

```
library(dplyr)
```

```
rladies_global %>%  
  filter(city == 'Austin')
```



R FOR DATA SCIENCE: R Markdown and graphics for communication (ggplot2)



Hello!

Welcome to R-Ladies



**Thanks to
our sponsors!**

R Studio | Web.com



1.

Introduction

R language, RStudio,
R4DS Workshop series



Three things you'll need to install

1. **Install R** -- this is the open-source programming language we'll use (download via CRAN -- Comprehensive R Archive Network)
2. **Install RStudio** -- this is the most popular IDE for R and will make your life a lot easier (download from rstudio.com/download)
3. **Install the tidyverse** -- this is the group of packages we'll use within R to work with data. Install with one line of code in R:
`install.packages("tidyverse")`



1b. Introduction

R for Data Science Workshop Series

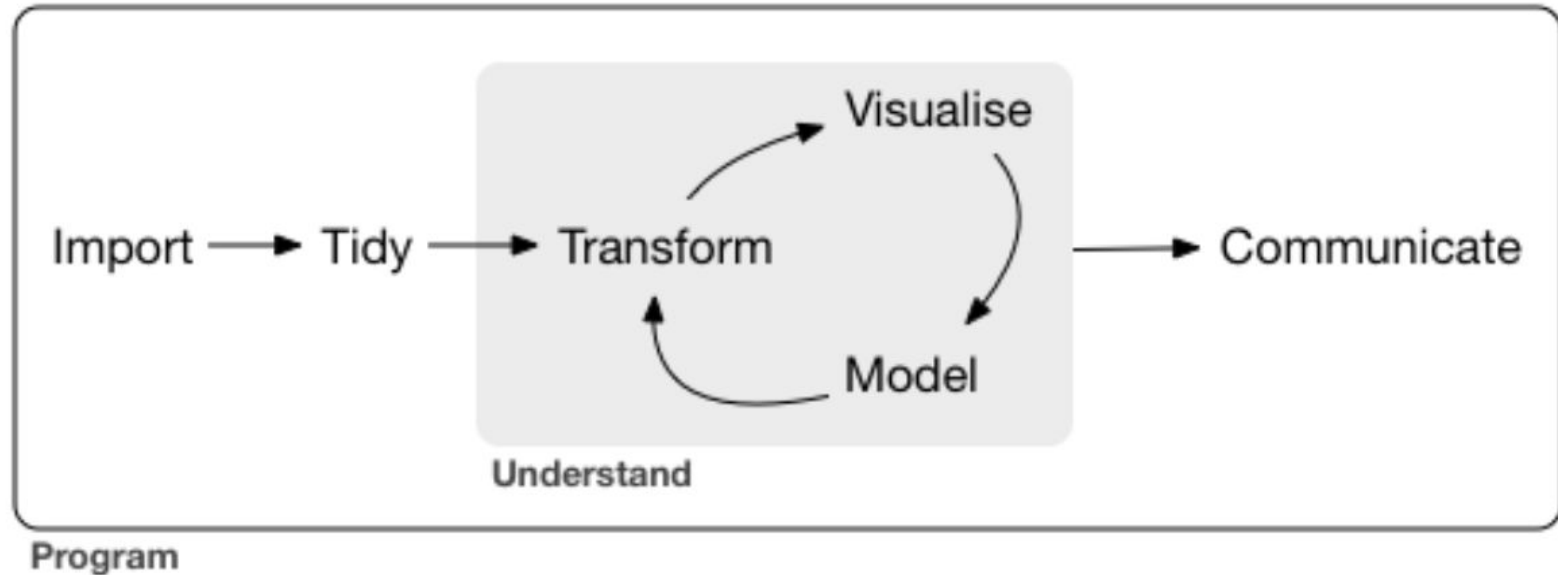


R4DS

Workshop Series

- [Exploring Data with ggplot2 + dplyr](#) [DONE]
- [Exploratory Data Analysis and Workflow](#) [DONE]
- [Data Wrangling in the Tidyverse](#) [November 28]
- [Programming -- Functions, Vectors, and Iteration](#) [December 13]
- [Modeling with modelr, purrr, and broom](#) [January 24]
- [Communicating Results with rmarkdown and ggplot2](#) [February 21]

The data science process (tidied)





What is the tidyverse?

- Collection of R packages based on tidy data principles
- Designed to work together
- An easier way to code!
- AKA “Hadleyverse” (most packages written by Hadley Wickham)

What is the tidyverse?



What is tidy data?

- Each variable is a column
- Each observation is a row
- Each type of observational unit is a table

id	artist	track	time
1	2 Pac	Baby Don't Cry	4:22
2	2Ge+her	The Hardest Part Of ...	3:15
3	3 Doors Down	Kryptonite	3:53
4	3 Doors Down	Loser	4:24
5	504 Boyz	Wobble Wobble	3:35
6	98~0	Give Me Just One Nig...	3:24
7	A*Teens	Dancing Queen	3:44
8	Aaliyah	I Don't Wanna	4:15
9	Aaliyah	Try Again	4:03
10	Adams, Yolanda	Open My Heart	5:30
11	Adkins, Trace	More	3:05
12	Aguilera, Christina	Come On Over Baby	3:38
13	Aguilera, Christina	I Turn To You	4:00
14	Aguilera, Christina	What A Girl Wants	3:18
15	Alice DeeJay	Better Off Alone	6:50



2.

R Markdown

+ formats

+ workflow



What is R Markdown?

A tool for integrating prose, code, and results in a fully reproducible way.

