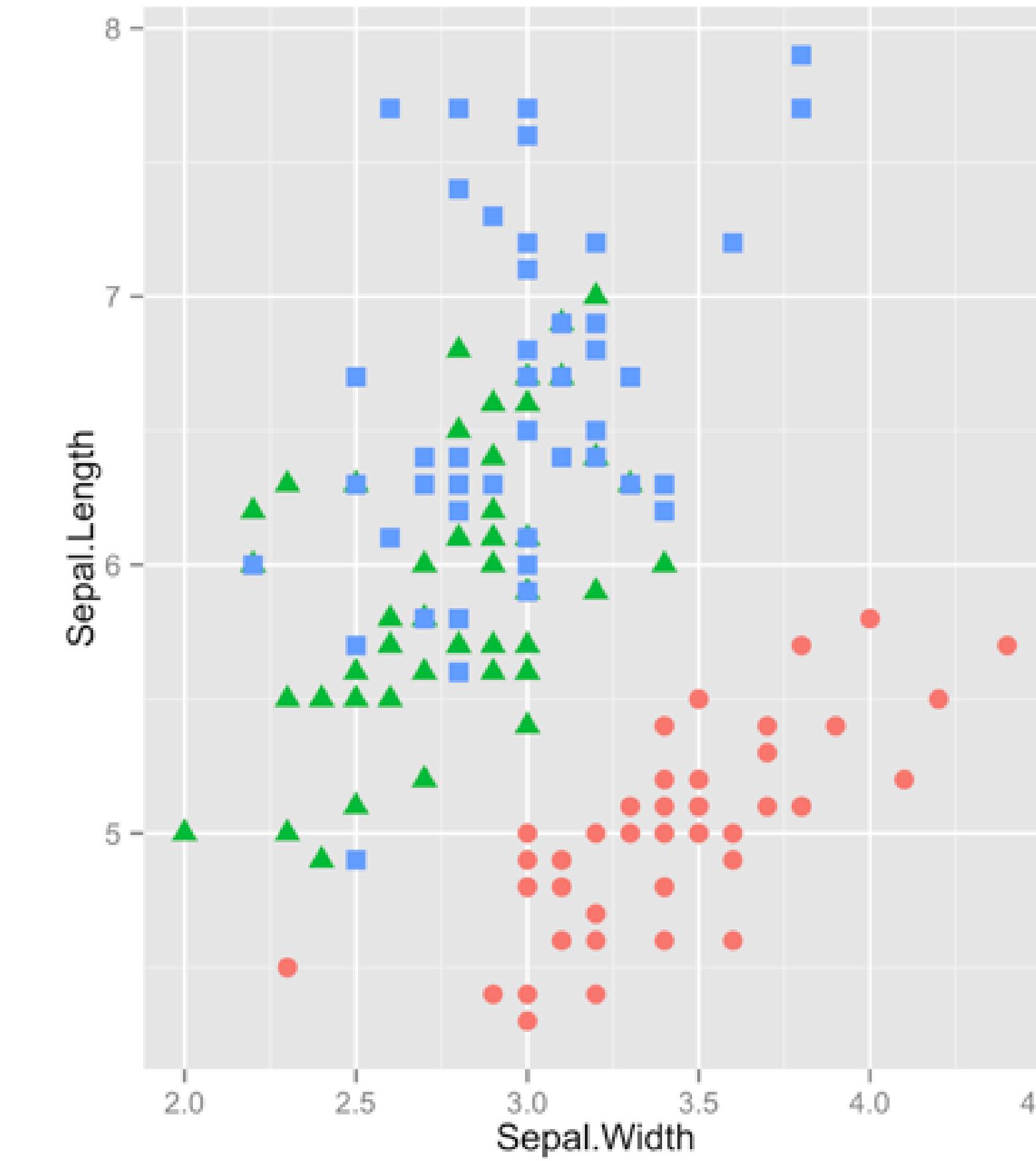
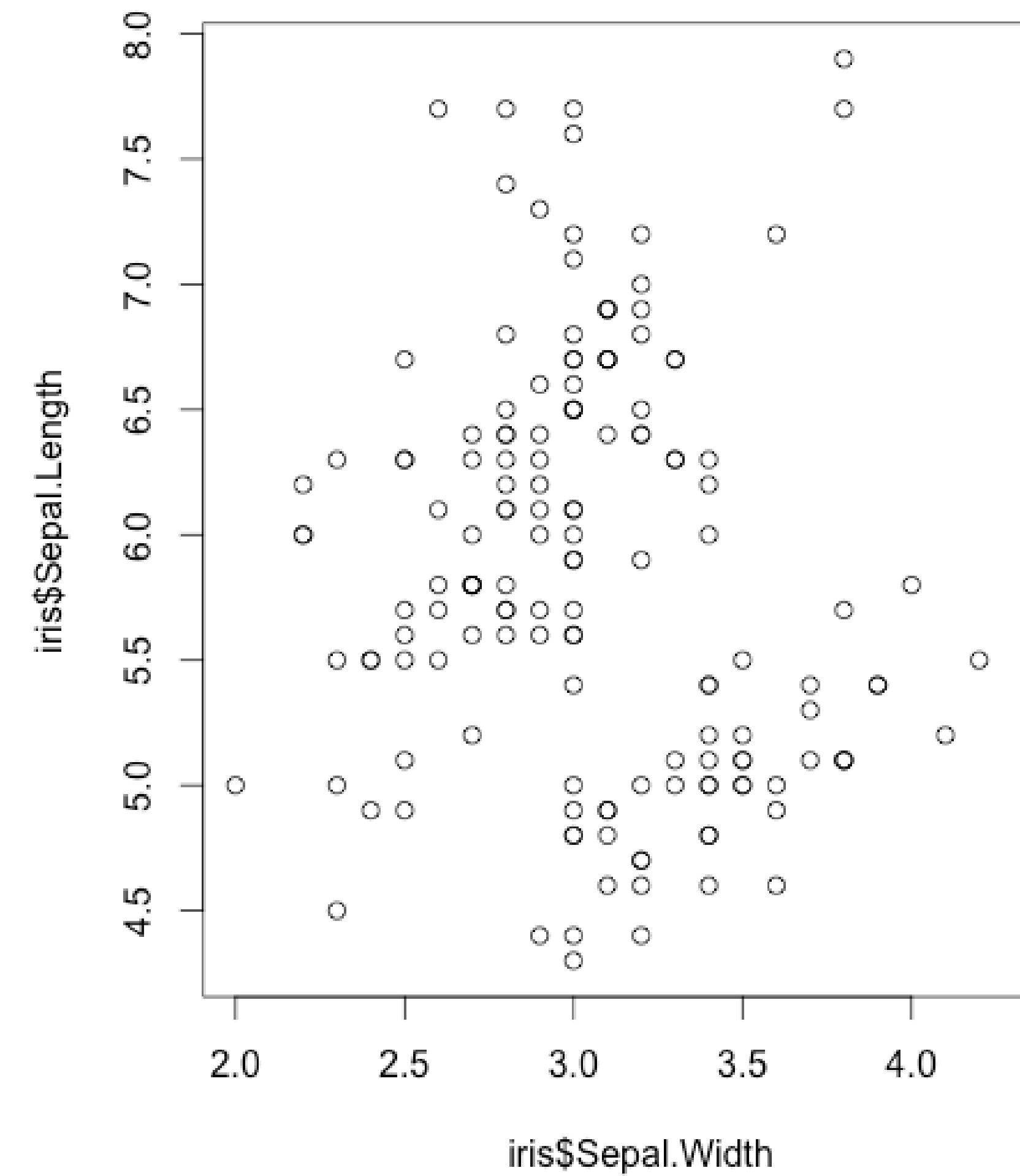
- Consistent underlying - Grammar of Graphics (Wilkinson, 2005)
- Very flexible
- Mature and complete graphics system
- Many users, active mailing list
- <http://www.cookbook-r.com> & <http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html>



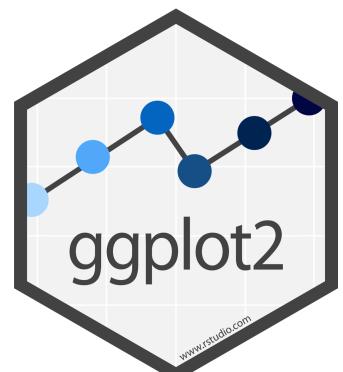
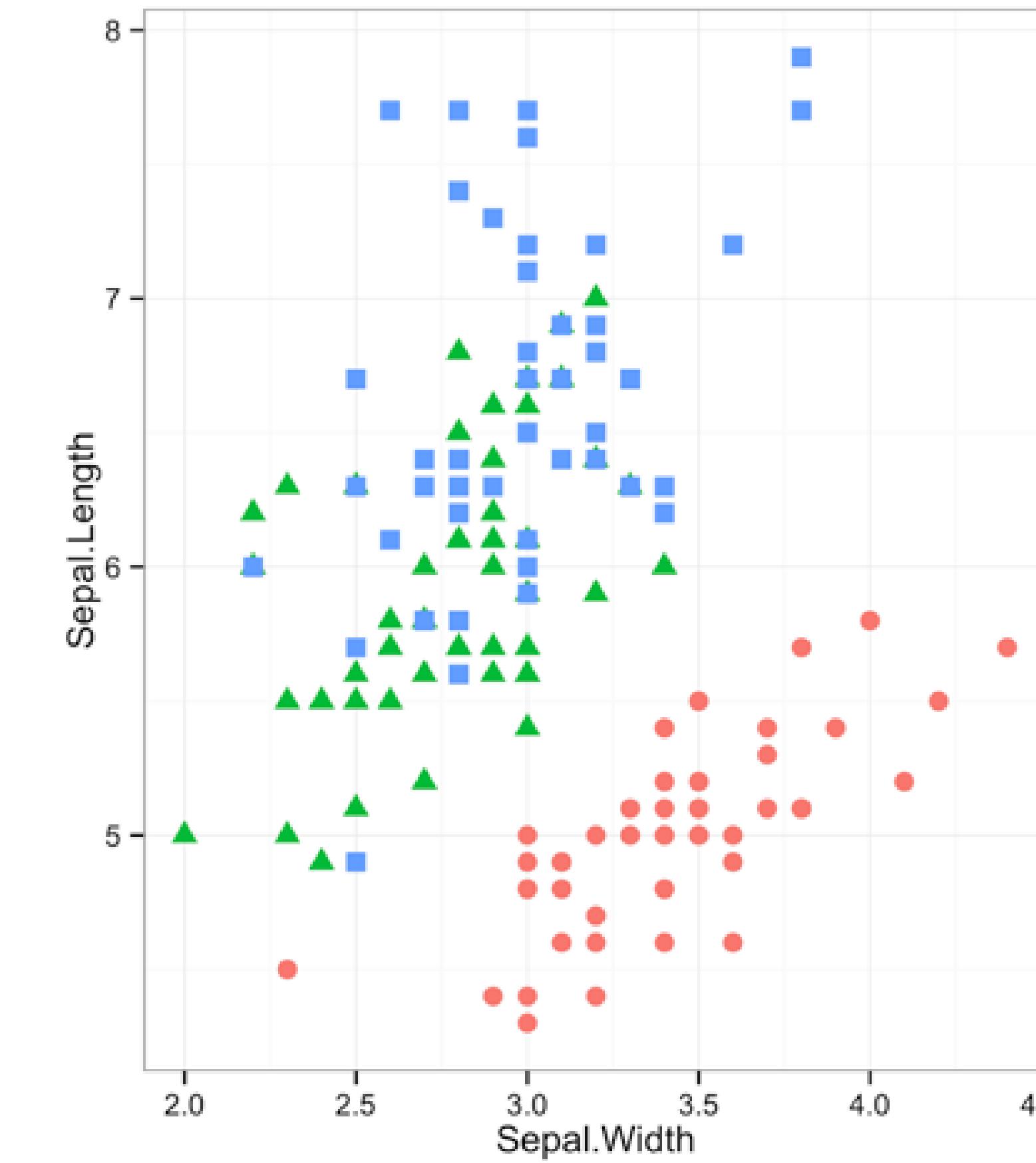
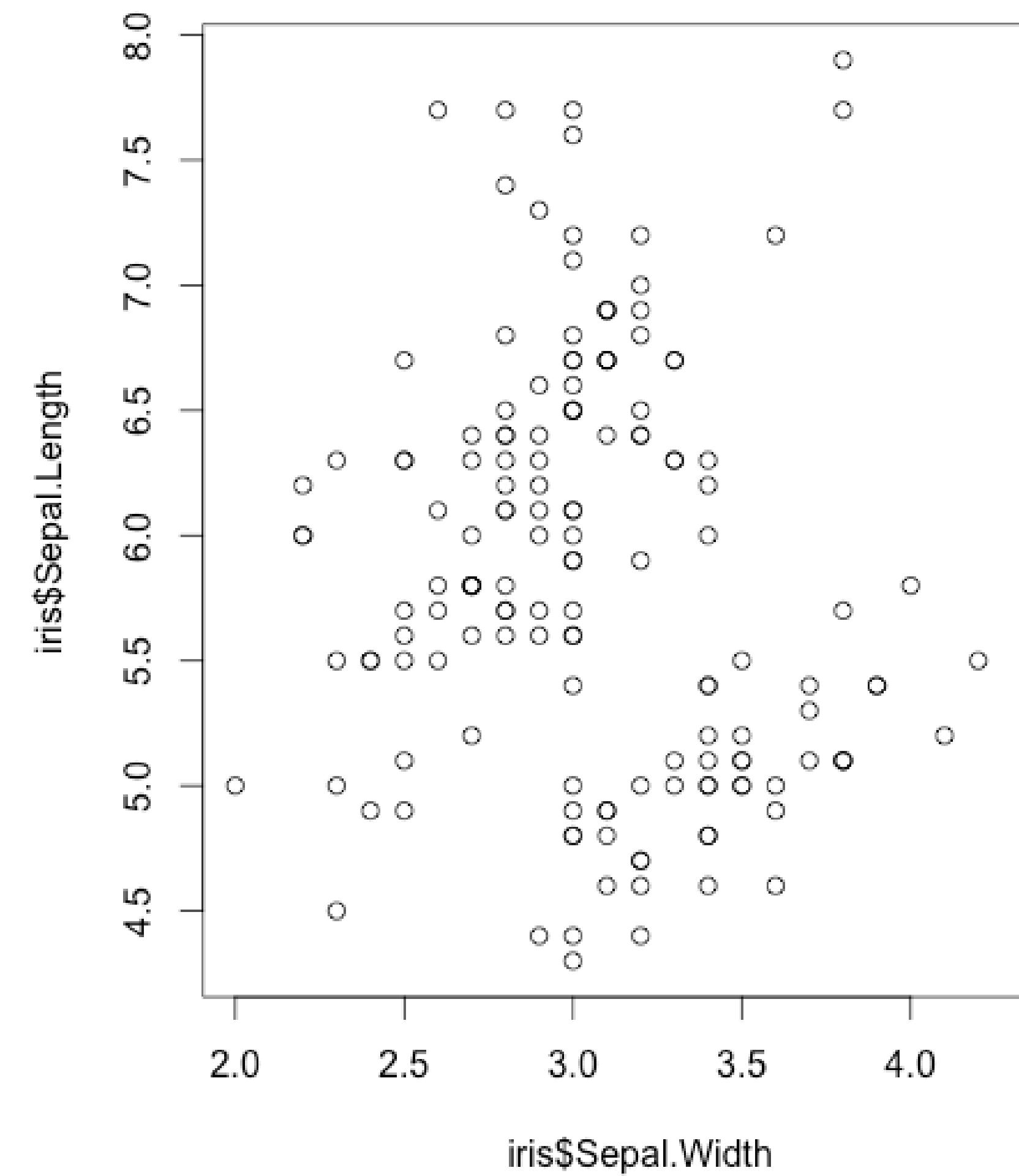
ggplot2



