Overview of R and the tidyverse

Lab

PMAP 8521: Program evaluation
Andrew Young School of Policy Studies

Plan for today

Packages and data

Visualize data with ggplot2

Transform data with dplyr

Packages and data

Using packages

```
install.packages("name")
```

Downloads files to your computer

Do this once per computer

```
library("name")
```

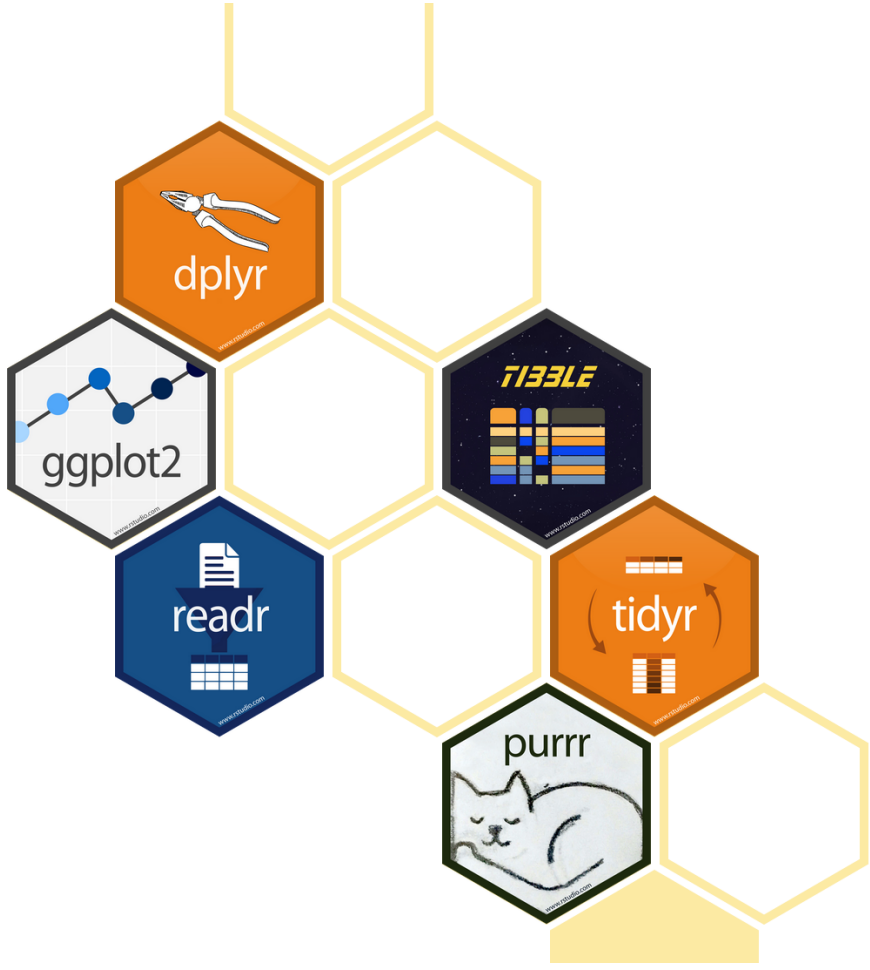
Loads the package

Do this once per R session

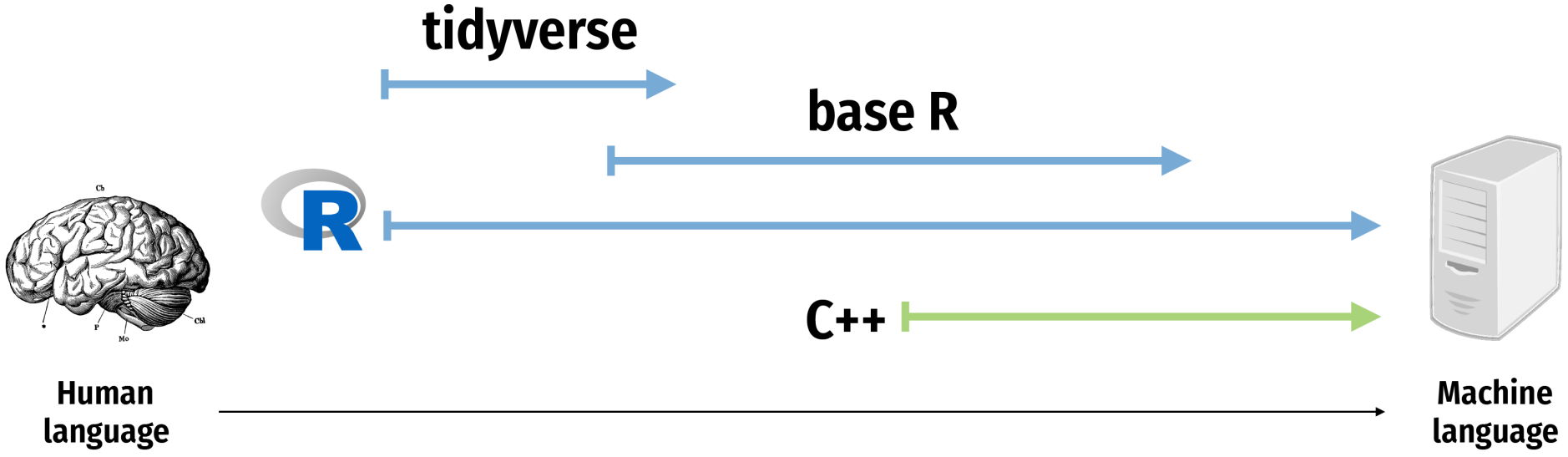
The tidyverse

"The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures."

... the tidyverse makes data science faster, easier and more fun...



The tidyverse



The tidyverse package

```
library(tidyverse)
```

The tidyverse package is a shortcut for installing and loading all the key tidyverse packages

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

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

```
install.packages("ggplot2")
install.packages("dplyr")
install.packages("tidyr")
install.packages("readr")
install.packages("purrr")
install.packages("tibble")
install.packages("stringr")
install.packages("forcats")
install.packages("lubridate")
install.packages("hms")
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("ggplot2")
library("dplyr")
library("tidyr")
library("readr")
library("purrr")
library("tibble")
library("stringr")
library("forcats")
```

Data frames and tibbles

Data frames are the most common kind of data objects; used for rectangular data (like spreadsheets)

Data frames: R's native data object

Tibbles (`tbl`): a fancier enhanced kind of data frame

(You really won't notice a difference in this class)

Vectors

Vectors are a list of values of the same time (all text, or all numbers, etc.)