Designed to be used 3 ways:

- + Communicating with decision makers who want to focus on conclusions (rather than the code behind analysis)
- + Collaborating with other data scientists (like future you!) who are interested in conclusions and code
- + As an environment to **do** analysis, like a modern lab notebook



R Markdown

File Basics

A plain-text file with a .Rmd extension

Contains three types of content:

1. An optional YAL header surrounded by `---`
2. Chunks of R code surrounded by `````
3. Text mixed with simple formatting like `#` heading and `_italics_`



Using R Markdown

Create a .Rmd file:

File -> New File -> R Markdown

Add a code chunk:

click insert or Cmd/Ctrl-Alt-I

Run each code chunk:

click run or Cmd/Ctrl-Shift-Enter

Produce full report:

click 'Knit' or Cmd/Ctrl-Shift-K

R Markdown and knitr

When you knit a document,

- + knitr executes all code chunks
- + knitr creates a new Markdown (.md) document
 - + includes both code and output
- + Markdown file is processed by pandoc



Text Formatting With Markdown

syntax

Plain text

End a line with two spaces to start a new paragraph.

italics and *_italics_*

****bold**** and **__bold__**

superscript^{^2^}

~~~~strikethrough~~~~

[link](www.rstudio.com)

# Header 1

## Header 2

### Header 3

#### Header 4

##### Header 5

##### Header 6

## becomes

Plain text

End a line with two spaces to start a new paragraph.

*italics* and *italics*

**bold** and **bold**

superscript<sup>2</sup>

~~strikethrough~~

[link](#)

# Header 1

## Header 2

### Header 3

#### Header 4

##### Header 5

###### Header 6

# Text Formatting

## With Markdown (Cont.)

image: 

horizontal rule (or slide break):

\*\*\*

> block quote

\* unordered list

- \* item 2
  - + sub-item 1
  - + sub-item 2

1. ordered list

- 2. item 2
  - + sub-item 1
  - + sub-item 2

| Table Header | Second Header |
|--------------|---------------|
| Table Cell   | Cell 2        |
| Cell 3       | Cell 4        |



image:

horizontal rule (or slide break):

block quote

- unordered list
- item 2
  - sub-item 1
  - sub-item 2

1. ordered list

- 2. item 2
  - sub-item 1
  - sub-item 2

| Table Header | Second Header |
|--------------|---------------|
| Table Cell   | Cell 2        |
| Cell 3       | Cell 4        |



# R Markdown

## Creating Code Chunks

Three ways to insert a code chunk:

1. The insert button icon in RStudio editor toolbar
2. Keyboard shortcut: Cmd/Ctrl-Alt-I
3. Manually typing delimiters ````{r}` and `````



# R Markdown

## Naming Code Chunks

Chunks can be given an optional name: ````{r name_here}`

Three advantages:

1. You can easily navigate to specific chunks using the drop-down code navigator in bottom-left of script editor
2. Makes it easier to access graphics produced by named chunks
3. You can cache “networks” of chunks to avoid re-performing expensive operations on each code run

Note: chunk named ‘setup’ will be run automatically before any others



# R Markdown

## Chunk Options

| option                  | default  | effect                                                    |
|-------------------------|----------|-----------------------------------------------------------|
| <code>eval</code>       | TRUE     | Whether to evaluate the code and include its results      |
| <code>echo</code>       | TRUE     | Whether to display code along with its results            |
| <code>warning</code>    | TRUE     | Whether to display warnings                               |
| <code>error</code>      | FALSE    | Whether to display errors                                 |
| <code>message</code>    | TRUE     | Whether to display messages                               |
| <code>tidy</code>       | FALSE    | Whether to reformat code in a tidy way when displaying it |
| <code>results</code>    | "markup" | "markup", "asis", "hold", or "hide"                       |
| <code>cache</code>      | FALSE    | Whether to cache results for future renders               |
| <code>comment</code>    | "###"    | Comment character to preface results with                 |
| <code>fig.width</code>  | 7        | Width in inches for plots created in chunk                |
| <code>fig.height</code> | 7        | Height in inches for plots created in chunk               |

For more details visit [yihui.name/knitr/](https://yihui.name/knitr/)



# R Markdown

## Inline Code

Inline code lets you quickly combine text with generated code outputs:

We have data about `nrow(diamonds)` diamonds. Only `nrow(diamonds) - nrow(smaller)` are larger than 2.5 carats. The distribution of the remainder is shown below:

We have data about 53940 diamonds. Only 126 are larger than 2.5 carats. The distribution of the remainder is shown below:

Tip: Use `format()` to control number of digits and `big.mark` for readability



# R Markdown

## Troubleshooting

1. Restart R, then “Run all chunks”
2. Checking the working directory of interactive environment vs. R Markdown environment (`getwd()`)
3. Look for other differences between interactive environment and R Markdown environment



# R Markdown

## Formats

- + HTML Document (this is default)
- + Other Documents (PDF, Word, RTF, Github, etc.)
- + Notebooks (like HTML docs but contain full source code)
- + Presentations (ioslides, slidy, beamer)
- + Dashboards (flexdashboard)
- + Bookdown
- + Blogdown

HTML formats can be interactive!

- + htmlwidgets
- + shiny





# R Markdown

## Workflow

R Markdown blurs the lines between interactive exploration and more long-term code capture.

Think of it as an analysis notebook:

- + Records what you did and why
- + Supports rigorous thinking
- + Speeds up time it takes to write up analysis
- + Helps others understand your work
- + Supports good reproducibility habits



# More resources: rstudio.com/cheatsheets

## Cheat Sheet

## Reference Guide

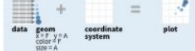
### Data Visualization with ggplot2 : : CHEAT SHEET

#### 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.



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



Complete the template below to build a graph.