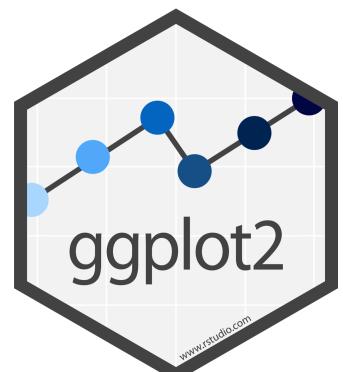
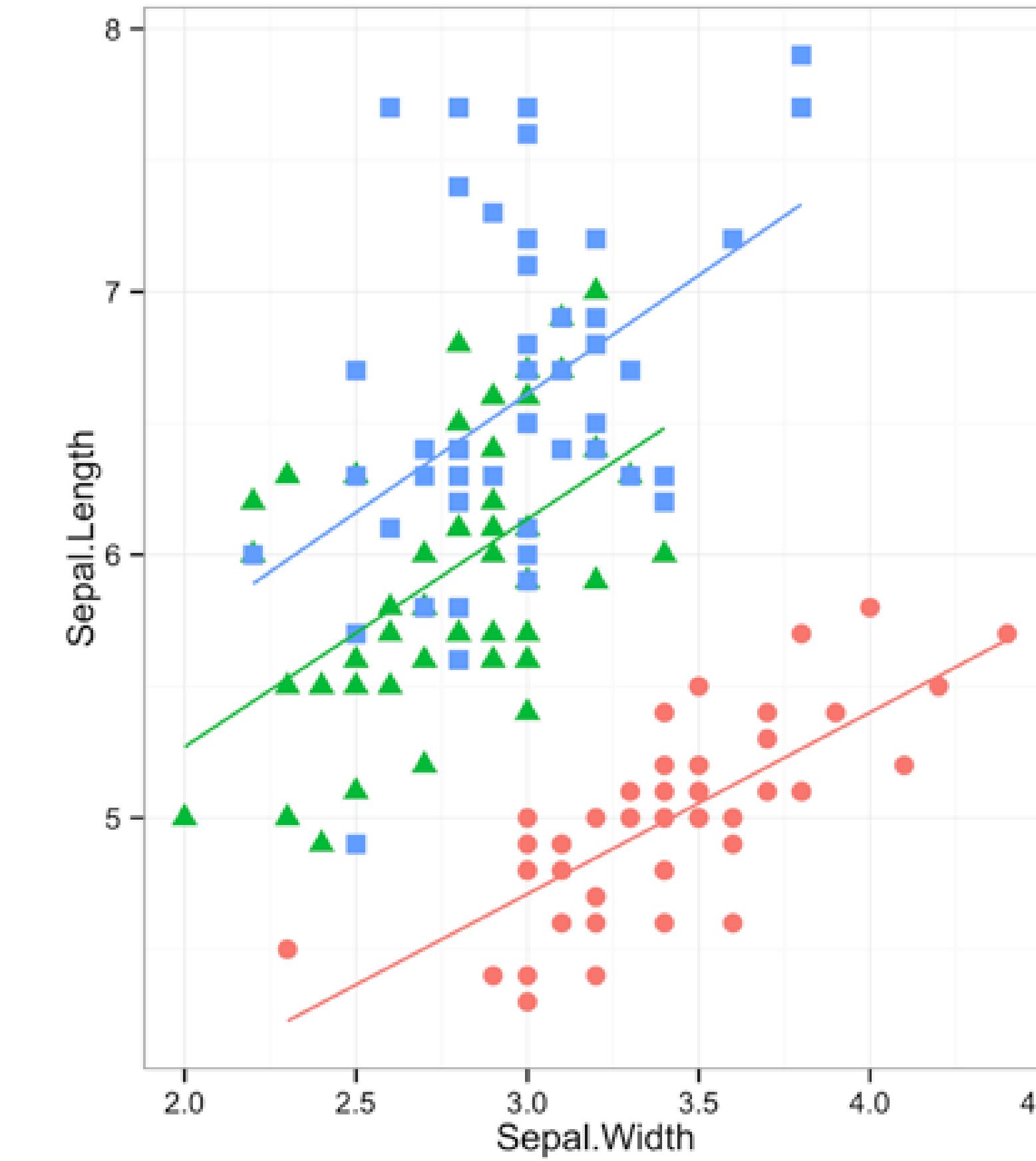
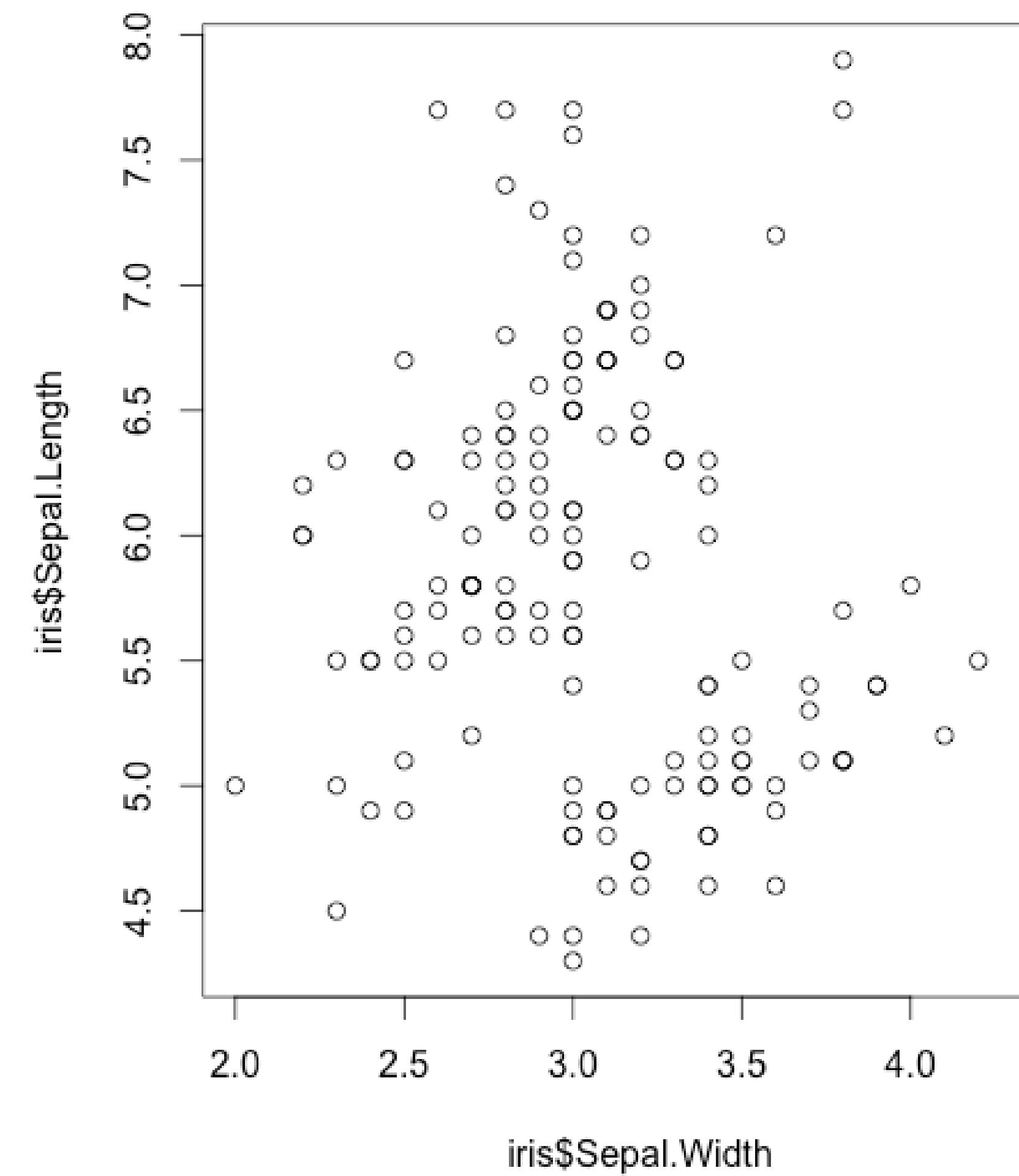
ggplot2



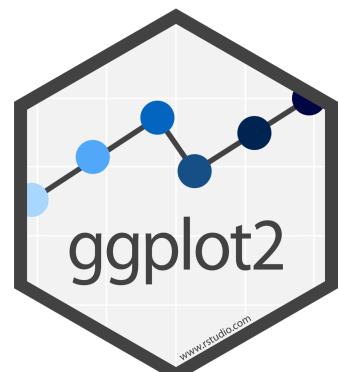
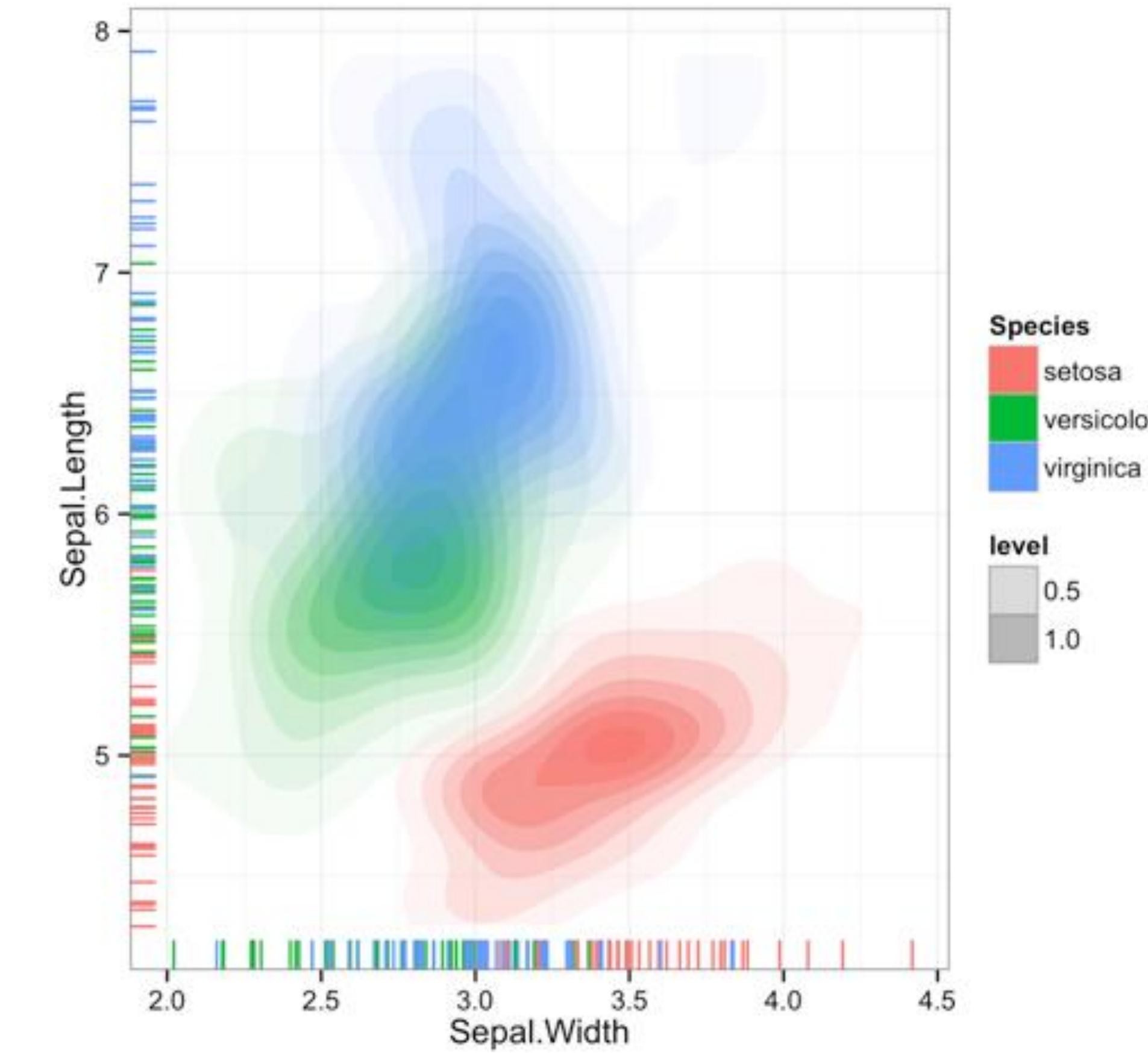
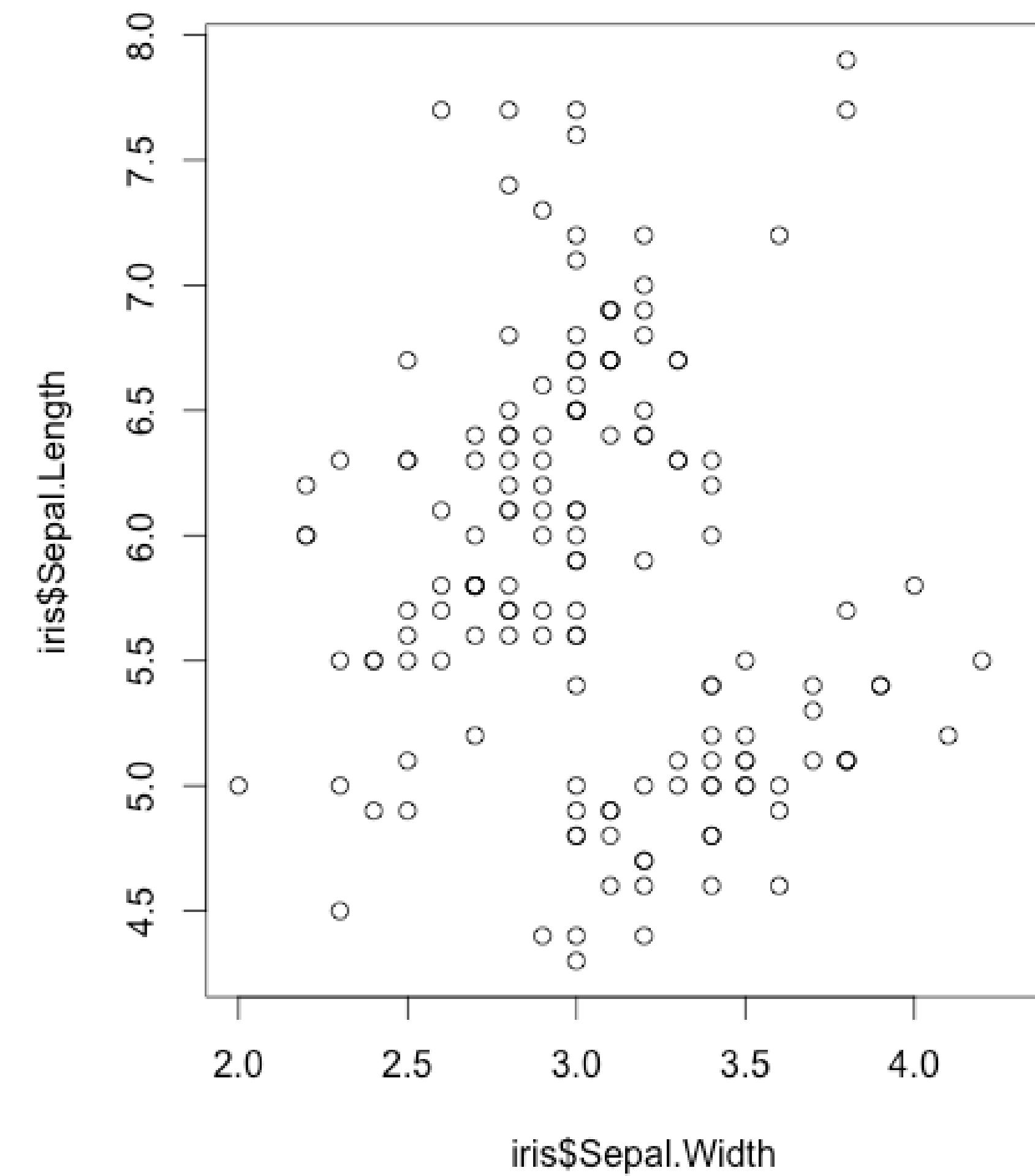
ggplot2