Make them with `c()`:

```
c(1, 4, 2, 5, 7)
```

You'll usually want to assign them to something:

```
neat_numbers <- c(1, 4, 2, 5, 7)
```

Basic data types

Integer	Whole numbers	`c(1, 2, 3, 4)`
Double	Numbers	`c(1, 2.4, 3.14, 4)`
Character	Text	`c("1", "blue", "fun", "monster")`
Logical	True or false	`c(TRUE, FALSE, TRUE, FALSE)`
Factor	Category	`c("Strongly disagree", "Agree", "Neutral")`

Packages for importing data

Work with plain text data

```
my_data <-
read_csv("file.csv")
```

Work with Excel files

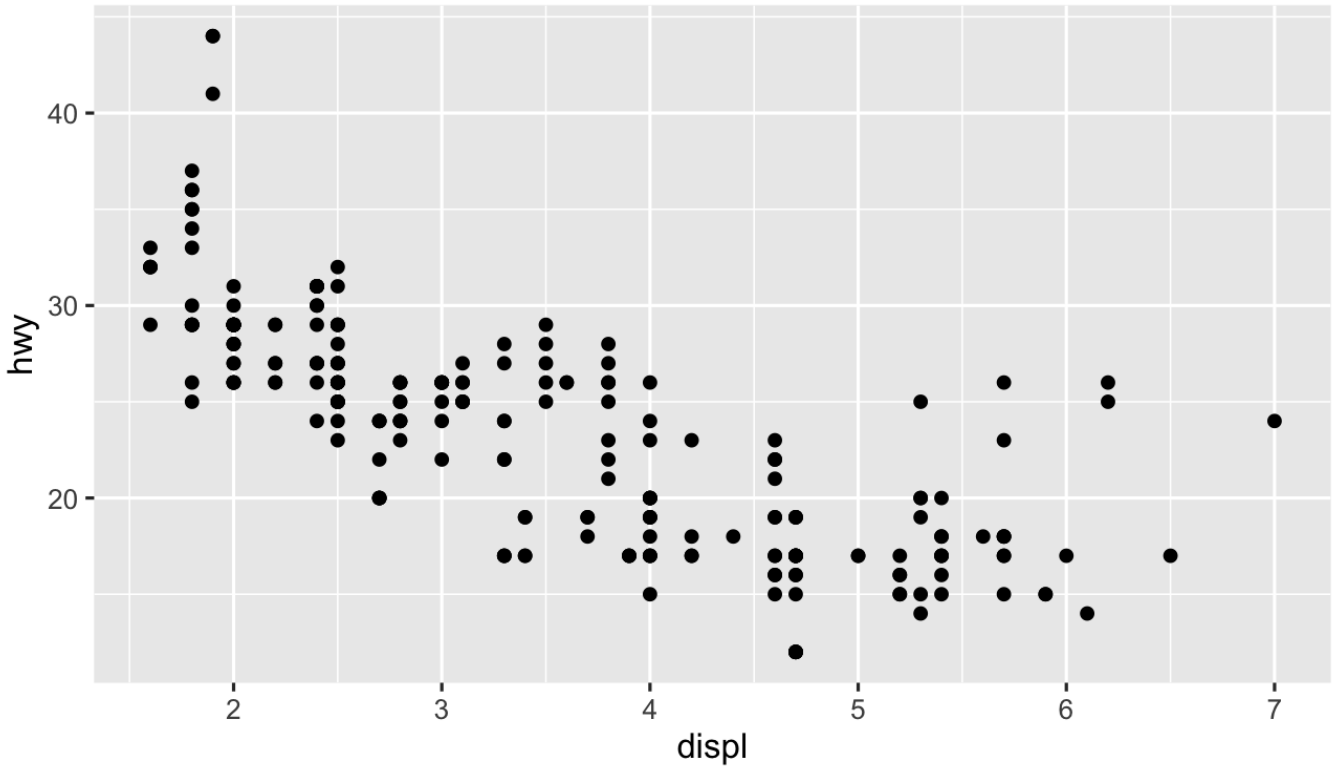
```
my_data <-
read_excel("file.xlsx")
```

Work with Stata, SPSS, and SAS data

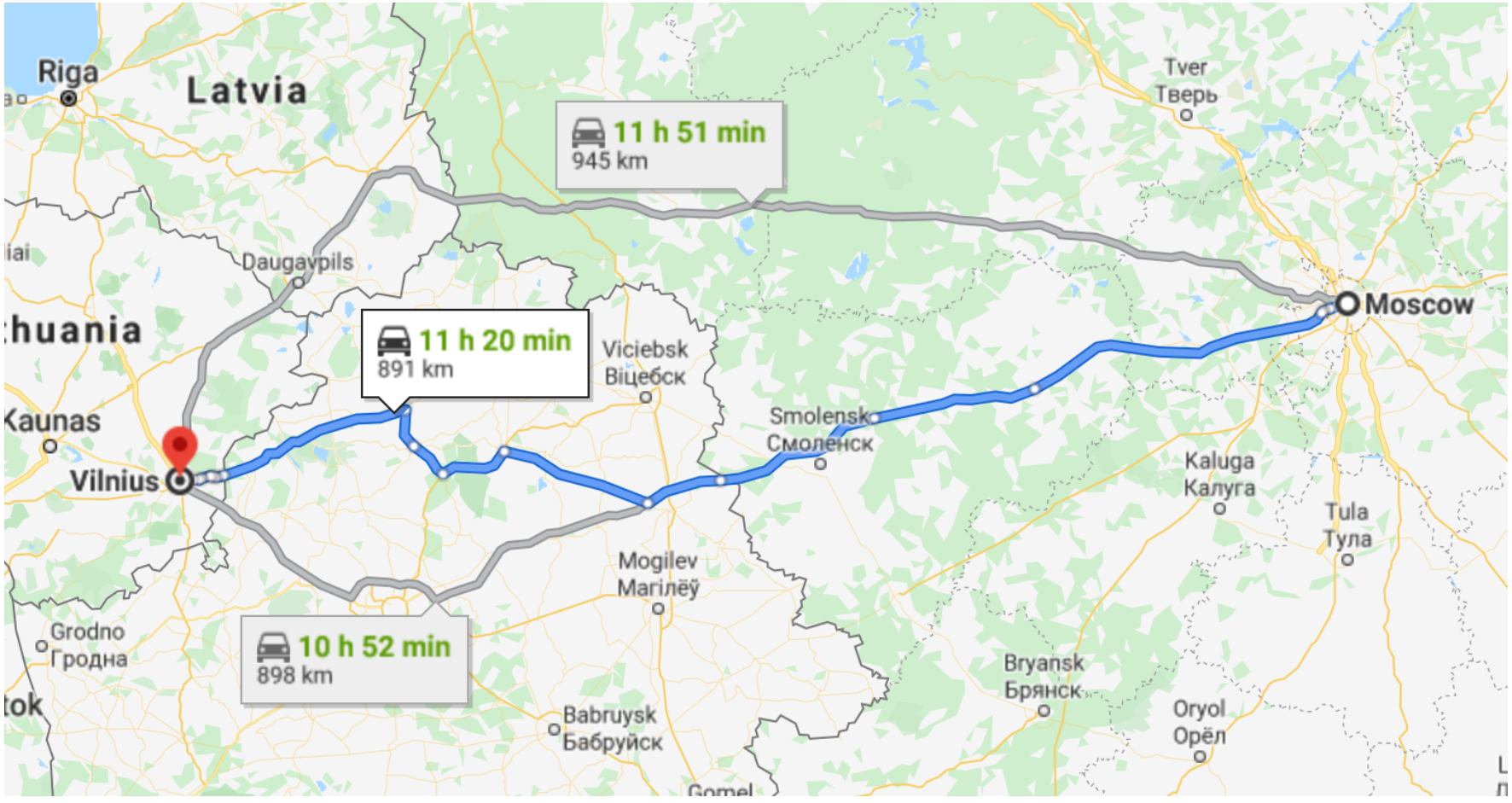
```
my_data <-
read_stata("file.dta")
```