```
ggplot(data = DATA) + GEOM_FUNCTION()
(mapping = aes(COORDINATE_VARIABLES))
size = SIZE, position = POSITION) +
COORDINATE_FUNCTION()
FACTOR_FUNCTION()
SCALE_FUNCTION()
CHUNK_OPTIONS
```

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

ggplot(mpg, aes(x = hwy, data = mpg, geom = "point")) Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last\_plot() Returns the last plot

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5" x 5" file named "plot.png" in working directory. Matches file type to file extension.

#### 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**

- `geom_blank()` (blank for expanding limits)
- `geom_curve()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_line()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_path()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_point()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_polygon()` (aes(group = group))
- `geom_rect()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_raster()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_smooth()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_text()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_tile()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_vline()` (aes(x, y, alpha, color, fill, shape, size, weight))

**LINE SEGMENTS**

- `geom_abline()` (aes(intercept, slope))
- `geom_hline()` (aes(yintercept))
- `geom_vline()` (aes(xintercept))
- `geom_line()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_path()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_polygon()` (aes(group = group))
- `geom_rect()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_raster()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_smooth()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_text()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_tile()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_vline()` (aes(x, y, alpha, color, fill, shape, size, weight))

**ONE VARIABLE continuous**

- `geom_area()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_bar()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_boxplot()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_density()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_histogram()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_jitter()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_line()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_path()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_polygon()` (aes(group = group))
- `geom_rect()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_raster()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_smooth()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_text()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_tile()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_vline()` (aes(x, y, alpha, color, fill, shape, size, weight))

**ONE VARIABLE discrete**

- `geom_area()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_bar()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_boxplot()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_density()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_histogram()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_jitter()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_line()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_path()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_polygon()` (aes(group = group))
- `geom_rect()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_raster()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_smooth()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_text()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_tile()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_vline()` (aes(x, y, alpha, color, fill, shape, size, weight))

**TWO VARIABLES**

- `geom_area()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_bar()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_boxplot()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_density()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_histogram()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_jitter()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_line()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_path()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_polygon()` (aes(group = group))
- `geom_rect()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_raster()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_smooth()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_text()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_tile()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_vline()` (aes(x, y, alpha, color, fill, shape, size, weight))

**THREE VARIABLES**

- `geom_area()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_bar()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_boxplot()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_density()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_histogram()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_jitter()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_line()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_path()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_polygon()` (aes(group = group))
- `geom_rect()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_raster()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_smooth()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_text()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_tile()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_vline()` (aes(x, y, alpha, color, fill, shape, size, weight))

**FOUR VARIABLES**

- `geom_area()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_bar()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_boxplot()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_density()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_histogram()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_jitter()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_line()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_path()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_polygon()` (aes(group = group))
- `geom_rect()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_raster()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_smooth()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_text()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_tile()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_vline()` (aes(x, y, alpha, color, fill, shape, size, weight))

**FIVE VARIABLES**

- `geom_area()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_bar()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_boxplot()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_density()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_histogram()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_jitter()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_line()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_path()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_polygon()` (aes(group = group))
- `geom_rect()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_raster()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_smooth()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_text()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_tile()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_vline()` (aes(x, y, alpha, color, fill, shape, size, weight))

**SIX VARIABLES**

- `geom_area()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_bar()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_boxplot()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_density()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_histogram()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_jitter()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_line()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_path()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_polygon()` (aes(group = group))
- `geom_rect()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_raster()` (aes(xmin, xmax, ymin, ymax, fill, color, size, weight))
- `geom_smooth()` (aes(linetype = "l", size = 1, weight = 1, curvature = 1))
- `geom_text()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_tile()` (aes(x, y, alpha, color, fill, shape, size, weight))
- `geom_vline()` (aes(x, y, alpha, color, fill, shape, size, weight))



### R Markdown Reference Guide

Learn more about R Markdown at [rmarkdown.rstudio.com](http://rmarkdown.rstudio.com)  
Learn more about interactive Docs at [shiny.rstudio.com/articles](http://shiny.rstudio.com/articles)

Contents:

1. Markdown Syntax
2. Knitr chunk options
3. Pandoc options

#### Syntax

Make a code chunk with three back ticks followed by an `r` in braces. End the chunk with three back ticks:

```
```r
paste("Hello", "World!")
```
```

Place code inline with a single back ticks. The first back tick must be followed by an `R`, like this `r` paste("Hello", "World!")

Add chunk options within braces. For example, `echo=FALSE` will prevent source code from being displayed:

```
```{r eval=TRUE, echo=FALSE}
paste("Hello", "World!")
```
```

Learn more about chunk options at <http://yihui.name/knitr/options>

#### Becomes

Make a code chunk with three back ticks followed by an `r` in braces. End the chunk with three back ticks:

```
paste("Hello", "World!")

## [1] "Hello World!"
```

Place code inline with a single back ticks. The first back tick must be followed by an `R`, like this `r` paste("Hello", "World!")

Add chunk options within braces. For example, `echo=FALSE` will prevent source code from being displayed:

```
## [1] "Hello World!"
```

#### Chunk options

| option          | default value | description                                                                                                                                                                                                                                                                     |
|-----------------|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Code evaluation |               |                                                                                                                                                                                                                                                                                 |
| child           | NULL          | A character vector of filenames. Knitr will knit the files and place them into the main document.                                                                                                                                                                               |
| code            | NULL          | Set to R code. Knitr will replace the code in the chunk with the code in the code option.                                                                                                                                                                                       |
| engine          | 'r'           | Knitr will evaluate the chunk in the named language, e.g. engine = 'python'. Run names(knitr::knit_engines\$get()) to see supported languages.                                                                                                                                  |
| eval            | TRUE          | If FALSE, knitr will not run the code in the code chunk.                                                                                                                                                                                                                        |
| include         | TRUE          | If FALSE, knitr will not run the chunk but not include the chunk in the final document.                                                                                                                                                                                         |
| purrr           | TRUE          | If FALSE, knitr will not include the chunk when running purrr() to extract the source code.                                                                                                                                                                                     |
| Result          |               |                                                                                                                                                                                                                                                                                 |
| collapse        | FALSE         | If TRUE, knitr will collapse all the source and output blocks created by the chunk into a single block.                                                                                                                                                                         |
| echo            | TRUE          | If FALSE, knitr will not display the code in the code chunk above it's results in the final document.                                                                                                                                                                           |
| results         | 'markup'      | If 'hide', knitr will not display the code's results in the final document. If 'hold', knitr will delay displaying all output blocks until the end of the chunk. If 'asis', knitr will pass through results without reformatting them (useful if results return raw HTML, etc.) |
| error           | TRUE          | If FALSE, knitr will not display any error messages generated by the code.                                                                                                                                                                                                      |
| message         | TRUE          | If FALSE, knitr will not display any messages generated by the code.                                                                                                                                                                                                            |
| warning         | TRUE          | If FALSE, knitr will not display any warning messages generated by the code.                                                                                                                                                                                                    |
| Code Decoration |               |                                                                                                                                                                                                                                                                                 |



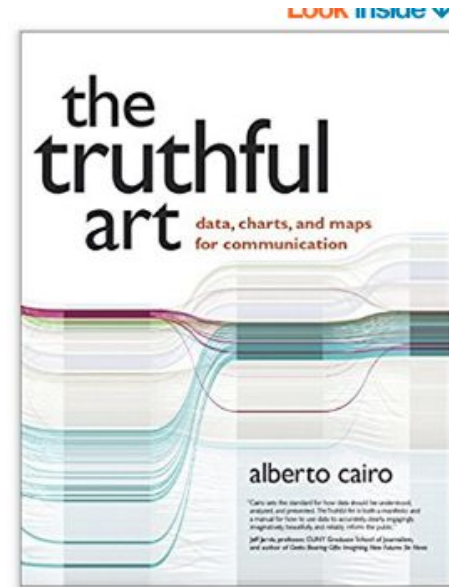


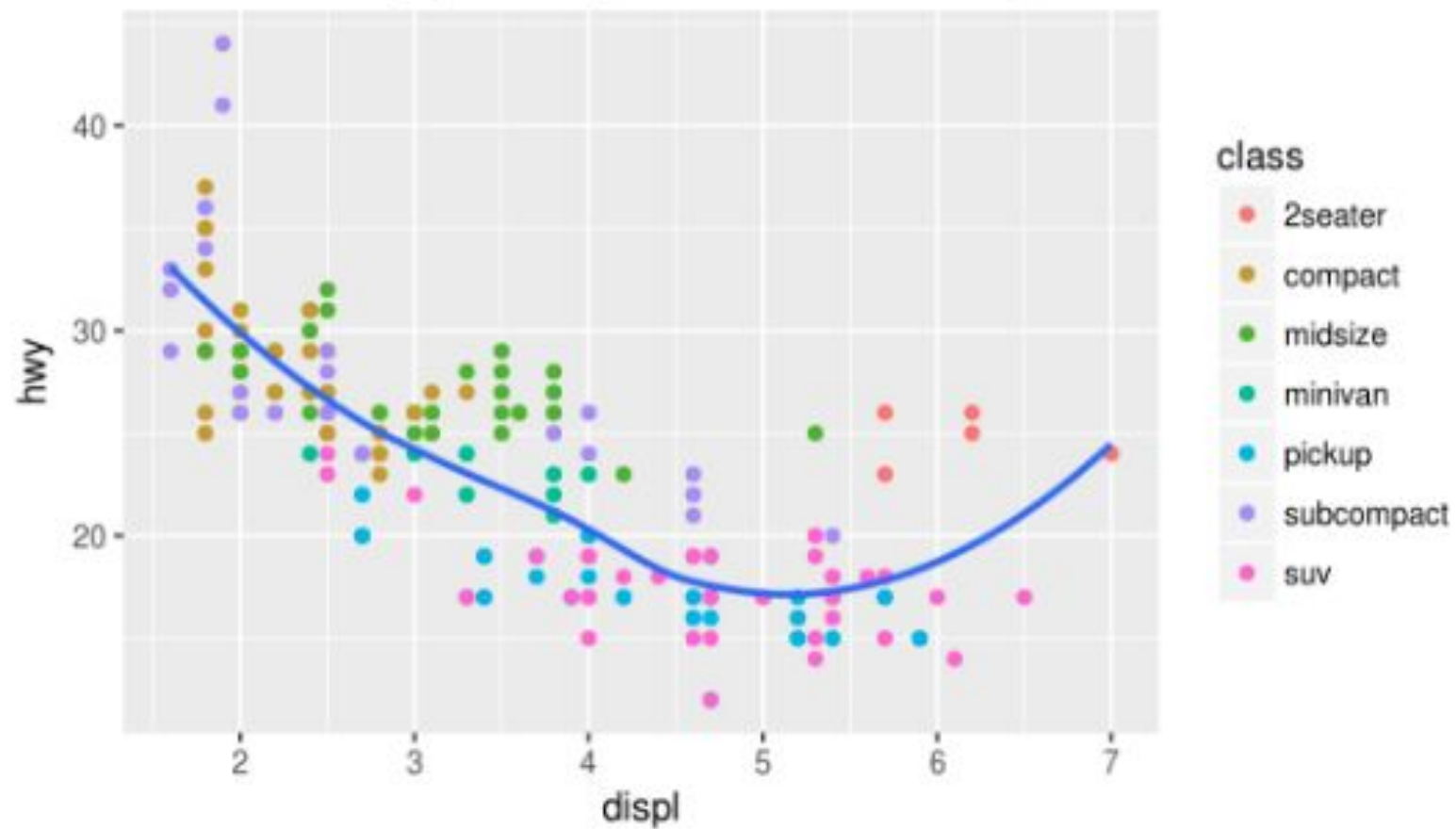
# 3.

## Graphics for communication with ggplot2

# ggplot2 is back!

- Communicate your results
- You know your data, but others likely don't
- GOAL:
  - Make your plots as self explanatory as possible!





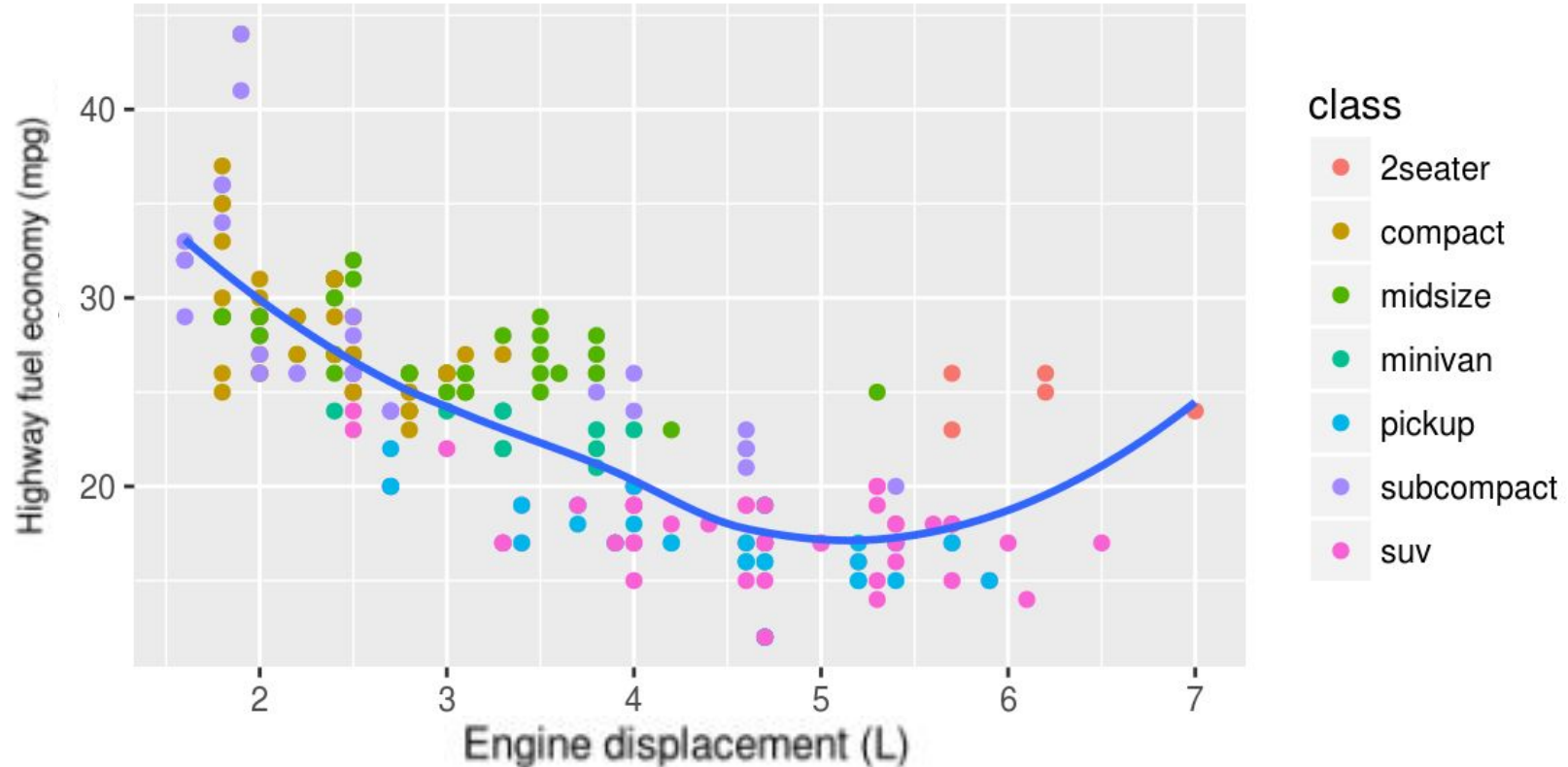
# label