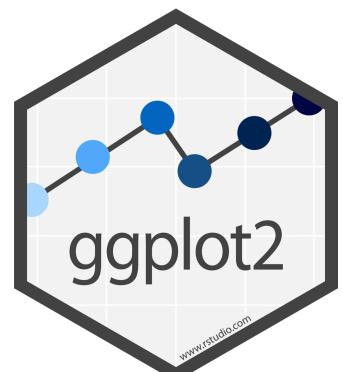
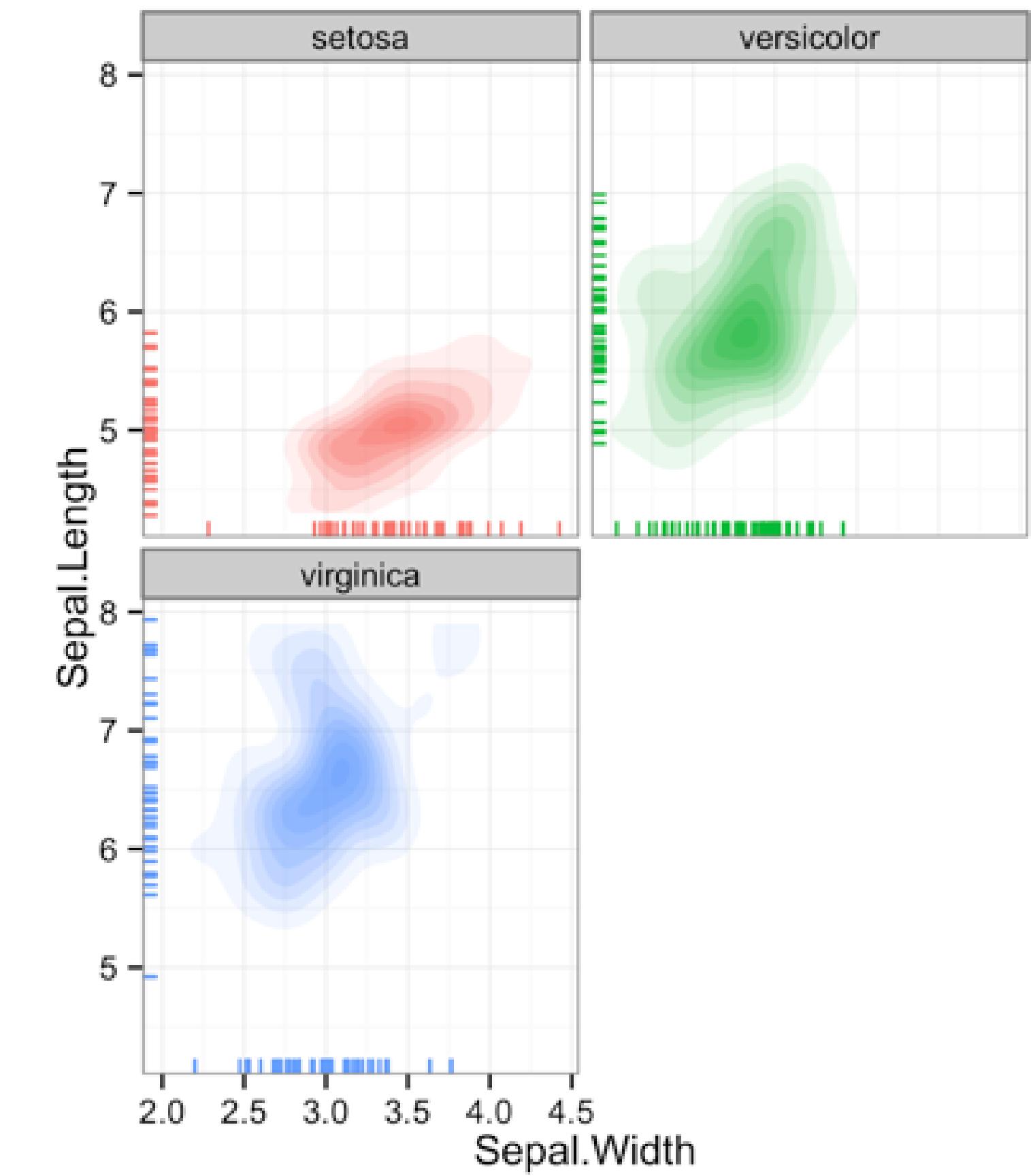
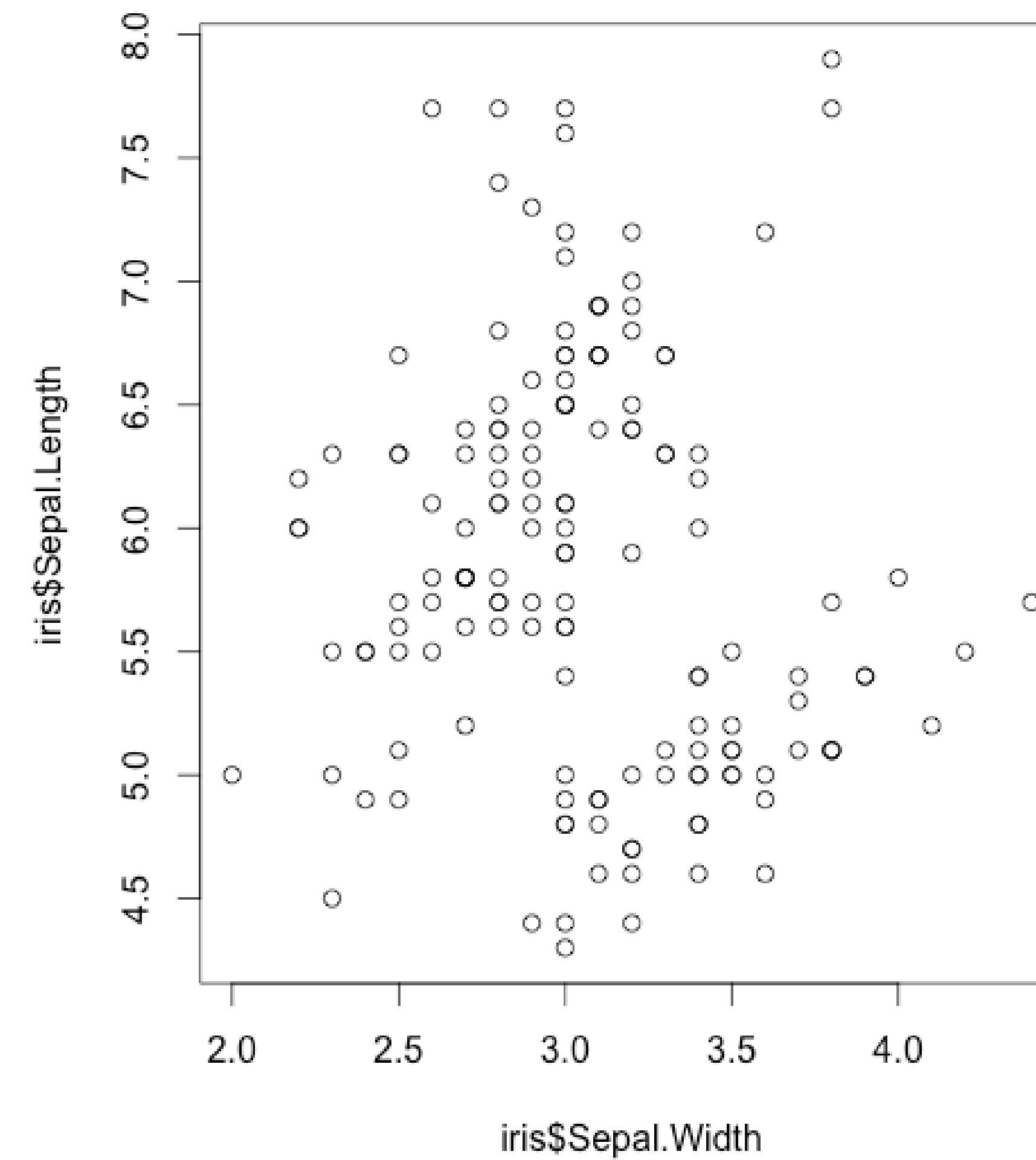
ggplot2

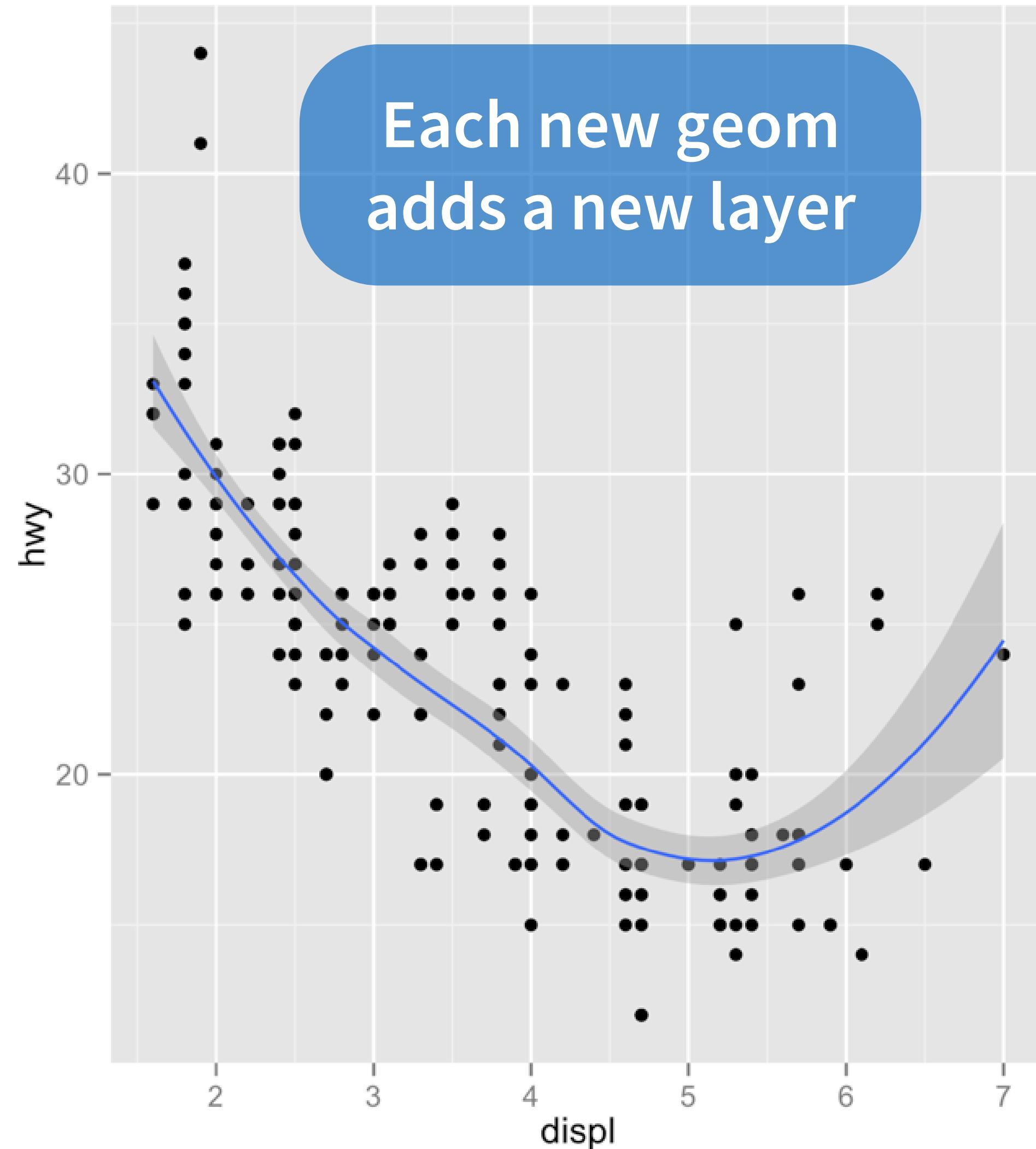


ggplot2



ggplot2





```
ggplot(mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy)) +  
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

A ggplot2 template

Make any plot by filling in the parameters of this template

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),  
    stat = <STAT>) +  
  <FACET_FUNCTION>
```

```
ggplot(data = diamonds) +  
  geom_bar(mapping = aes(x = cut), stat = "count")
```



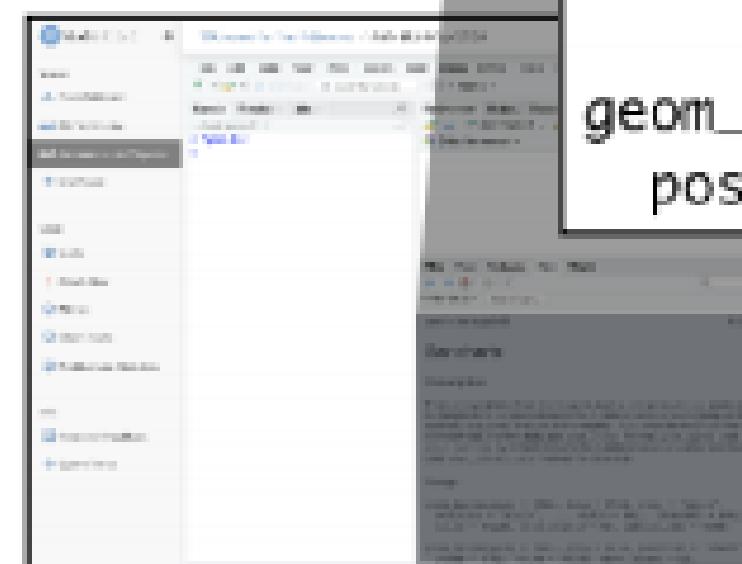
"Help" pages

To open the documentation for a function, type

```
?geom_histogram
```

?

function name (no parentheses)



geom_freqpoly (ggplot2)

R Documentation

Histograms and frequency polygons

Description

Visualise the distribution of a single continuous variable by dividing the x axis into bins and counting the number of observations in each bin. Histograms (`geom_histogram()`) display the counts with bars; frequency polygons (`geom_freqpoly()`) display the counts with lines. Frequency polygons are more suitable when you want to compare the distribution across the levels of a categorical variable.