Visualize data with ggplot2

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

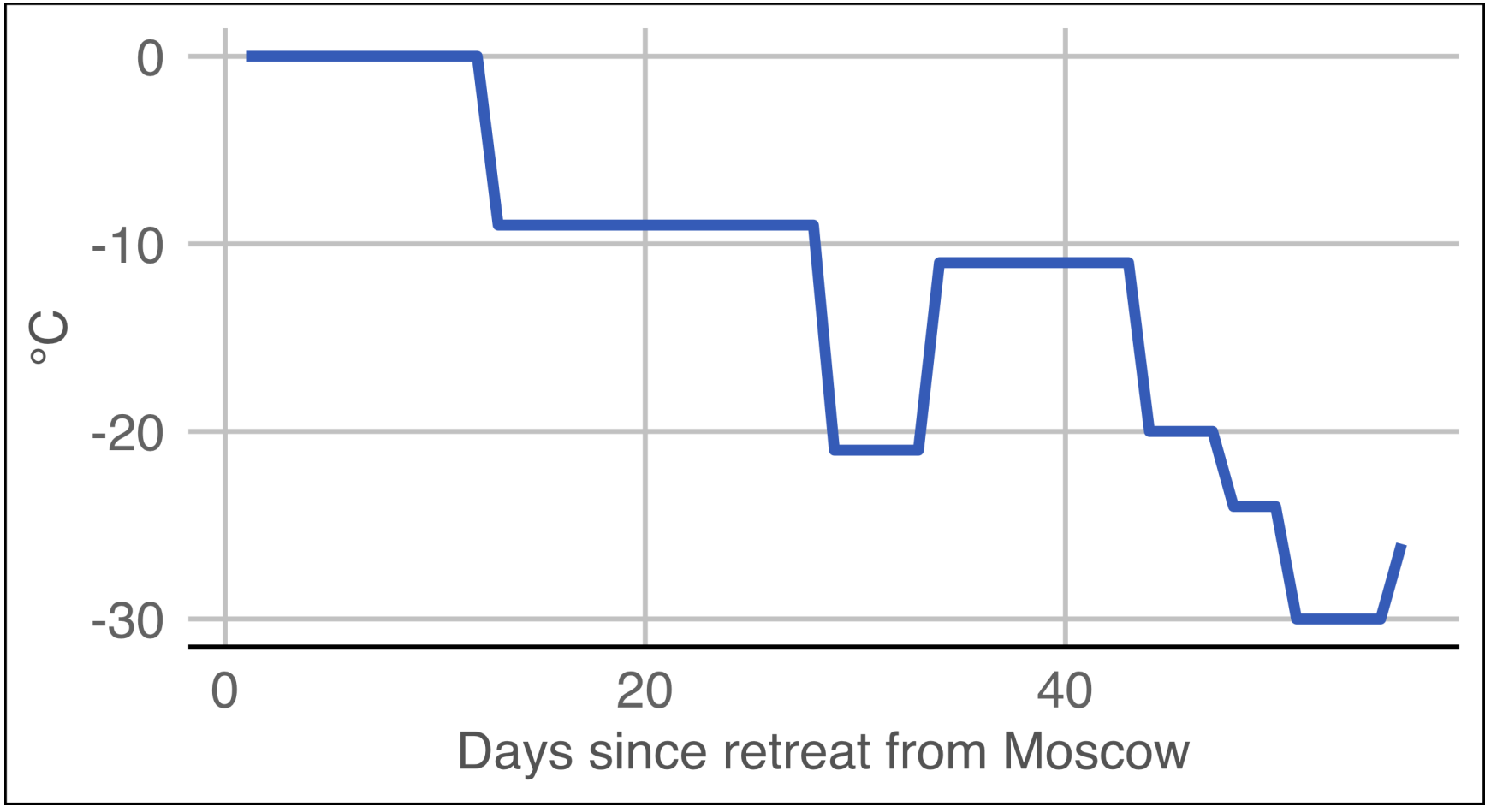


Long distance!

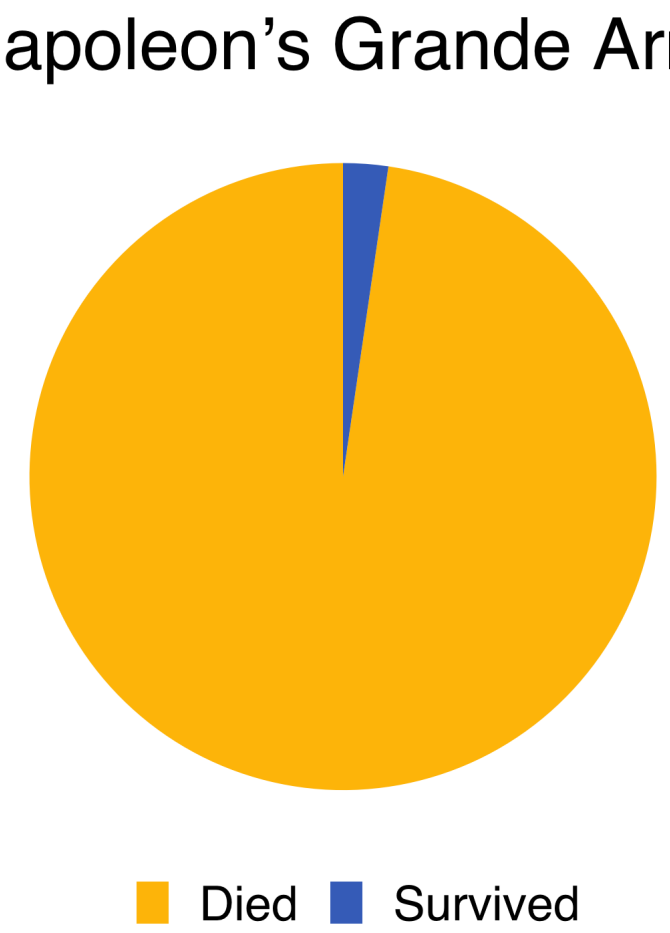


Moscow to Vilnius

Very cold!



Lots of people died!



Carte Figurative des pertes successives en hommes de l'Armée Française dans la campagne de Russie 1812 ~1813.

Dressée par M. Minard, Inspecteur Général des Ponts et Chaussées en retraite.

Paris, le 20 Novembre 1869.

Les nombres d'hommes présents sont représentés par les largeurs des zônes colorées à raison d'un millimètre pour dix mille hommes; ils sont de plus écrits en travers des zônes. Le rouge désigne les hommes qui entrent en Russie, le noir ceux qui en sortent. ___ Les renseignements qui ont servi à dresser la carte ont été puisés dans les ouvrages de M.M. Thiers, de Ségur, de Fezensac, de Chambray et le journal inédit de Jacob, pharmacien de l'Armée depuis le 28 Octobre.