```
ggplot(mpg, aes(displ, hwy)) +  
  geom_point(aes(color = class)) +  
  geom_smooth(se = FALSE) +  
  labs(  
    title = "Fuel efficiency generally decreases with engine size",  
    subtitle = "Two seaters (sports cars) are an exception because  
of their light weight",  
    caption = "Data from fueleconomy.gov",  
  
    x = "Engine displacement (L)",  
    y = "Highway fuel economy (mpg)"  
  )
```

- + Can add mathematical equations as labels--see book for examples

# Fuel efficiency generally decreases with engine size

Two seaters (sports cars) are an exception because of their light weight

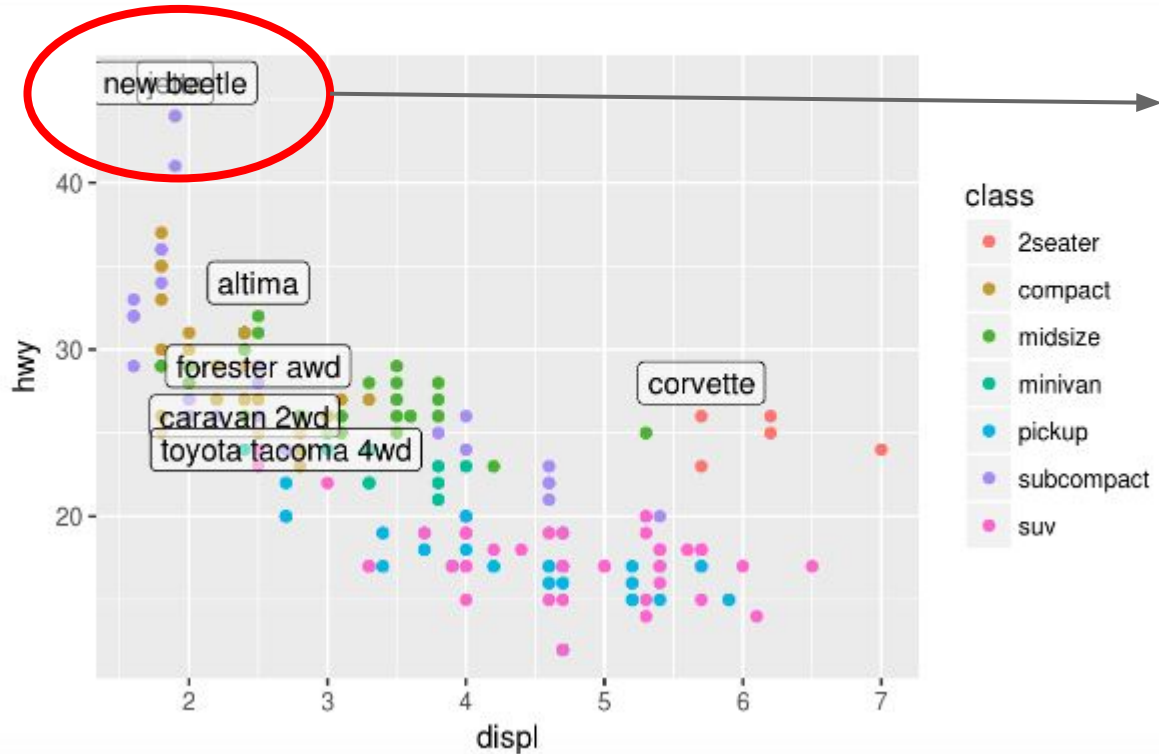


# annotations

- Annotations vs. labels
  - Annotations add information to the data points
  - But annotation uses `geom_label...`
  - In general, good to have a tibble with your labels in it
- `geom_text()`
  - Has label argument
  - Can bring in other datasets with labels
  - Labels with text only
- `geom_label()`
  - Has label and data args too
  - Labels with a box around the label



```
ggplot(mpg, aes(displ, hwy)) +  
  geom_point(aes(colour = class)) +  
  geom_label(aes(label = model), data = best_in_class, nudge_y = 2, alpha = 0.5)
```

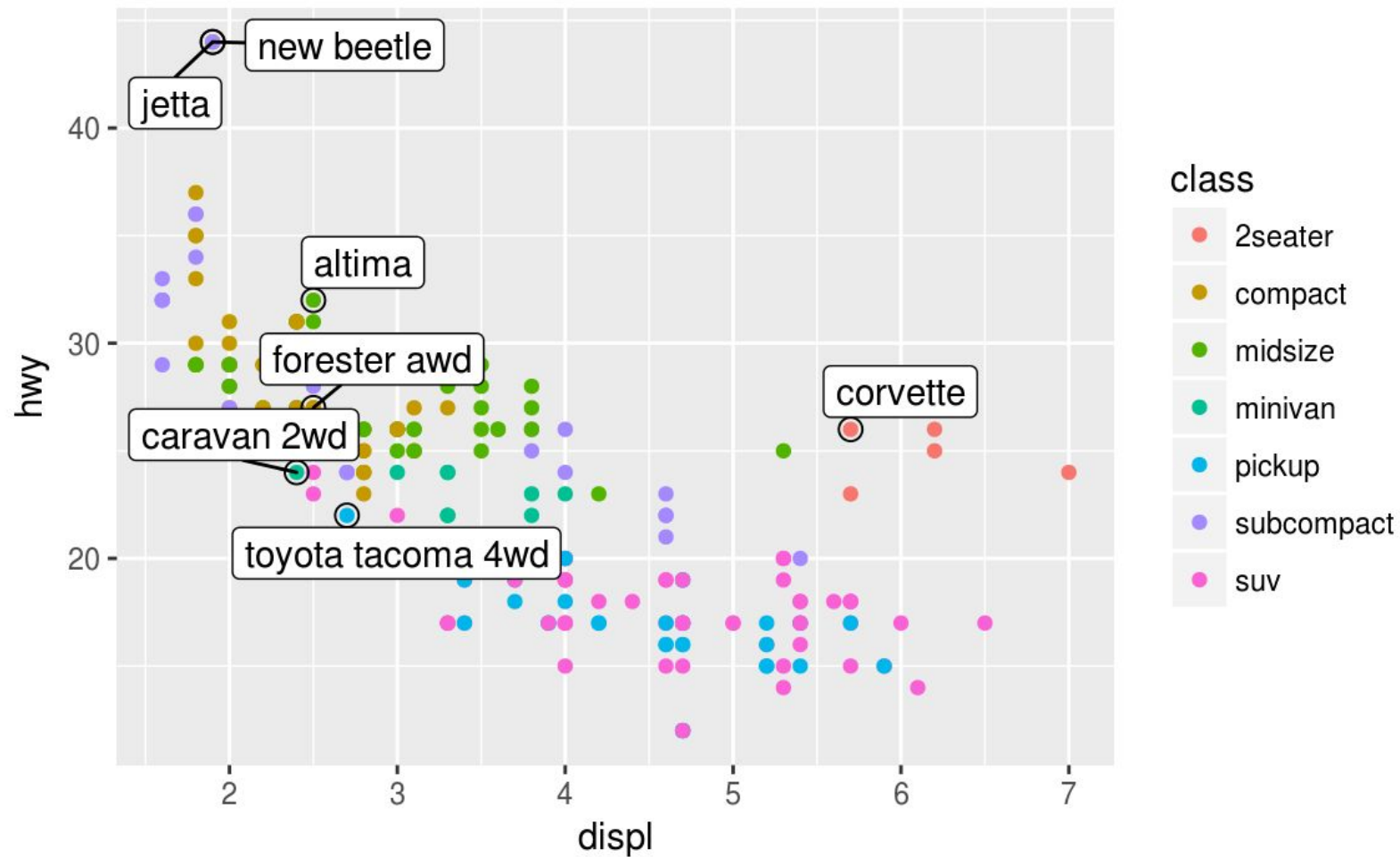


Overlapping  
labels--how  
to solve?

# overlapping labels

- Nudge--moves all labels the same way
- ggrepel package!!
  - Moves so you can see them!