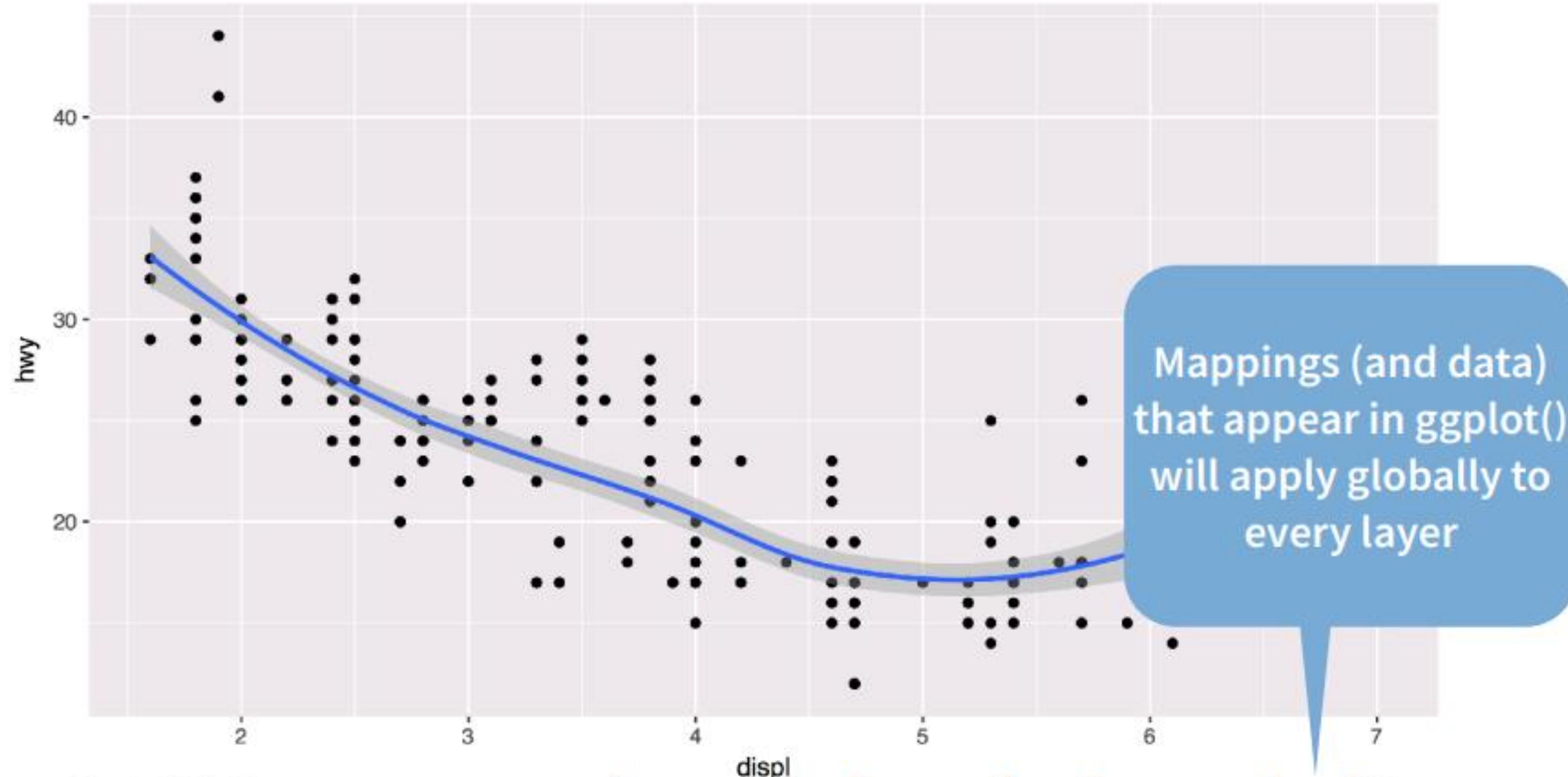
Usage

```
geom_freqpoly(mapping = NULL, data = NULL, stat = "bin",
  position = "identity", ..., na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE)

geom_histogram(mapping = NULL, data = NULL, stat = "bin",
  position = "stack", ..., binwidth = NULL, bins = NULL,
```

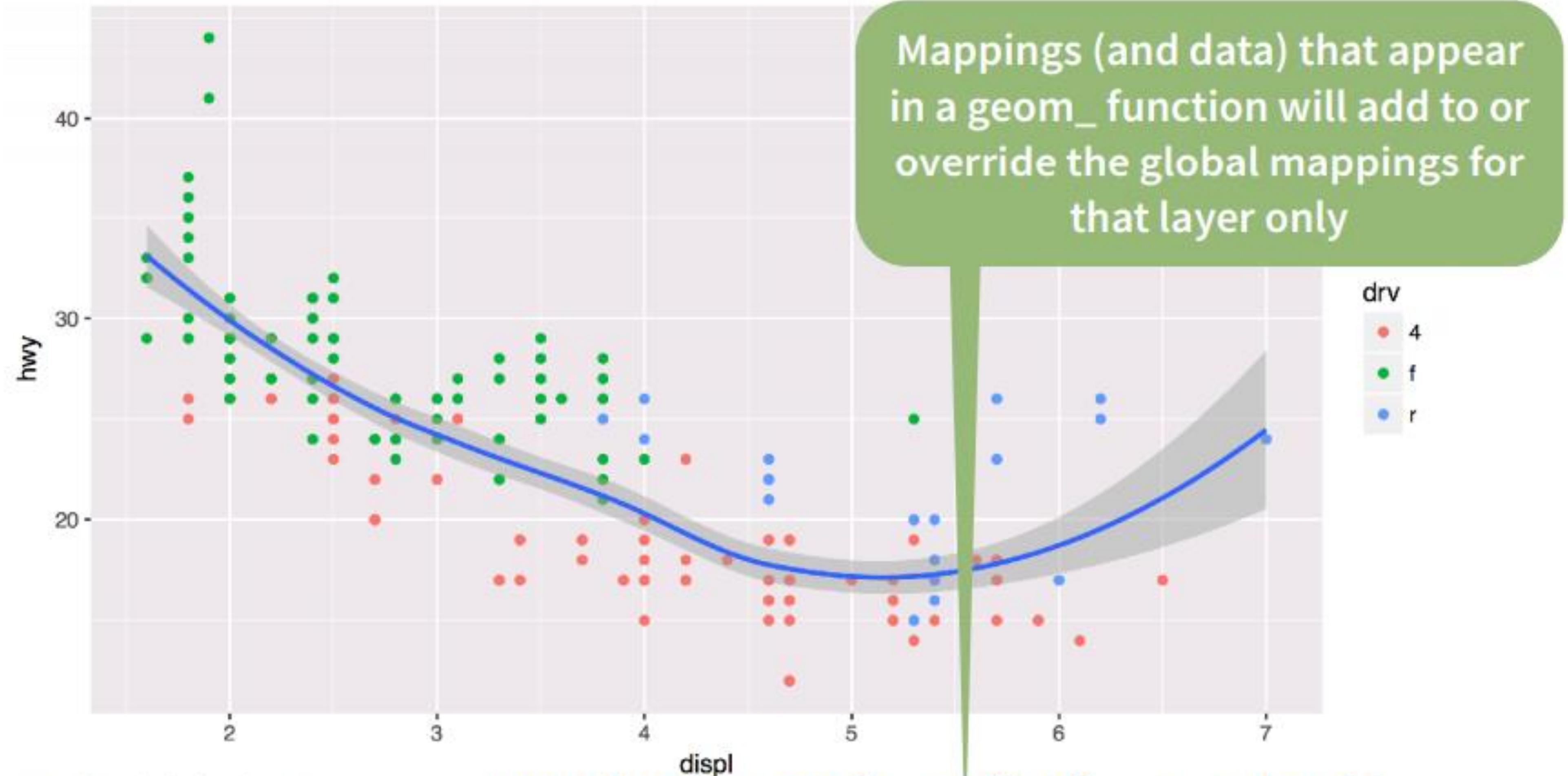


global vs. local



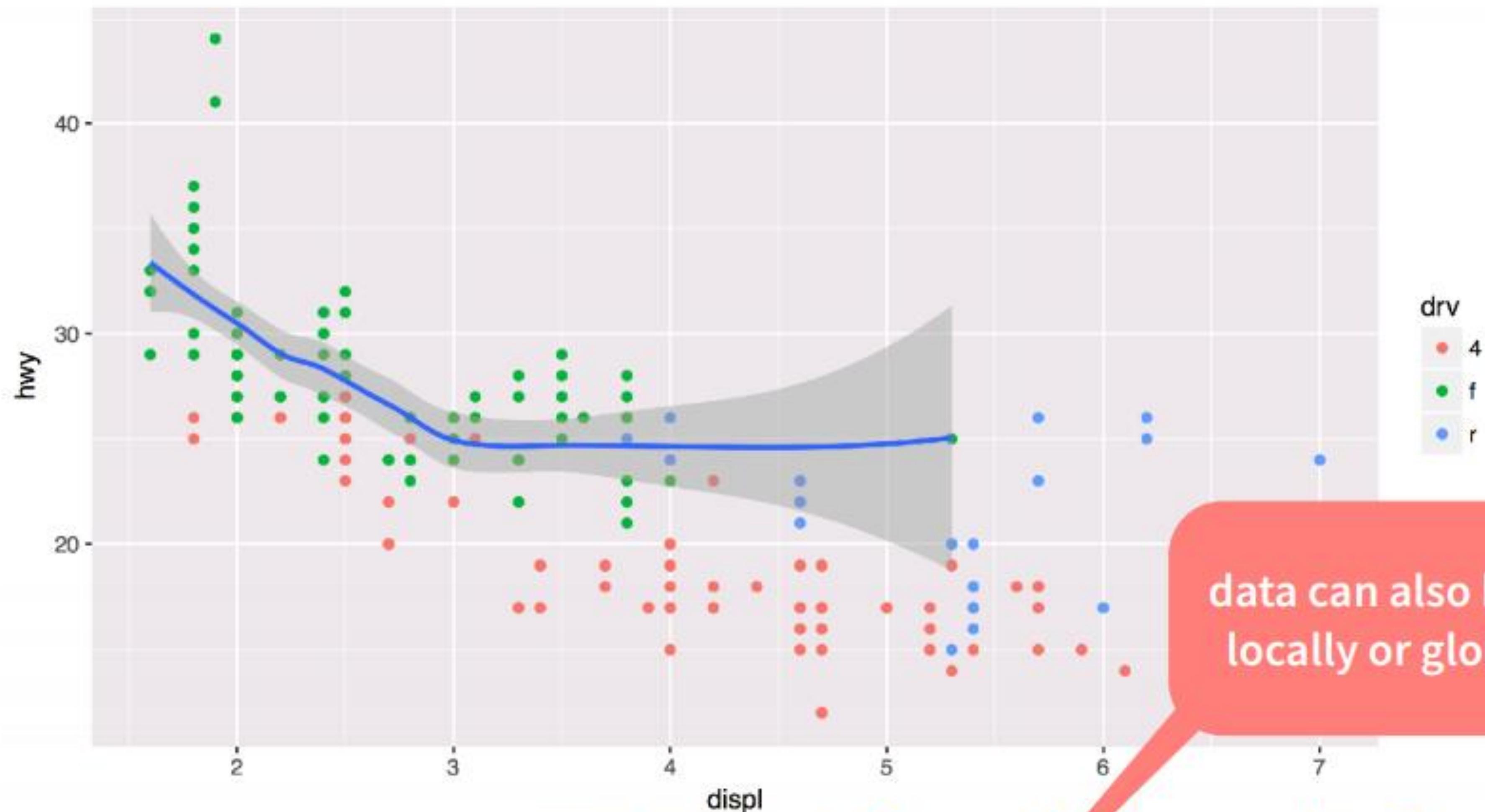
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +  
  geom_point() +  
  geom_smooth()
```





```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +  
  geom_point(mapping = aes(color = drv)) +  
  geom_smooth()
```