Pour mieux faire juger à l'œil la diminution de l'armée, j'ai supposé que les corps du Prince Jérôme et du Maréchal Davoust qui avaient été détachés sur Minsk et Mobilow et ont rejoint vers Orscha et Witebsk, avaient toujours marché avec l'armée.

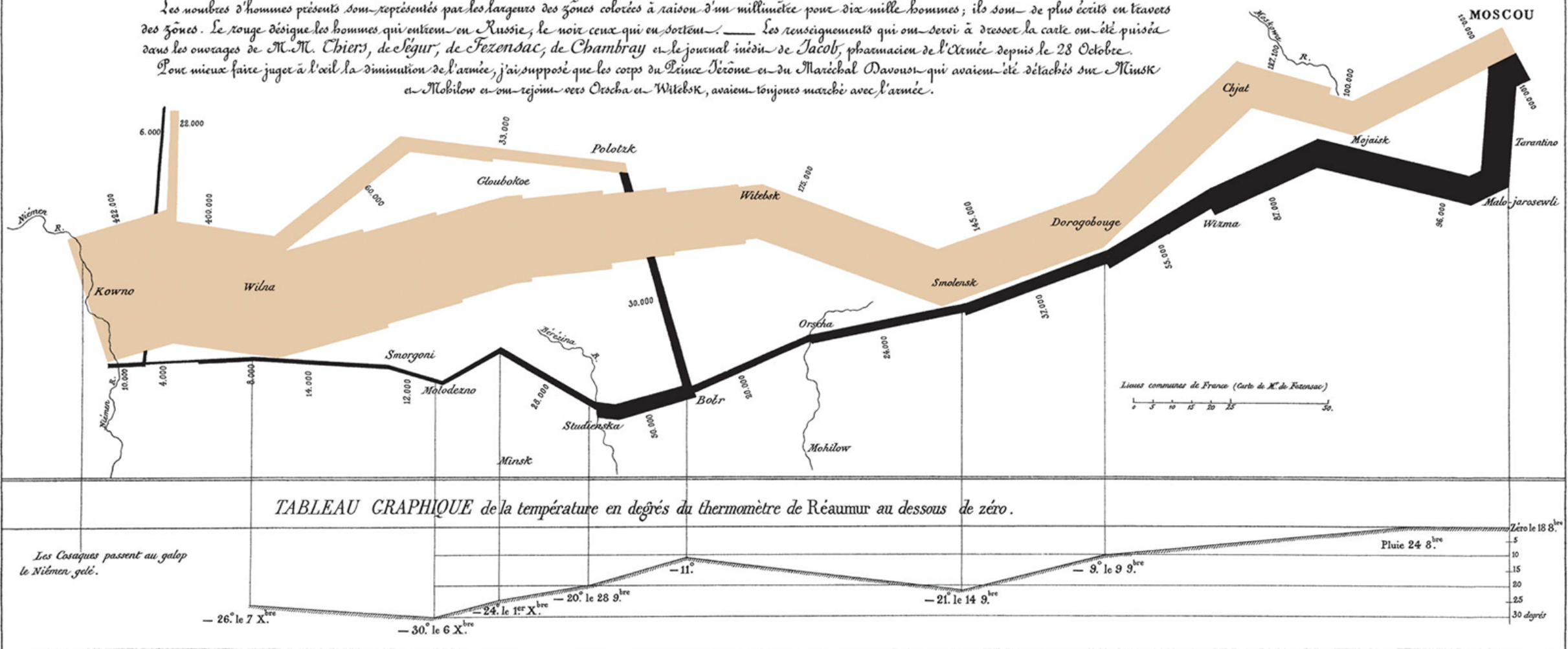
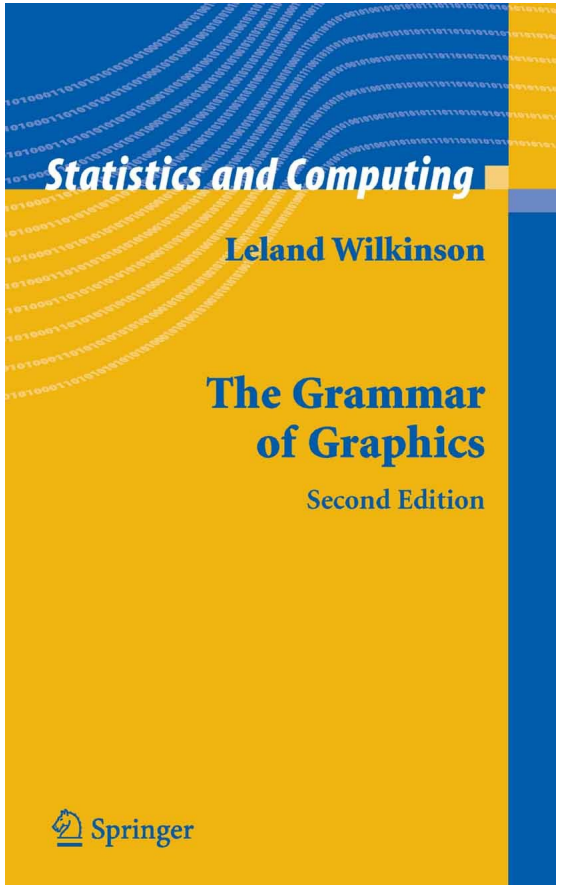


TABLEAU GRAPHIQUE de la température en degrés du thermomètre de Réaumur au dessous de zéro.

Autog. par Regnier, 8. Pas. Ste Marie St Gain à Paris.

Imp. Lith. Regnier et Dourdet.

Mapping data to aesthetics



Aesthetic

Visual property of a graph

Position, shape, color, etc.

Data

A column in a dataset

Mapping data to aesthetics

Data	Aesthetic	Graphic/Geometry
Longitude	Position (x-axis)	Point
Latitude	Position (y-axis)	Point
Army size	Size	Path
Army direction	Color	Path
Date	Position (x-axis)	Line + text
Temperature	Position (y-axis)	Line + text

Mapping data to aesthetics

Data	**`aes()`**	**`geom`**
Longitude	`x`	`geom_point()`
Latitude	`y`	`geom_point()`
Army size	`size`	`geom_path()`
Army direction	`color`	`geom_path()`
Date	`x`	`geom_line() + geom_text()`
Temperature	`y`	`geom_line() + geom_text()`