```
ggrepel::geom_label_repel(aes(label =  
model), data = best_in_class)
```



## Other annotations

- Use `geom_hline()` and `geom_vline()` to add reference lines. I often make them thick (`size = 2`) and white (`colour = white`), and draw them underneath the primary data layer. That makes them easy to see, without drawing attention away from the data.
- Use `geom_rect()` to draw a rectangle around points of interest. The boundaries of the rectangle are defined by aesthetics `xmin`, `xmax`, `ymin`, `ymax`.
- Use `geom_segment()` with the `arrow` argument to draw attention to a point with an arrow. Use aesthetics `x` and `y` to define the starting location, and `xend` and `yend` to define the end location.

# scales

- Default scales are part of each ggplot
- Sometimes you want to change this
  - Breaks
    - Control tick marks
  - Labels
    - Control what labels appear on the axes
    - Use NULL to remove entirely
  - Scale\_x\_date--a bit different



# Pic of scales

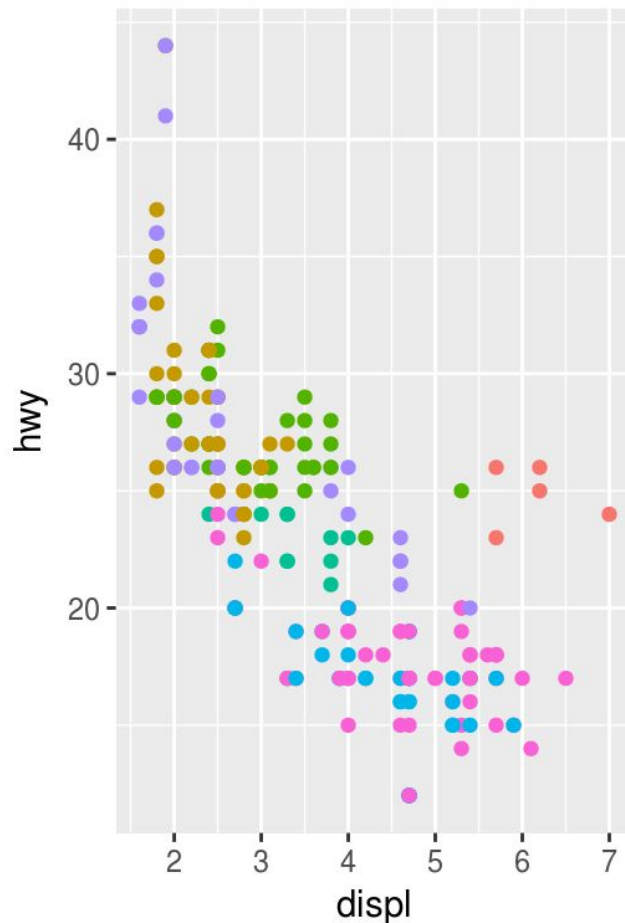
Try in R!