```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +  
  geom_point(mapping = aes(color = drv)) +  
  geom_smooth(data = filter(mpg, drv == "f"))
```



data can also be set
locally or globally

A ggplot2 template

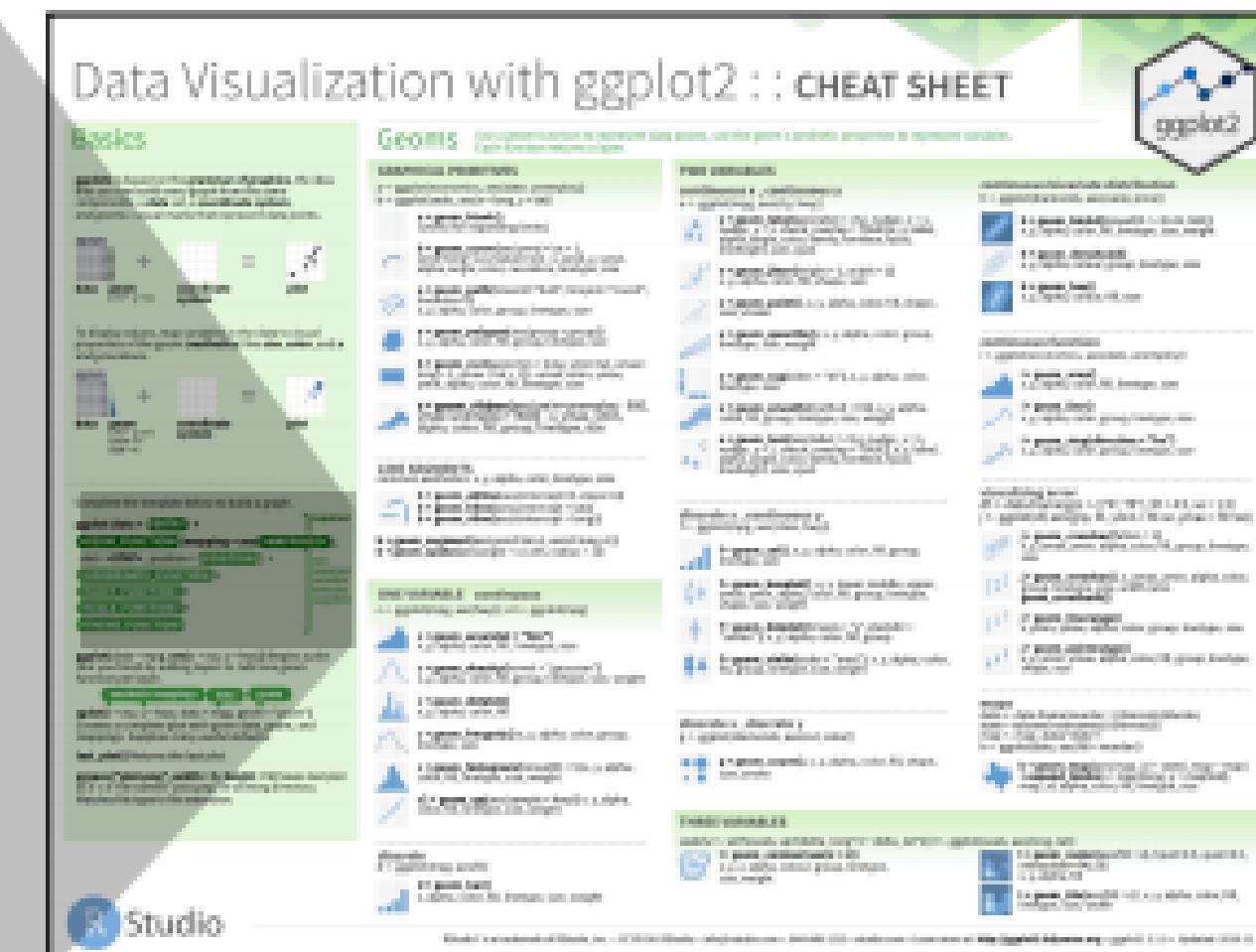
Make any plot by filling in the parameters of this template

Complete the template below to build a graph.

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

required

Not required,
sensible
defaults
supplied



• Addins ▾

- Insert Image
- Touch File
- Quote Poem
- BOOKDOWN

- Preview Book
- Input LaTeX Math
- CLIPR

- Value to clipboard
- Output to clipboard
- DEVTOOLS

- Run a test file
- Report test coverage for a file
- Report test coverage for a package

- ESQUISSE

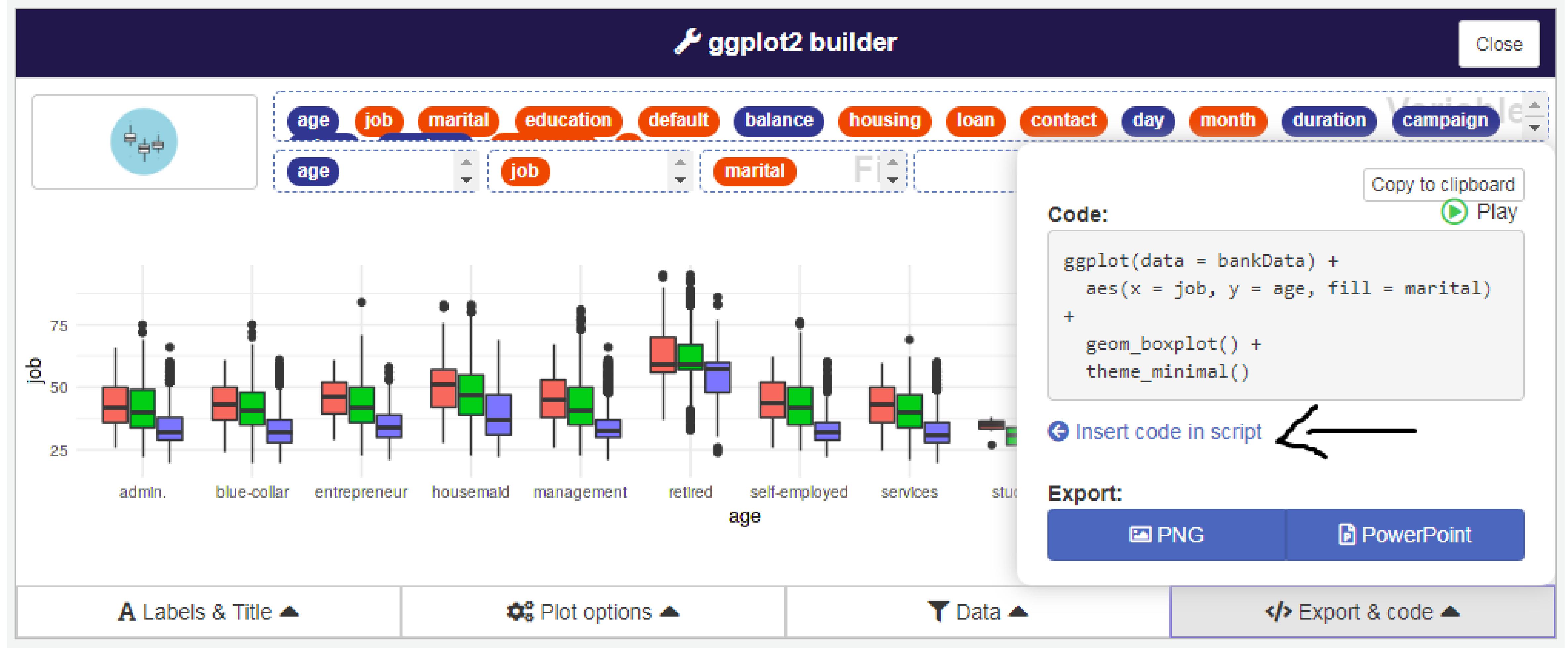
- 'ggplot2' builder
- Export 'ggplot2' plots to 'PowerPoint'
- Explore your data with 'ggplot2'.
- GGTHEMEASSIST

- ggplot Theme Assistant
- PKGDOWN

- Build pkgdown



Dessine-moi un mouton...



EXERCISE 7MINS:

- Import some data and build some visualizations using esquisse...Take your data import code and visualization code and insert it into a notebook. Pick any dataset below to work on.
- landdata-states.csv
- starbucks.csv
- adae.csv
- dm.csv
- bank-full.csv
- ad_treatment.xlsx
- dmae.sas7bdat

htmlwidgets

<http://www.wthr.com/article/watch-video-shows-tornado-destroying-kokomo-starbucks>

Demo Example

- 1_htmlwidgets_tornadoes.R
- Compare R script vs Notebook – what are the differences?

Plotly

Tools for making interactive plots. plot.ly/ggplot2/

The screenshot shows a web browser window with the title "ggplot2 Graphing Library | Plotly". The address bar displays "plot.ly/ggplot2/". The main content area is titled "Plotly ggplot2 Library". It features a sidebar with links like "Quick Start", "Getting Started", "User Guide", "Examples" (which is currently selected), "Basic", "Statistical", "Animations", "Layout Options", and "Community". A search bar at the top right contains the placeholder "Search Plotly's R & ggplot2 D". Below the search bar, there is a section titled "Basic Charts" with a small icon of a line graph.



htmlwidgets for R:

- R bindings to JavaScript libraries
- Used to create interactive visualizations
- A line or two of R code is all it takes to produce an example

Use htmlwidgets in:

- RStudio viewer pane
- R Markdown files
- Shiny Apps

www.htmlwidgets.org

htmlwidgets

The screenshot shows the official website for htmlwidgets. At the top, there's a navigation bar with links for Home, Showcase, Develop, and GitHub. Below the navigation, a main heading reads "htmlwidgets for R". To the right, there's a screenshot of a Shiny application titled "Oregon Climate Station Data". The Shiny app interface includes dropdown menus for "Color counties by" (set to "(None)"), "Color markers by" (set to "Annual temperature"), and "Size markers by" (set to "Rainfall"). The main area of the Shiny app displays a map of Oregon with numerous red circles of varying sizes scattered across it, representing climate station data. Below the Shiny app screenshot, there are three smaller thumbnail images labeled "Widgets in action" showing examples of Leaflet maps, dygraphs time-series plots, and networkD3 network visualizations.

htmlwidgets for R

Home Showcase Develop GitHub

Bring the best of JavaScript data visualization to R

Use JavaScript visualization libraries at the R console, just like plots

Embed widgets in R Markdown documents and Shiny web applications

Develop new widgets using a framework that seamlessly bridges R and JavaScript

At the R console In R Markdown docs In Shiny apps

Widgets in action

See how just a line or two of R code can be used to create interactive visualizations with Leaflet (mapping), dygraphs (time-series), networkD3 (graph visualization), and more.

javascript:void

See the showcase »

htmlwidgets gallery

<http://gallery.htmlwidgets.org/>

The screenshot shows the main page of the htmlwidgets gallery. At the top, there are search and filter controls: Sort (set to "Github stars"), Text Filter (empty), Author Filter (empty), Tag Filter (empty), and CRAN Only (checked). Below these, a message says "55 registered widgets available to explore".

Three examples are displayed:

- DiagrammeR**: Shows a graph diagram with several nodes (green and orange) connected by arrows.
- leaflet**: Shows a map of Seattle, Washington, with various locations labeled.
- networkD3**: Shows a complex flow diagram illustrating energy and material flows through a system, with components like Nuclear, Oil imports, Thermal generation, and District heating.

Each example includes a star icon and a count (e.g., 366 for DiagrammeR, 141 for leaflet, 125 for networkD3). Below each example is a brief description and a list of author, tag, and library details.

DiagrammeR (366)
With DiagrammeR, you can easily create graph diagrams using R.
▪ author: rich-iannone
▪ tags: visualization, diagram
▪ is libraries: d3.viz.mermaid

leaflet (141)
Leaflet is an open-source JavaScript library for interactive maps. This R package makes it easy to create Leaflet maps from R.
▪ author: rstudio
▪ tags: visualization, maps

networkD3 (125)
A port of Christopher Gandrud's d3Network package to the htmlwidgets framework.
▪ author: christophergandrud
▪ tags: visualization, networks
▪ is libraries: d3

EXERCISE 10MINS:

- 02-Visualize-Exercises.Rmd Beginner
- Run through the chunks
- 2_r4ds_ggplot2_tidyverse.Rmd More Advanced

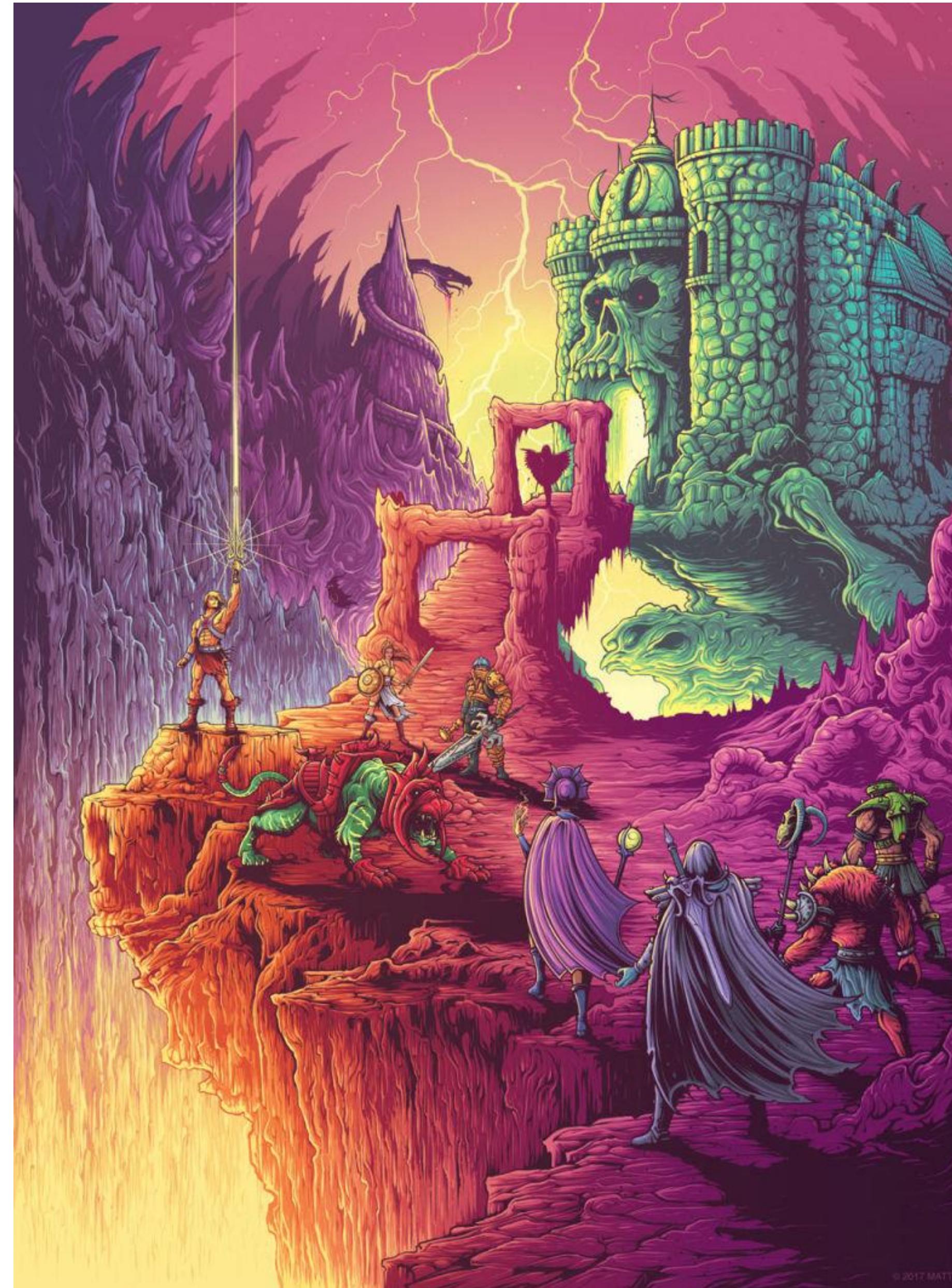
BREAK TIME

- Fun Video

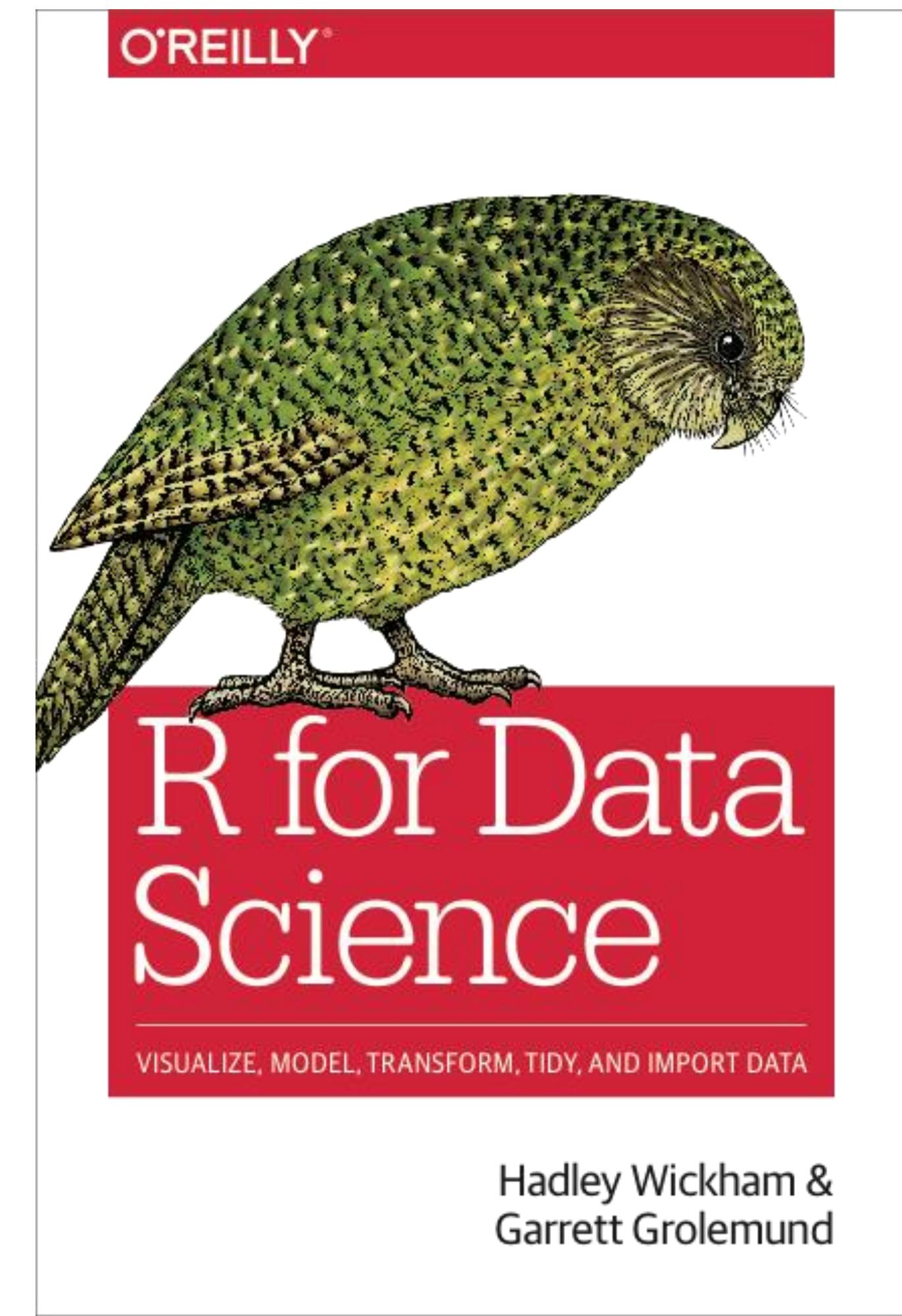


R

Masters of the Tidyverse



Art by Dan Mumford



<https://github.com/rstudio/RStartHere>

THE TIDYVERSE WORKS WITH TIBBLES



I SAID TIBBLES,
NOT TRIBBLES
**...Darn
Quadrotriticale!**

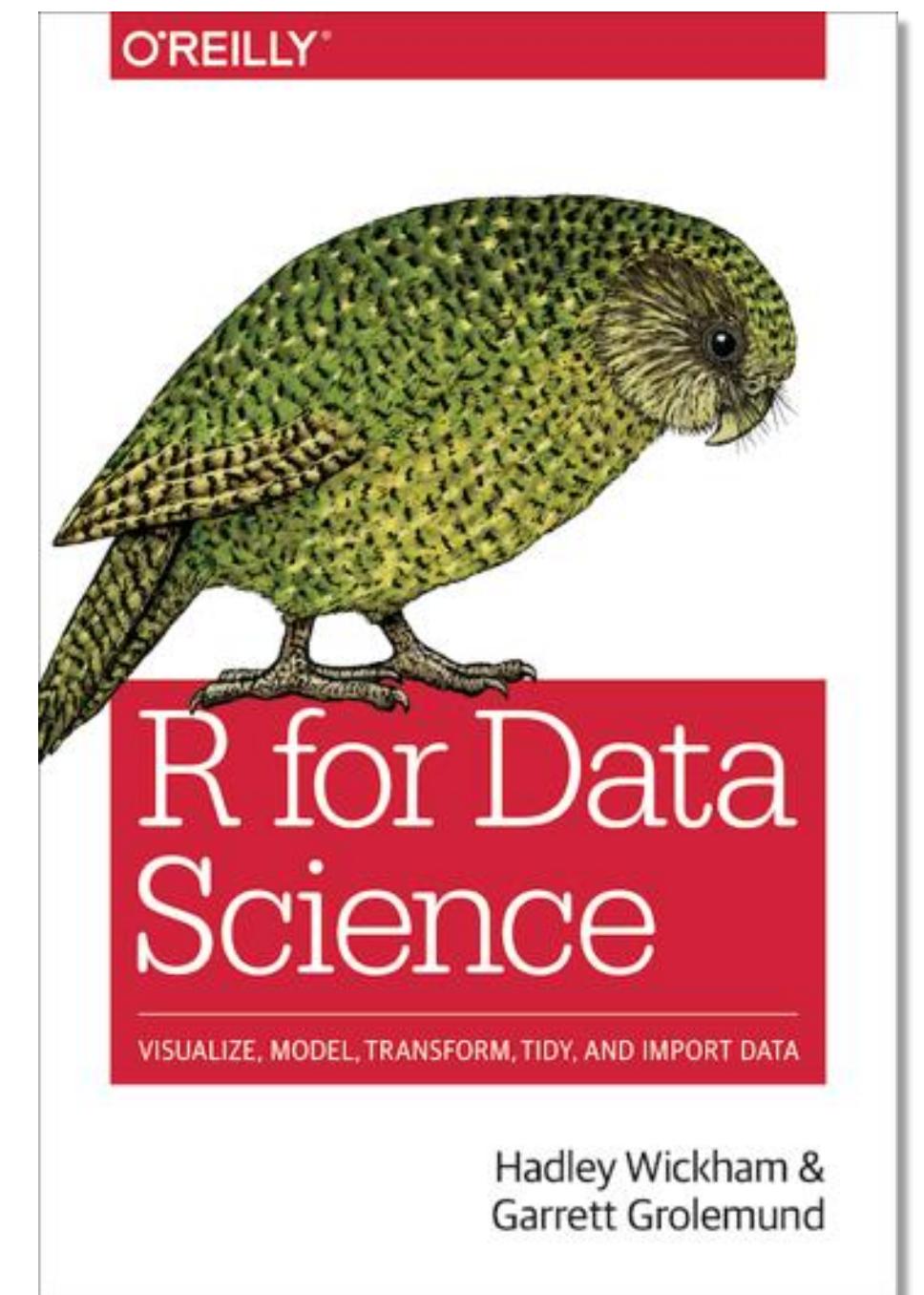
Tidyverse

A collection of R packages that share common philosophies and are designed to work together.



tidyverse.org

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



DATA TYPES

- R has a wide variety of data types...
- Vectors
- Lists
- **Matrix**
- **Factors**
- **Data frame**
- **Tibble**
- Is ToothGrowth a Tibble? Hint: class(ToothGrowth)

Toy data