`ggplot()` template

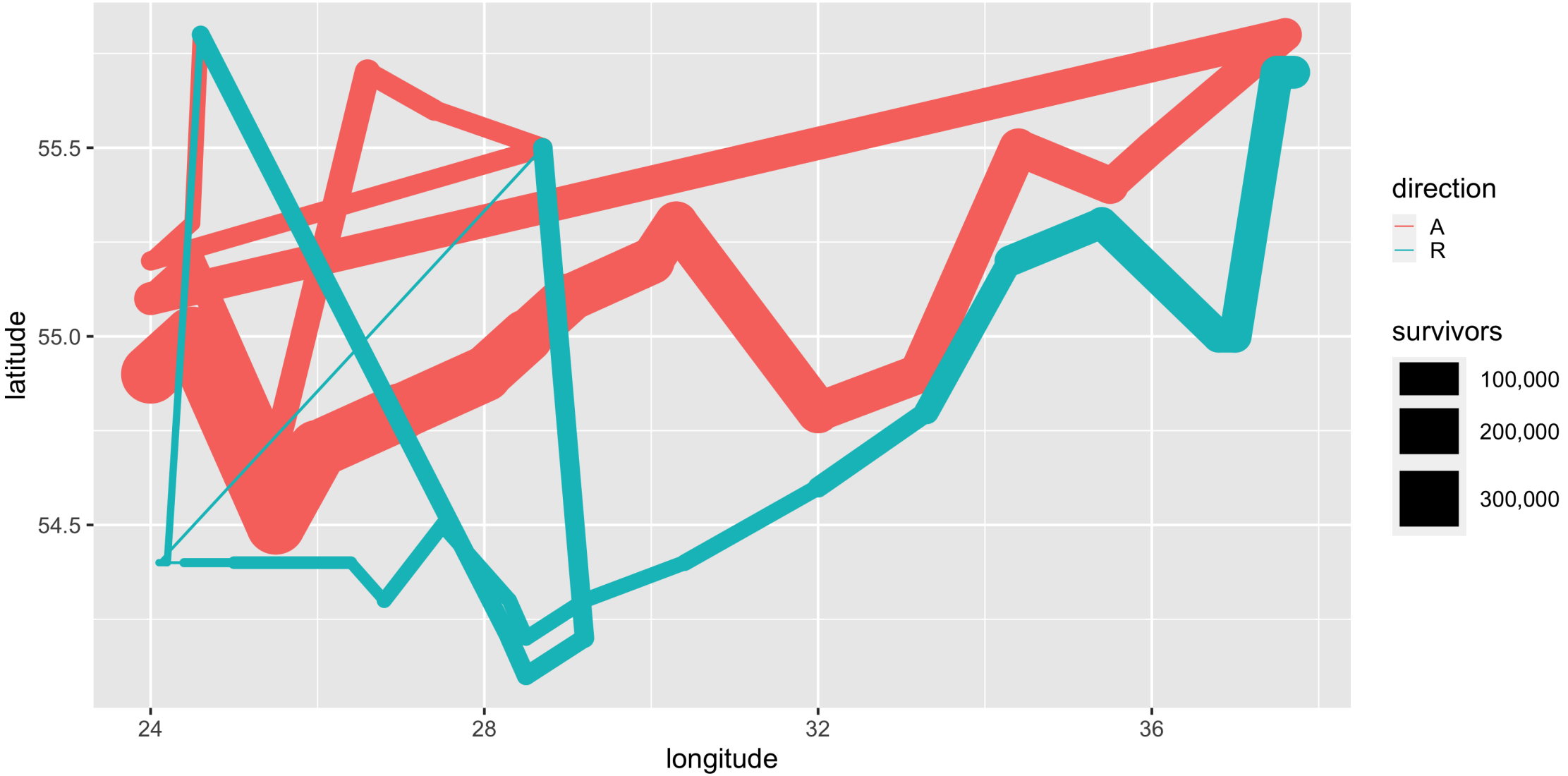
```
ggplot(data = DATA) +
  GEOM_FUNCTION(mapping = aes(AESTHETIC MAPPINGS))
```

```
ggplot(data = troops) +
  geom_path(mapping = aes(x = longitude,
                          y = latitude,
                          color = direction,
                          size = survivors))
```

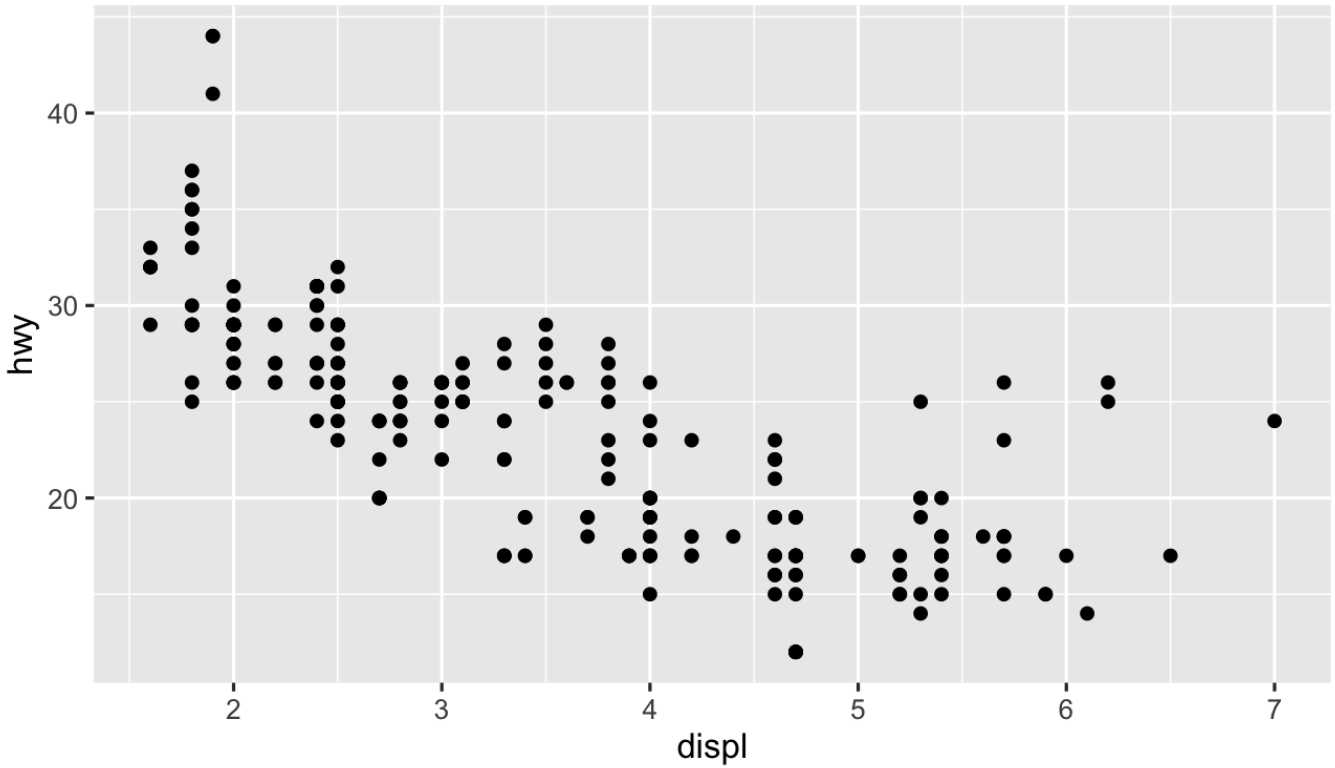
This is a dataset named `troops`:

longitude	latitude	direction	survivors
24	54.9	A	340000
24.5	55	A	340000
…	…	…	…

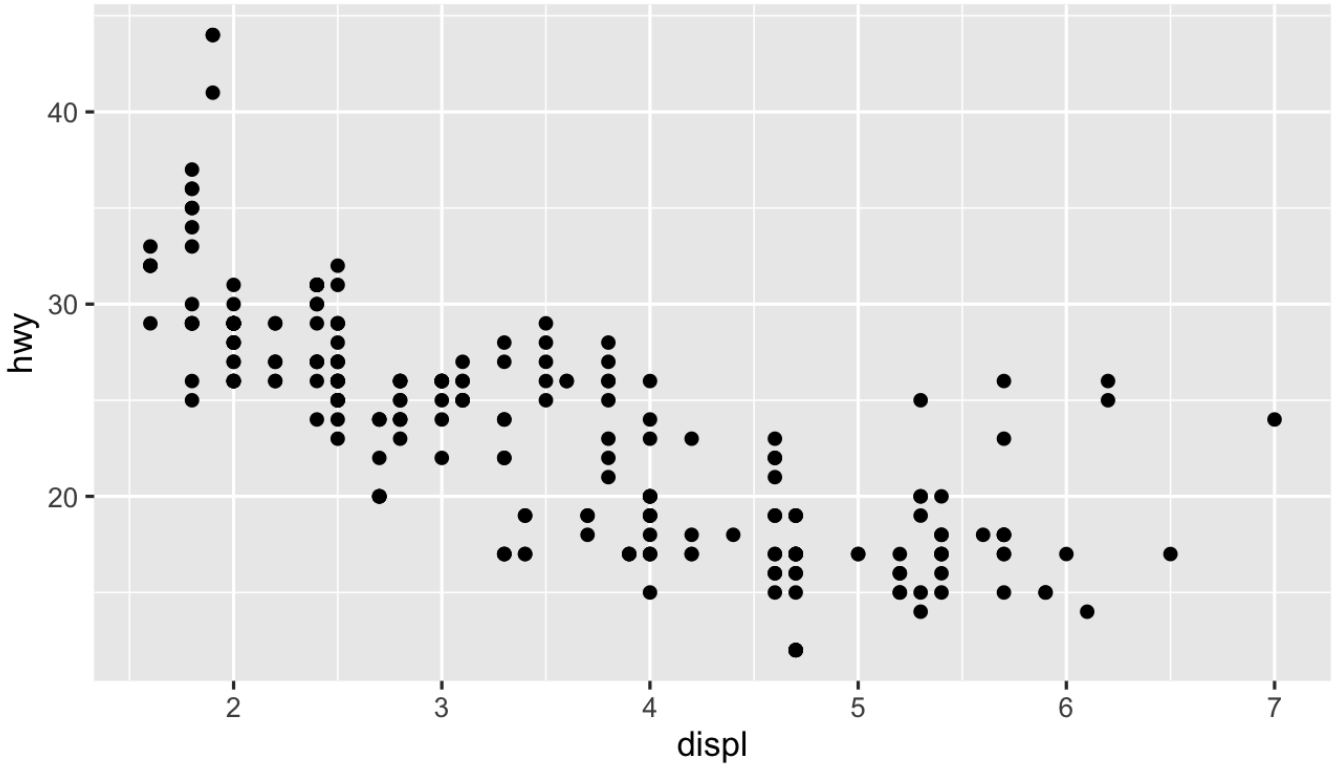
```
ggplot(data = troops) +
  geom_path(mapping = aes(x = longitude,
                          y = latitude,
                          color = direction,
                          size = survivors))
```


55.5
55.0
54.5
latitude
24
28
32
36
longitude
direction
A
R
survivors
100,000
200,000
300,000

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

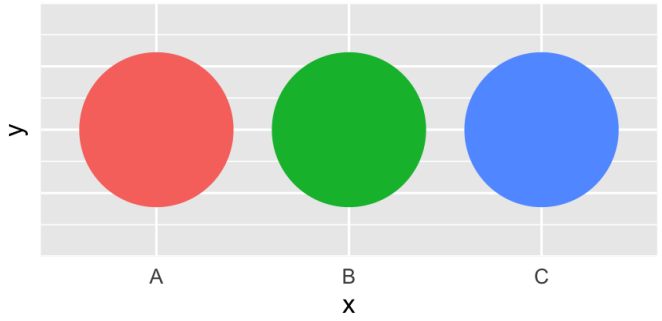


Heavy cars with better mileage?

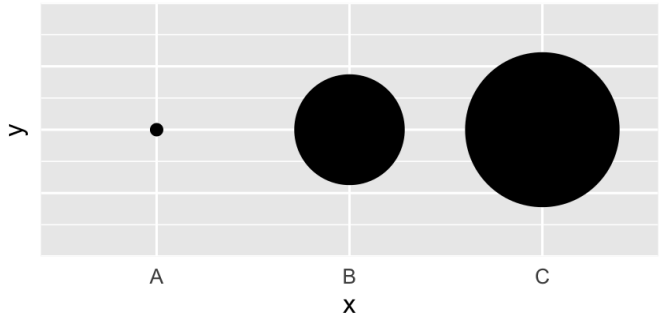


Aesthetics

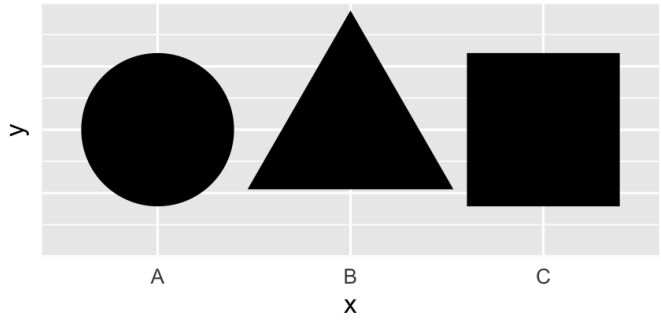
`color` (discrete)



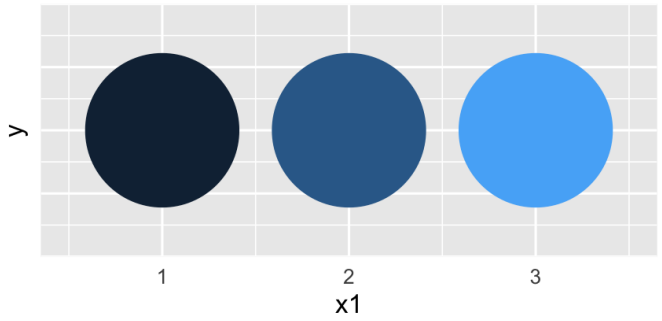
`size`



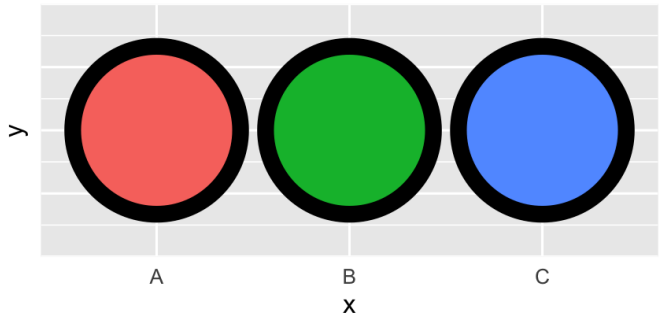
`shape`



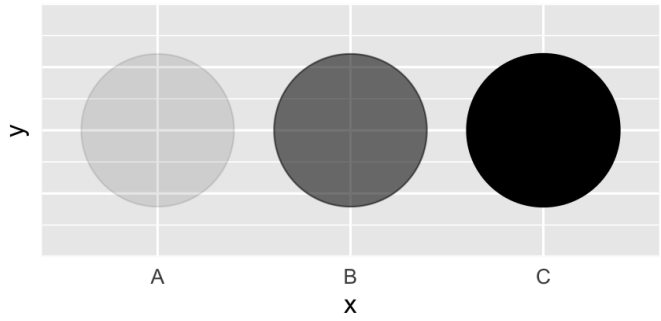
`color` (continuous)