```
ggplot(mpg, aes(displ, hwy)) +  
  geom_point() +  
  scale_y_continuous(breaks = seq(15, 40, by = 5))
```

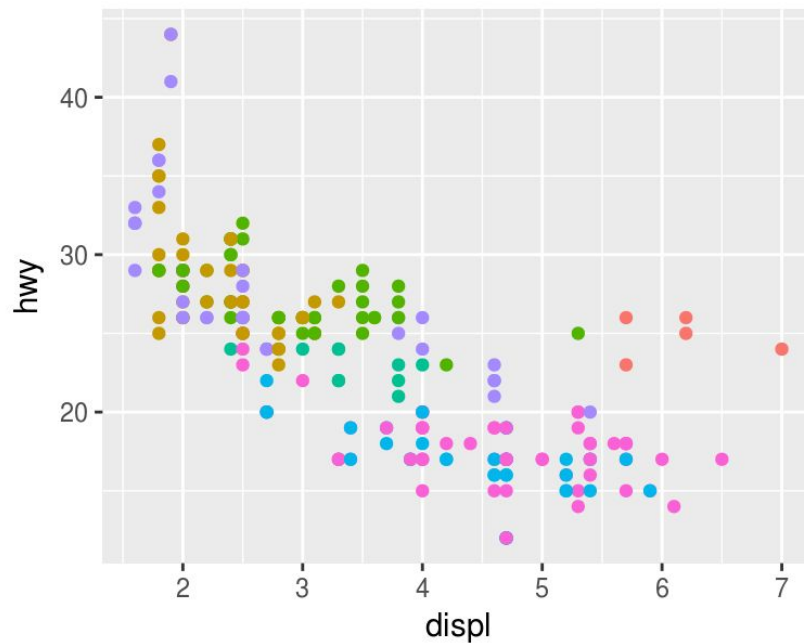
# Working with legends

- Change the position or layout of a legend
  - `base <- ggplot(mpg, aes(displ, hwy)) +  
 geom_point(aes(colour = class))  
 base + theme(legend.position =  
 "left/right/top/bottom")`
- Remove a legend
  - `legend.position = "none"`
- Change the aesthetics of the legend
  - `guides(colour = guide_legend(nrow = 1,  
 override.aes = list(size = 4)))`

class



class



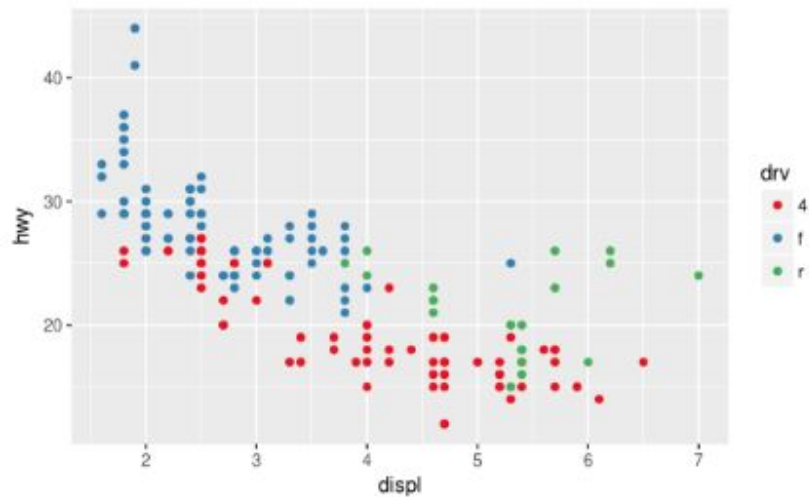
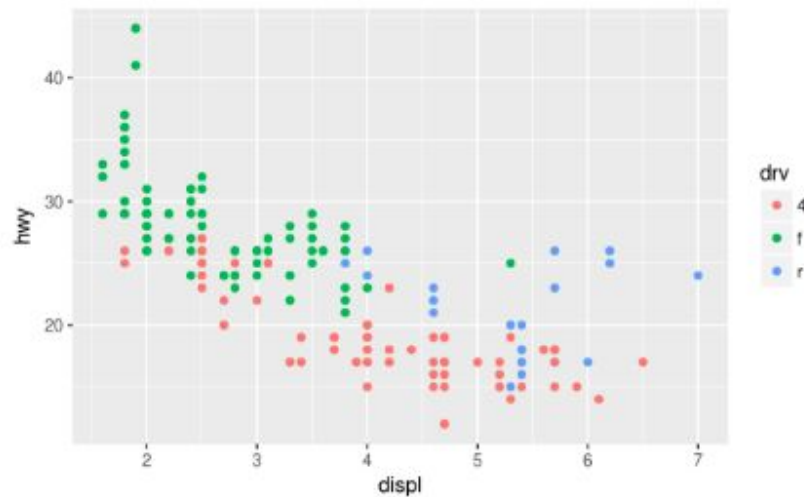


# Changing scales

- Transformed data vs untransformed scales
  - E.g. plot log data, but scale in original form for easier interpretation
- Changing colors!
  - R Color Brewer!
    - ▶ <http://colorbrewer2.org>
    - ▶ [http://www.cookbook-r.com/Graphs/Colors\\_\(ggplot2\)/](http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/)
    - ▶ `scale_fill_brewer()`
    - ▶ `scale_colour_brewer()`
  - `scale_fill_manual()`
    - ▶ Select exact colors you want

```
ggplot(mpg, aes(displ, hwy)) +  
  geom_point(aes(color = drv))
```

```
ggplot(mpg, aes(displ, hwy)) +  
  geom_point(aes(color = drv)) +  
  scale_colour_brewer(palette = "Set1")
```



# zooming

There are three ways to control the plot limits:

1. Adjusting what data are plotted
2. Setting the limits in each scale
3. Setting `xlim` and `ylim` in `coord_cartesian()`

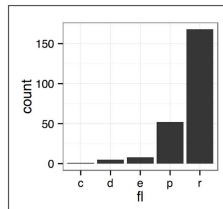
# themes

```
ggplot(mpg, aes(displ, hwy)) +  
  geom_point(aes(color = class)) +  
  geom_smooth(se = FALSE) +  
  theme_bw()
```



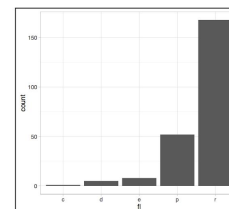
## Themes

Theme functions change the appearance of your plot.



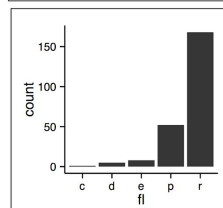
### theme\_bw()

White background  
with grid lines



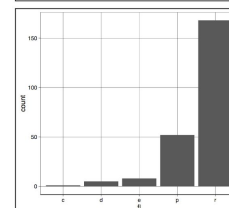
### theme\_light()

Light axes and grid  
lines



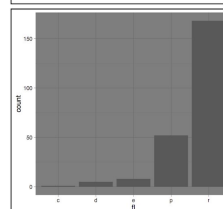
### theme\_classic()

Classic theme,  
axes but no grid  
lines



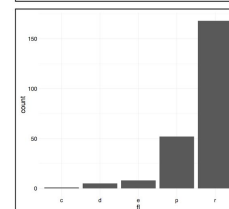
### theme\_linedraw()

Only black lines



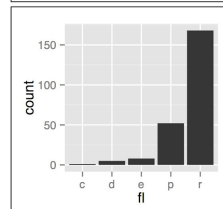
### theme\_dark()

Dark background  
for contrast



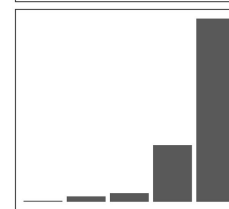
### theme\_minimal()

Minimal theme, no  
background



### theme\_gray()

Grey background  
(default theme)



### theme\_void()

Empty theme, only  
geoms are visible



# saving & learning more

Go to R--other options:

R markdown

- `fig.width`, `fig.height`, `fig.asp`, `out.width` and `out.height`  
`ggsave()`



# R-Ladies Austin

## Upcoming Events

ALL the Ladies in Tech HH [March 7]

From Zero to Web App with Shiny! [March 28]

*Lightning talks?*

*Another book for book club?*