```
storms <- tribble(  
  ~storm, ~wind, ~pressure, ~date,  
  "Alberto", 110, 1007, "2000-08-12",  
  "Alex", 45, 1009, "1998-07-30",  
  "Allison", 65, 1005, "1995-06-04",  
  "Ana", 40, 1013, "1997-07-01",  
  "Arlene", 50, 1010, "1999-06-13",  
  "Arthur", 45, 1010, "1996-06-21"  
)
```

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

TIBBLES – QUICK INTRO

- `as_tibble()`
- `as_tibble(ToothGrowth)`
- This will work for reasonable inputs that are already `data.frames`, lists, matrices, or tables.
- There are two main differences in the usage of a tibble vs. a classic `data.frame`: printing and subsetting
- Tibbles show only the first 10 rows
- Each column reports its type, a nice feature borrowed from `str()`
- `package?tibble`

TIBBLES – WHAT TO KNOW

- tibble() does much less than data.frame():
- A. it never changes the type of the inputs (e.g. it never converts strings to factors!)
- B. it never changes the names of variables, and it never creates row names
- Cdata.frame() vs. modern data_frame():
- Base R has a burning desire to turn character information into factor...via read.table() & data.frame() and other functions are also eager
 - To shut this down, use stringsAsFactors = FALSE in read.table() and data.frame() or
 - even better – use the tidyverse!
- readr::read_csv(), readr::read_tsv(), etc. For data frame creation, use tibble::tibble()

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

does the equivalent of

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

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

does the equivalent of

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

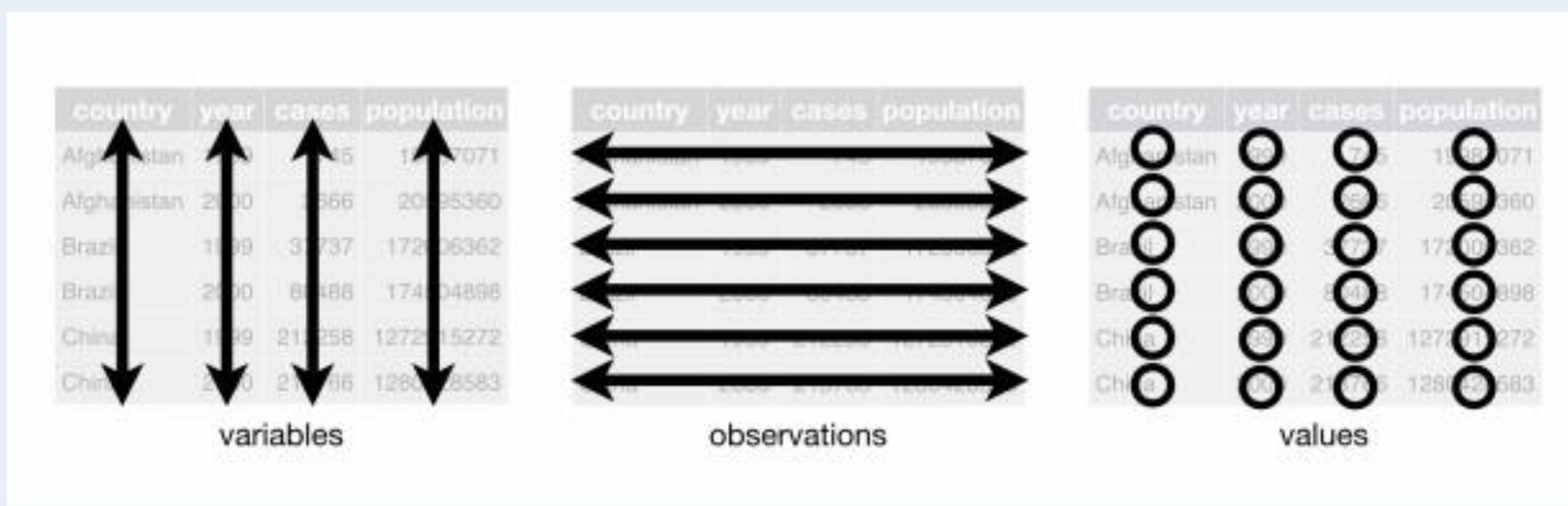
```
library("tidyverse")
```

does the equivalent of

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

tidyverse Core Principles

- Built around data - usually as a `data.frame` or `tibble`
- Built around tidy data
 - Each variable in its own column
 - Each observation or case in its own row
 - Each type of observational units forms a table



babynames



Names of male and female babies born in the US from 1880 to 2008. 1.8M rows.

```
# install.packages("babynames")  
library(babynames)
```



View(babynames)

	year	sex	name	n	prop
1	1880	F	Mary	7065	0.0723835869
2	1880	F	Anna	2604	0.0266789611
3	1880	F	Emma	2003	0.0205214897
4	1880	F	Elizabeth	1939	0.0198657856
5	1880	F	Minnie	1746	0.0178884278
6	1880	F	Margaret	1578	0.0161672045
7	1880	F	Ida	1472	0.0150811946
8	1880	F	Alice	1414	0.0144869628
9	1880	F	Bertha	1320	0.0135238973
10	1880	F	Sarah	1288	0.0131960453
11	1880	F	Annie	1258	0.0128886840



WE CALL THIS DATA **TIDY**

Data are organized in **rows** and **columns**

Each **variable** has its own column

Each **observation** or record has its own row

Every cell has only one value in it

Name	Address	City	State

The diagram shows a 5x4 grid representing a dataset. The columns are labeled "Name", "Address", "City", and "State". Four orange arrows point from the bottom of each row to their respective column headers, illustrating that each row represents an observation and each column represents a variable.

Untidy data

```
untidy_df
```

```
## # A tibble: 5 x 7
##   age_group male_2016 female_2016 male_2017 female_2017 male_2018
##   <chr>       <dbl>      <dbl>       <dbl>      <dbl>       <dbl>
## 1 < 18        22000     20000      22000     20000      22000
## 2 18-30       36000     35000      36000     35000      36000
## 3 31-50       50000     40000      50000     40000      50000
## 4 51-60       62000     60000      62000     60000      62000
## 5 > 60        75000     72000      75000     72000      75000
## # ... with 1 more variable: female_2018 <dbl>
```

Tidy data

```
tidy_df
```

```
## # A tibble: 30 x 4
##   age_group gender year  income
##   <chr>     <chr>  <chr> <dbl>
## 1 < 18      male   2016  22000
## 2 18-30     male   2016  36000
## 3 31-50     male   2016  50000
## 4 51-60     male   2016  62000
## 5 > 60      male   2016  75000
## 6 < 18      female 2016  20000
## 7 18-30     female 2016  35000
## 8 31-50     female 2016  40000
## 9 51-60     female 2016  60000
## 10 > 60     female 2016  72000
## # ... with 20 more rows
```

BREAK TIME

- Fun Video

Transforming Data & Data Visualization



Art by Lou Pimentel

dplyr



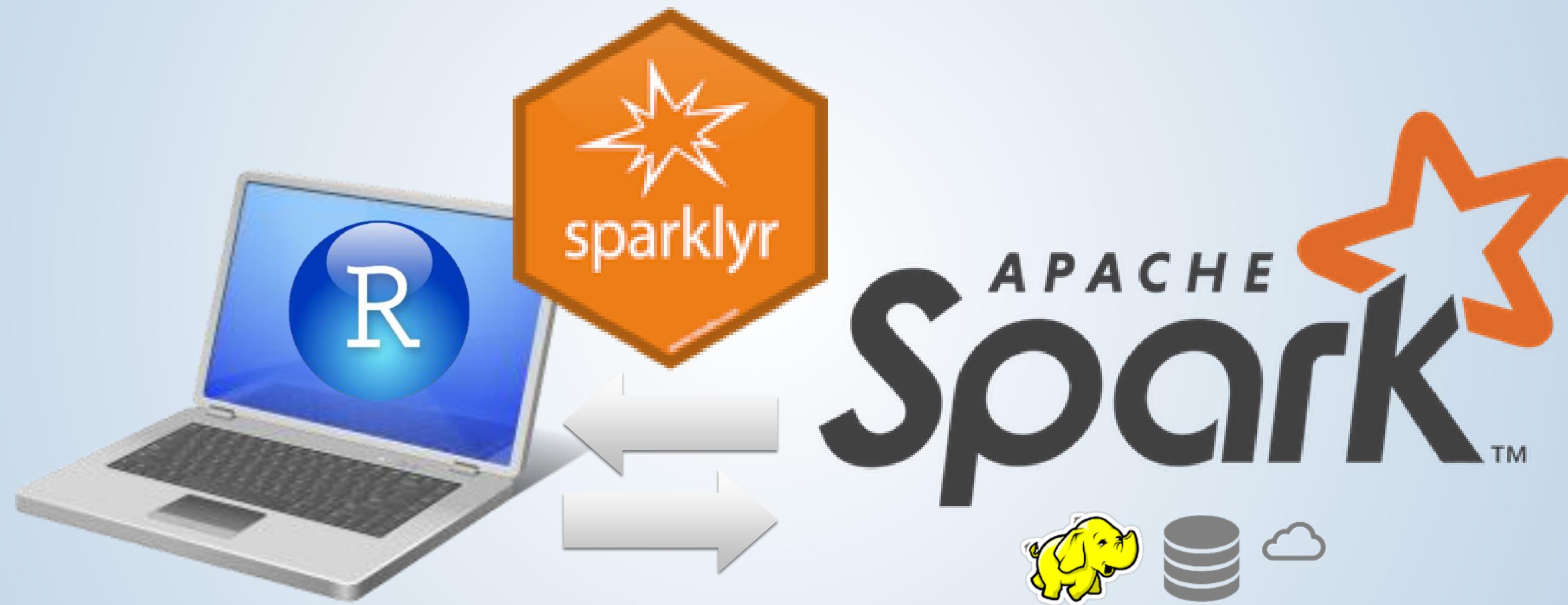
A package that transforms data.
dplyr implements a *grammar* for
transforming tabular data.

Data transformation toolbox



Sparklyr

Makes it easy to use R with Spark



[Spark with AWS EMR](#)
[Spark and the data lake](#)

dplyr

6 Main verbs

- `filter()`
- `arrange()`
- `select()`
- `mutate()`
- `group_by()`
- `summarise()`

Simple use

- `pull()`
- `n()/count()`
- `glimpse()`

Advanced iterations

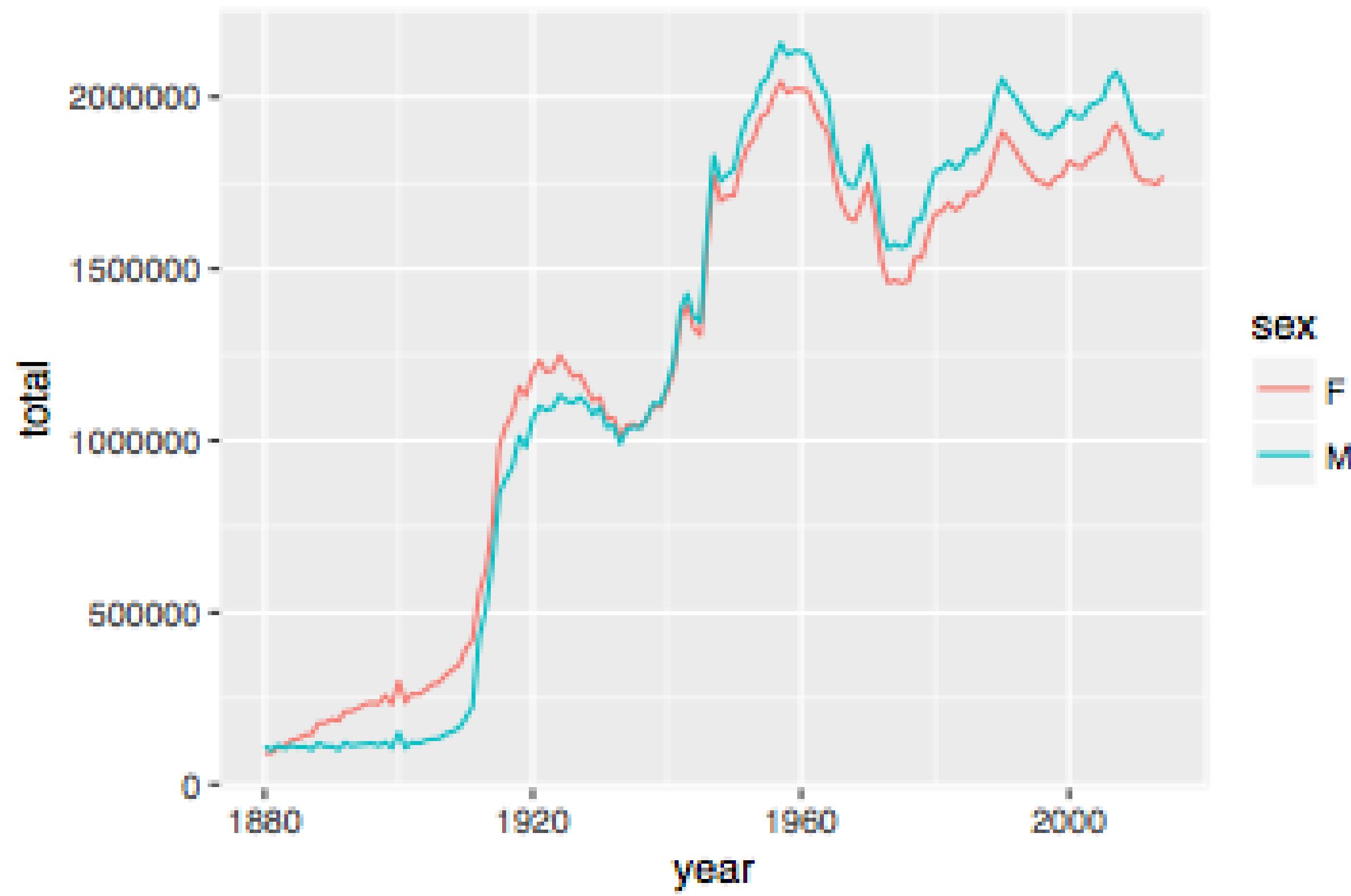
- `summarize_at()`
- `mutate_at()`

More info

- dplyr.tidyverse.org
- R for Data Science