`fill`



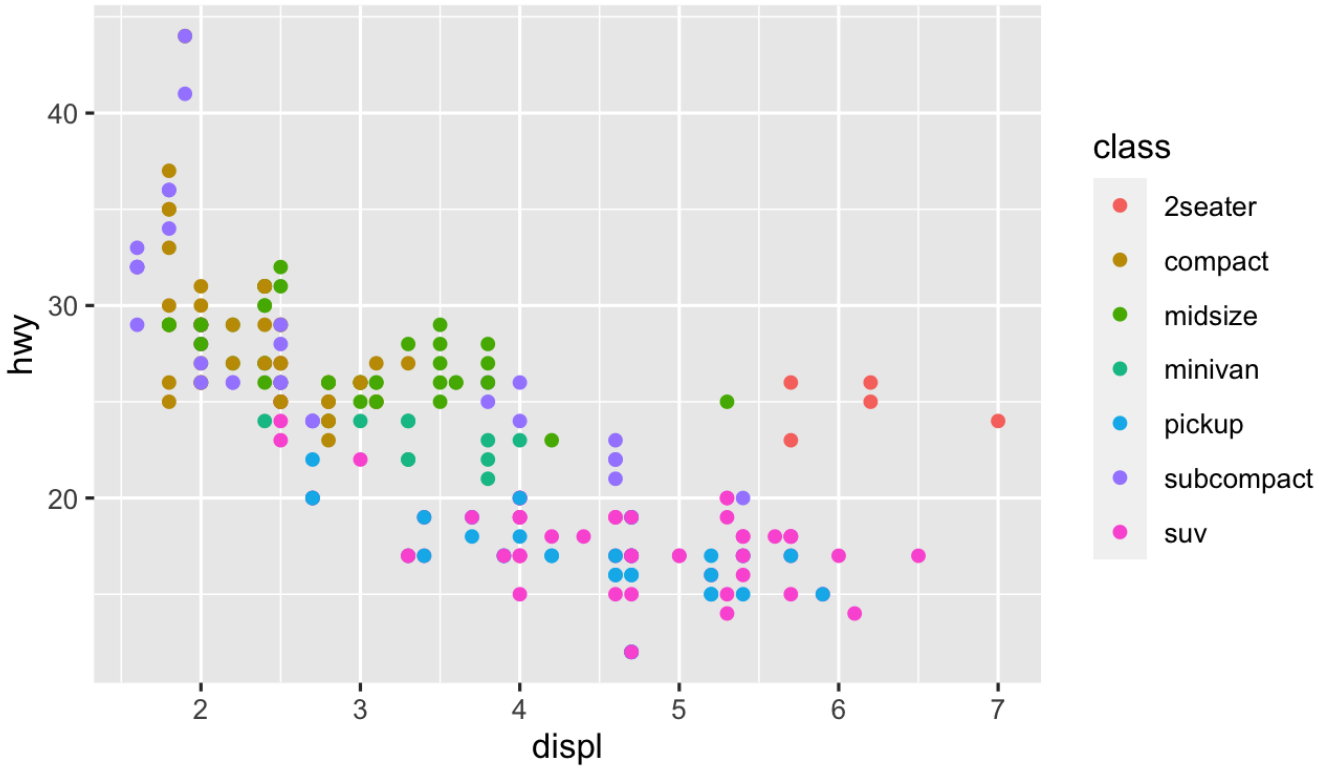
`alpha`



Mapping columns to aesthetics

```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, size = class))
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, shape = class))
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, alpha = class))
```

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



ggplot playground

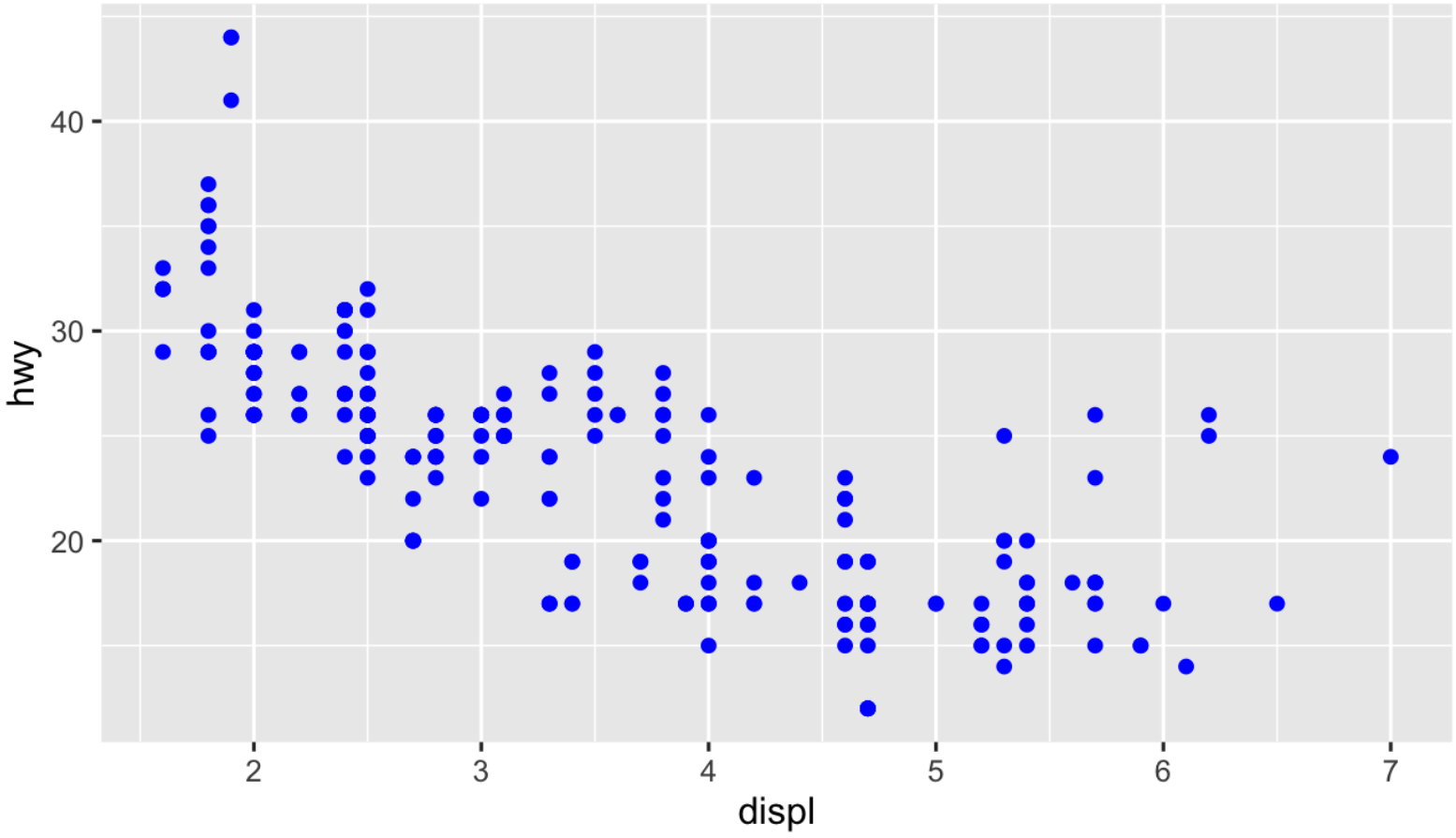
Add color, size, alpha, and shape aesthetics to your graph.