```
babynames %>%  
  group_by(year, sex) %>%  
  summarise(total = sum(n)) %>%  
  ggplot(aes(x = year, y = total, color = sex)) +  
  geom_line()
```



%>% IS CALLED THE PIPE OPERATOR

- Connects the output of one operation to the input of the next.
- Allows us to see the flow of data from left to right

```
employees %>%  
  filter(Department == "Engineering") %>%  
  count()
```

```
# A tibble: 1 × 1
```

```
  n  
  <int>  
1    71
```



The %>% == and then

Rather than multiple assignment or nesting functions

```
did_something <- do_something(data)

did_another_thing <- do_another_thing(did_something)

final_thing <- do_last_thing(did_another_thing)
```

```
final_thing <- do_last_thing(
  do_another_thing(
    do_something(
      data
    )
  )
)
```

```
final_thing <- data %>%
  do_something() %>%
  do_another_thing() %>%
  do_last_thing()
```

```
Phil <- filter(babynames, name == "Phil", sex == "M")
summarise(Phil, min = min(prop), mean = mean(prop),
max = max(prop))
```

```
filter(babynames, name == "Phil", sex == "M") %>%
summarise(min = min(prop), mean = mean(prop),
max = max(prop))
```

```
Phil <- filter(babynames, name == "Phil", sex == "M")  
summarise(Phil, min = min(prop), mean = mean(prop),  
max = max(prop))
```

```
babynames %>%  
filter(name == "Phil", sex == "M") %>%  
summarise(min = min(prop), mean = mean(prop),  
max = max(prop))
```

Shortcut to type %>%

Cmd + **Shift** + **M** (Mac)

Ctrl + **Shift** + **M** (Windows)

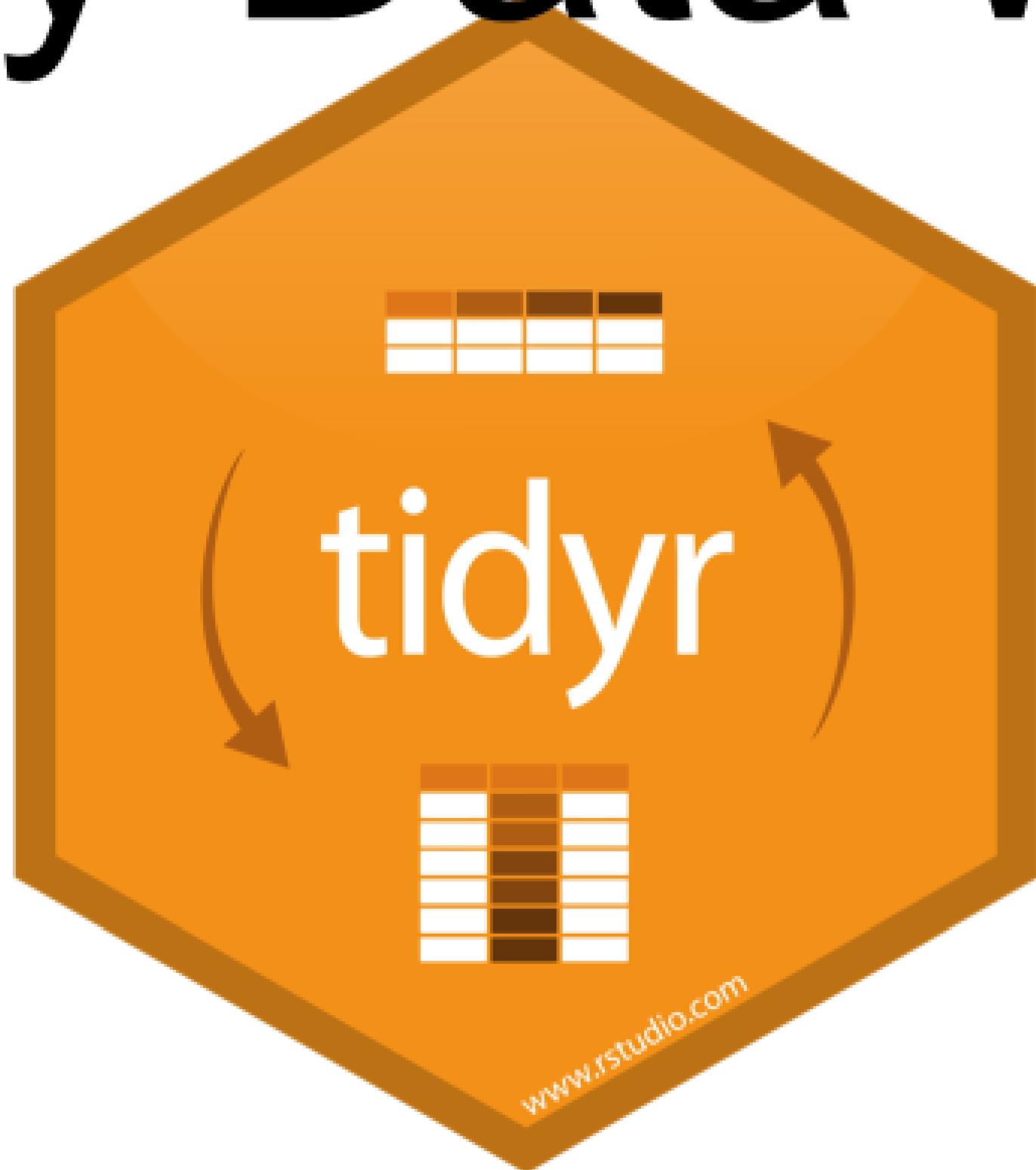


Your Turn – Pick One of Interest:

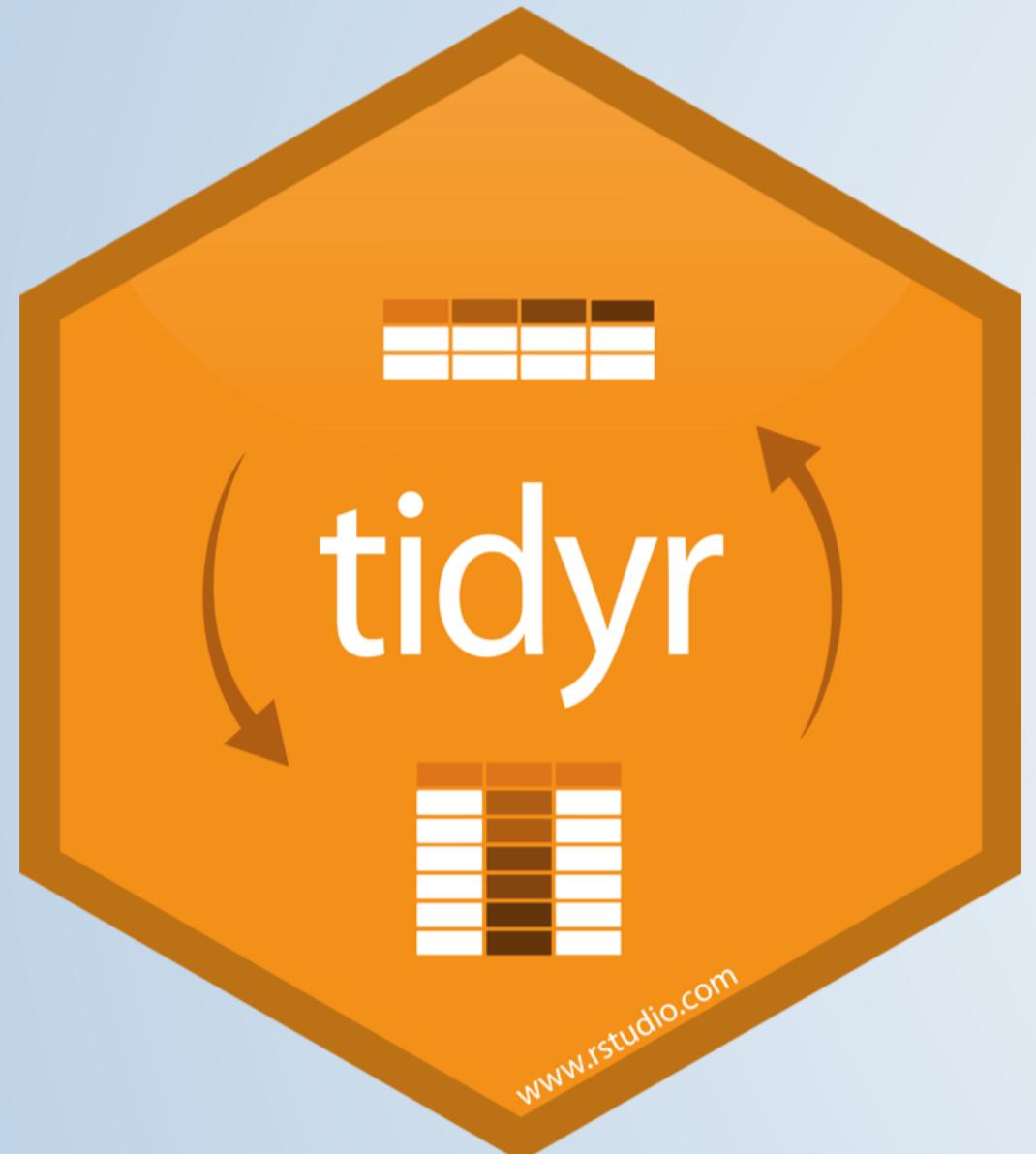
10min

- 03-Transform-Exercises.Rmd Beginner
- 1_dplyr_tidyr_r4ds_tidyverse.Rmd Advanced
- In folder RMD_Clinical_Tidyverse, A gentle guide to Tidy statistics in R.rmd Clinical

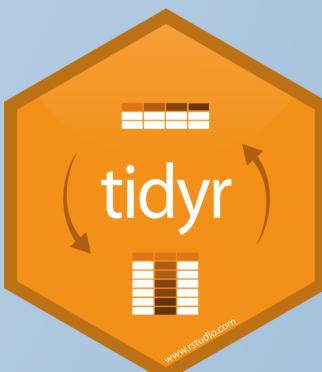
Tidy Data with



tidyr



A package that reshapes the layout of tabular data.



tidyr

The goal of `tidyr` is to help you create tidy data. Tidy data is data where:

- Each variable is in a column.
- Each observation is a row.
- Each value is a cell.

Make Taller and Make Wider

- `gather()` - takes multiple columns, and gathers them into key-value pairs: it makes “wide” data longer.
- `spread()` - takes two columns (key & value), and spreads into multiple columns: it makes “long” data wider.

Separate and unite columns

- `separate()` - Separate one column into multiple columns.
- `unite()` - Unite multiple columns into one.

Your Turn: 3min

- 1_dplyr_tidyr_r4ds_tidyverse.Rmd
- At Bottom

BREAK TIME

- Fun Video

R Markdown

R Markdown

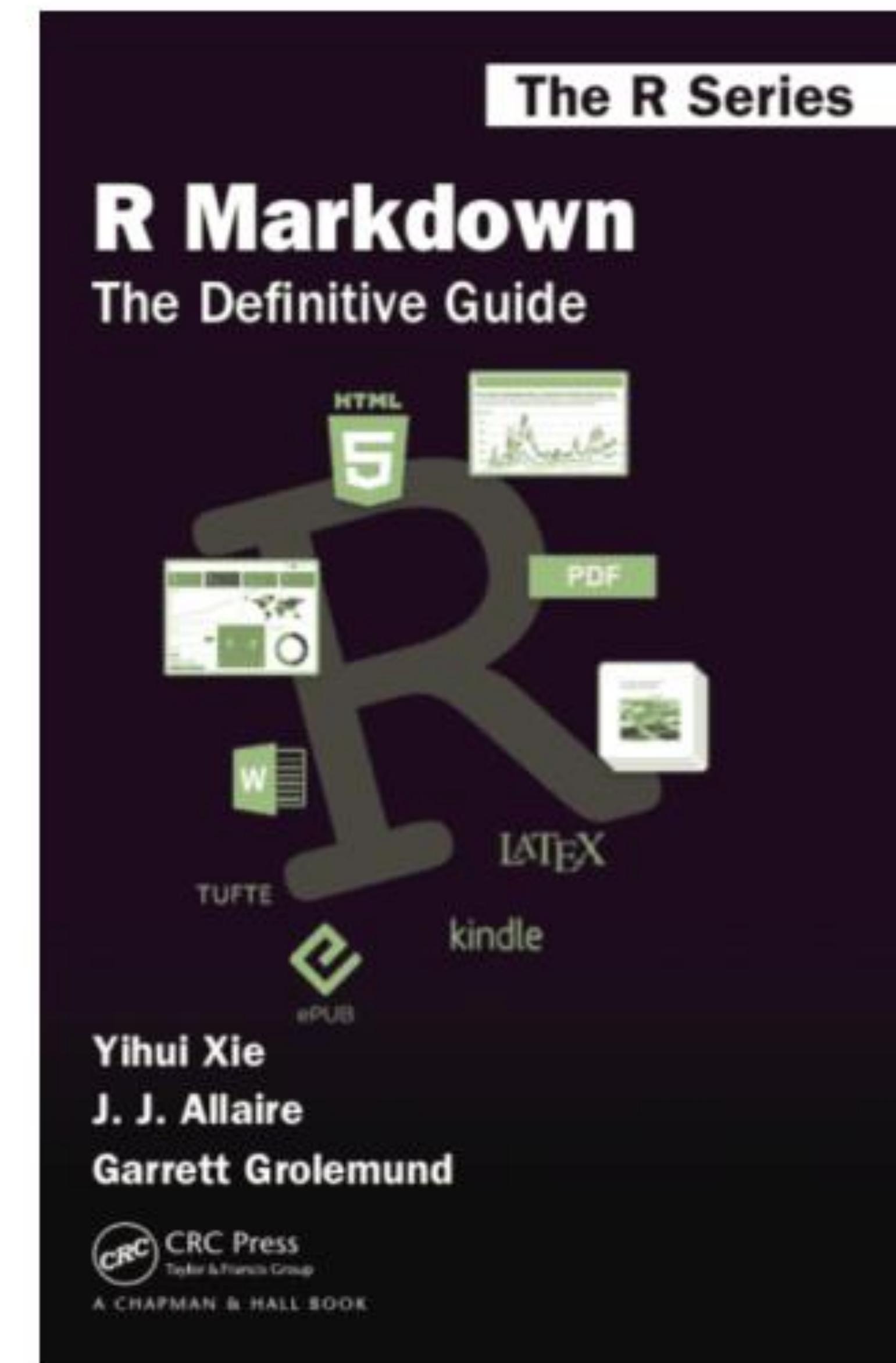
An authoring format for Data Science.

The screenshot shows the RStudio interface with an R Markdown file open. The file contains the following content:

```
1 ---  
2 title: "R Notebook"  
3 output: html_notebook  
4 ---  
5  
6 Text written in **markdown**  
7  
8 ```{r}  
9 # code written in R  
10 (x <- rnorm(7))  
11 ...  
12  
13 Text written in _markdown_  
14  
15 ```{r}  
16 # code written in R  
17 hist(x)  
18 ...  
18.4 (Top Level 2)  
Console
```

Annotations with arrows point to specific parts of the code:

- A callout bubble points to the code chunk boundary: ````{r}`. It contains the text: **Code goes in a chunk**.
- A callout bubble points to the green play button icon in the toolbar of the code chunk area. It contains the text: **Click to run code in chunk**.
- A callout bubble points to the output console area where the R command `(x <- rnorm(7))` has been run and its result `[1] -1.2 1.0 -0.5 0.9 -0.6 -1.1 -1.5` is displayed. It contains the text: **Code result**.



bookdown.org/yihui/rmarkdown/

ONLINE, FREE



rmarkdown

- Starts as a notebook style interface
 - Mixes code with prose
- Knits to dozens of different formats
 - HTML
 - PDF
 - Handouts
 - Books
 - Reports
 - dashboards
 - Interactive shiny apps
 - Articles
 - Websites

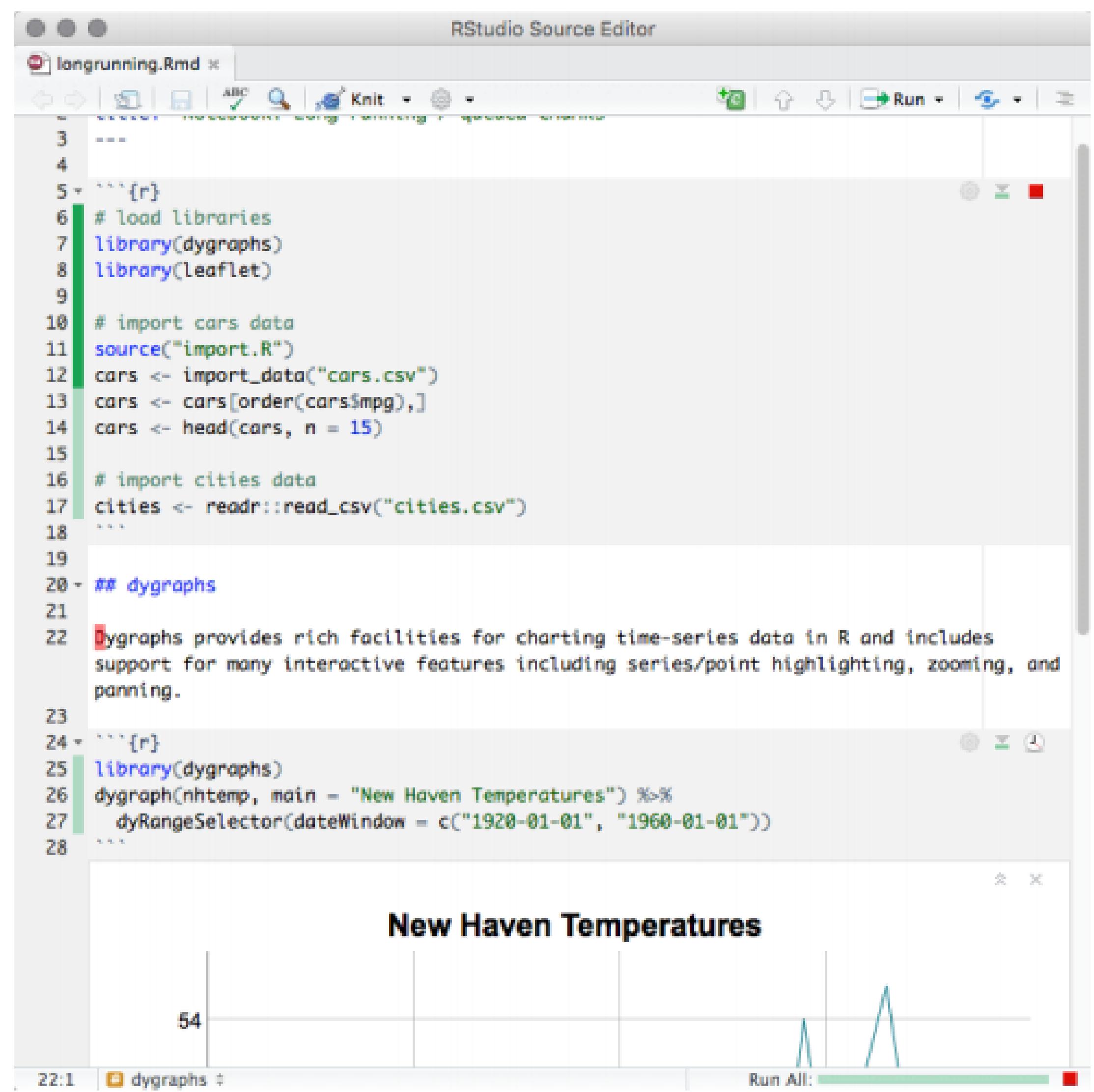
Example

RStudio Source Editor

```
longrunning.Rmd
```

```
3 ---
4
5 ```{r}
6 # load libraries
7 library(dygraphs)
8 library(leaflet)
9
10 # import cars data
11 source("import.R")
12 cars <- import_data("cars.csv")
13 cars <- cars[order(cars$mpg),]
14 cars <- head(cars, n = 15)
15
16 # import cities data
17 cities <- readr::read_csv("cities.csv")
18 ...
19
20 ## dygraphs
21
22 dygraphs provides rich facilities for charting time-series data in R and includes support for many interactive features including series/point highlighting, zooming, and panning.
23
24 ```{r}
25 library(dygraphs)
26 dygraph(nhtemp, main = "New Haven Temperatures") %>%
27   dyRangeSelector(dateWindow = c("1920-01-01", "1960-01-01"))
28 ...
```

New Haven Temperatures



Your Turn: 3mins

- Go to 3_Report.
- Open 01-RMarkdown-Exercises.Rmd.
- Read through the file and do everything it tells you to do.

KNITR IS MULTILINGUAL!

 **SAS**

 **PYTHON**

 **MORE**

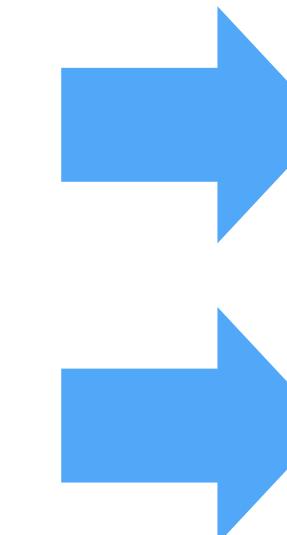
Your Turn

- demo-notebook.Rmd in python folder
- Which section is changing the language engine via knitr?

Parameters

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

params list
**elements and
values**



```
---
```

```
title: "Untitled"
```

```
output: html_document
```

```
params:
```

```
filename: "data.csv"
```

```
symbol: "GOOG"
```

```
--
```

Access as **params\$filename** and **params\$symbol**

Your Turn: 5mins

- Part 1 - Easy
 - 1_RMD_Stocks.Rmd
 - Pick a new stock and generate a new report
 - Now Make it a PDF
-
- Part 2 - Harder
 - Go to 05-Report-Exercise.Rmd
 - See if you can make it parameterized for your name

R Markdown Render Function

rmarkdown::render

Render at the command line with YAML options

```
> render("doc.Rmd")
```

Render at the command line, override output format.

```
> render("doc.Rmd", "html_document")
```

Render at the command line to multiple formats.

```
> render("doc.Rmd", c("html_document", "pdf_document"))
```

rmarkdown::render

Render at the command line with YAML options

```
> render("doc.Rmd")
```

Render at the command line, set parameters.

```
> render("doc.Rmd", params = list(  
  filename = "other_data.csv",  
  symbol = "AAPL")
```

Your Turn: 2mins

- 3_RMD_stock_Flex_CSS
- Render the report programatically using the render function.
- Now do it manually using the “Knit with Parameters” button

What about many reports?

Say Hi to purrr



Demo Example

- 6_RMD_Report_Versions
- Don't forget to set the WD
- Review the airplane-report.Rmd
- Notice how it is connecting to a DB?
- build_airplane_report, run this function as well as the list above it
- Then run the purr command at the bottom
- See how the reports generate dynamically?
- How would you automate this? Ask your neighbor

What if I Start to Have a
Collection of RMDs or If I
Want a Website?

Other R Markdown Output Types

- Blogdown
- RMD Websites - Example
- Bookdown
- Presentations
- Package Documentation

Your Turn: 5mins

- Go to the 11_RMD_IL_Home_Prices and knit the portfolio.Rmd
- Now go to 12_RMD_Stocks_RMarkdown_Website
- Knit the Index.RMD file...
- How is this different than what we have done so far?
How are they similar?
- Now Knit index.Rmd in
8_RMD_Immunogenicity_RMarkdown_Distill
- How is this different? What about Blogdown and Bookdown? Same or different?

But all of my data are in DBs?

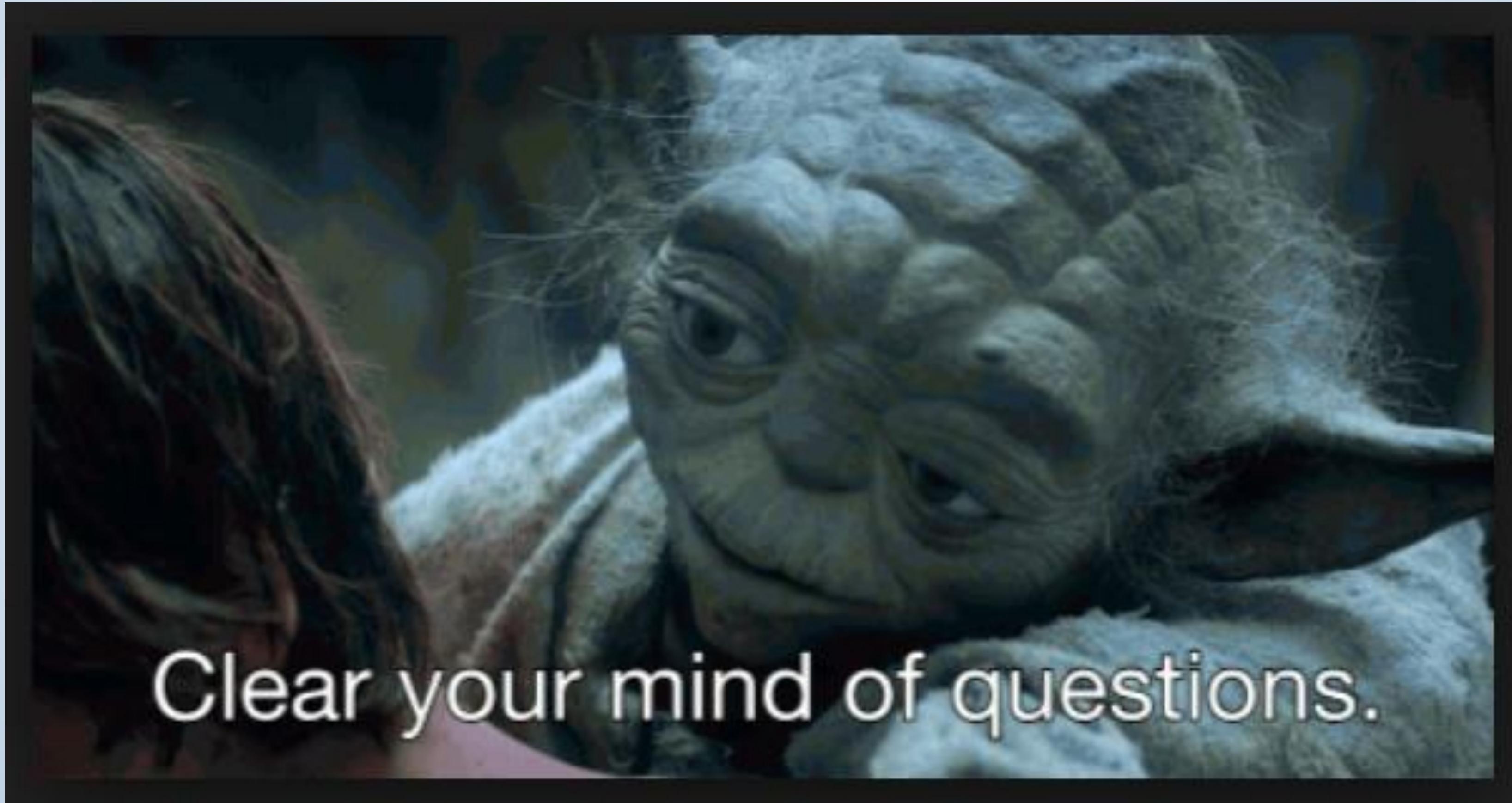
DB - Three ways to write queries

1. DBI code
2. dplyr syntax
3. R Notebook SQL language engine

Your Turn

- `1_DB_Examples`
- `quick_db_demo.Rmd`
- Review `5_RMD_Flex_Database` to see a report built on data in a DB

Q/A...



Clear your mind of questions.