Experiment!

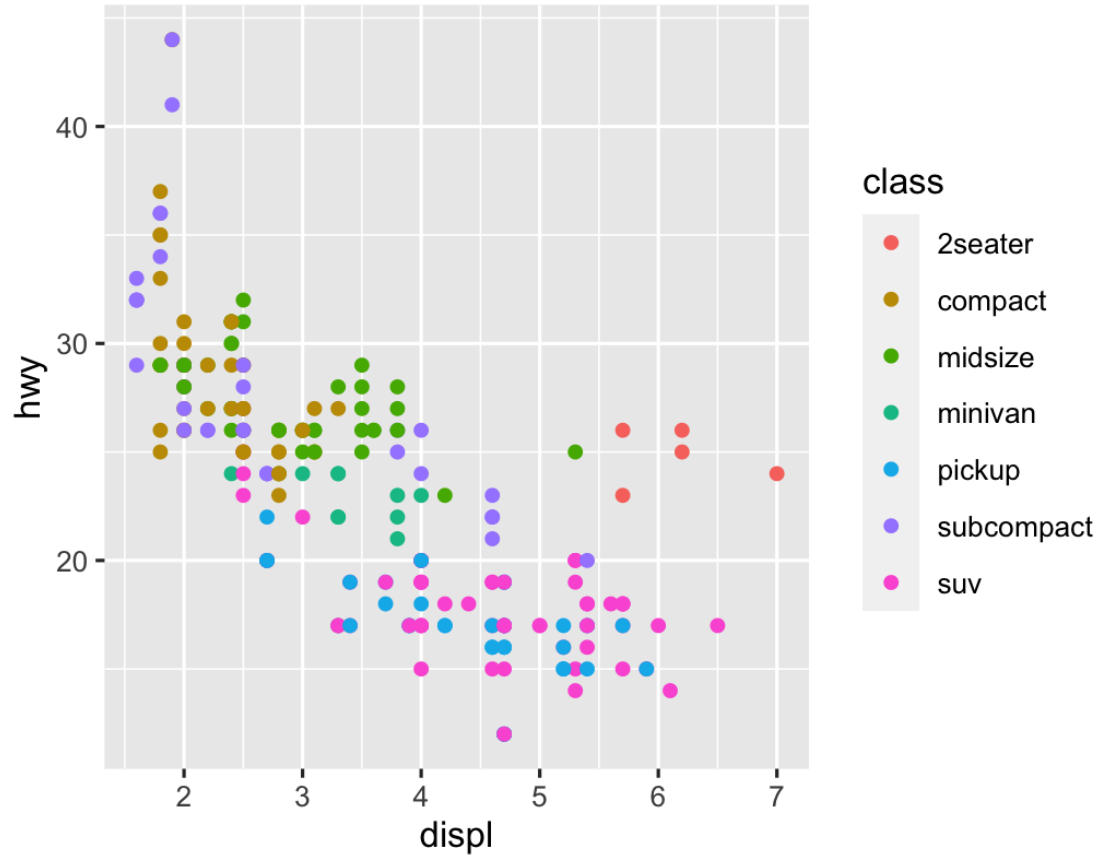
Do different things happen when you map aesthetics to discrete and continuous variables?

What happens when you use more than one aesthetic?

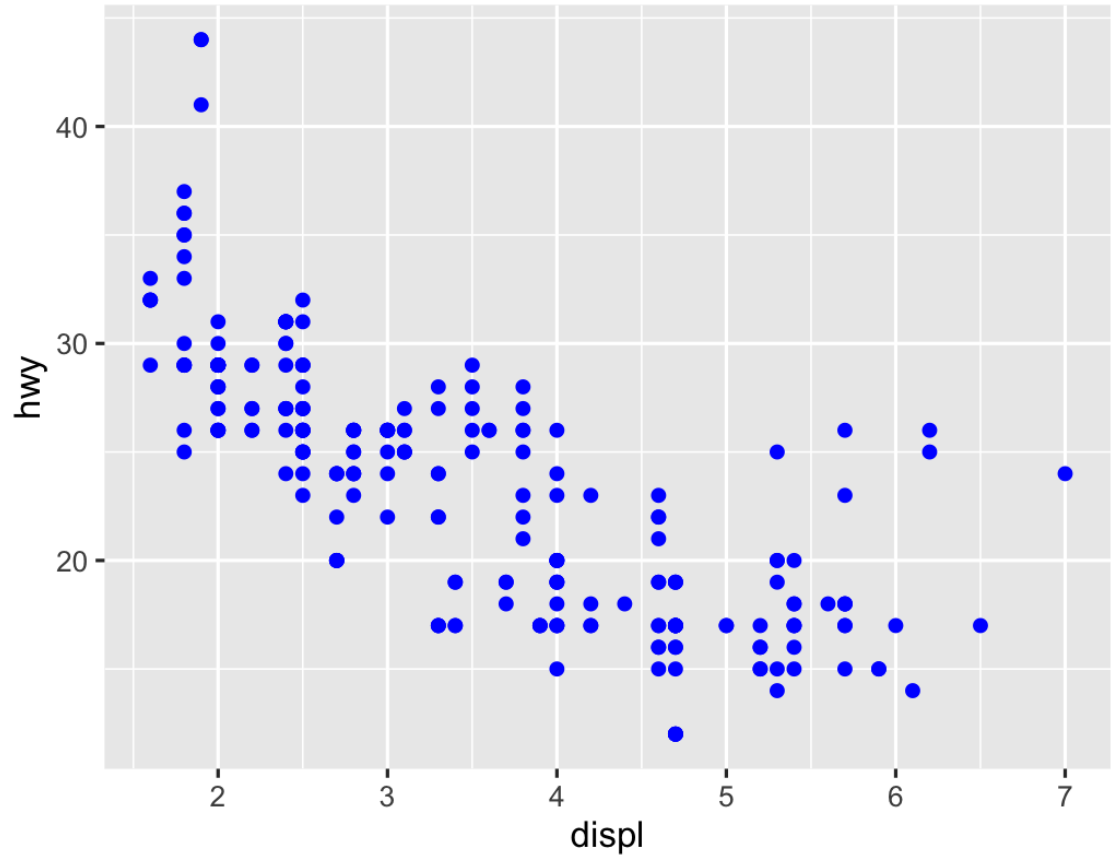
How would you make this plot?



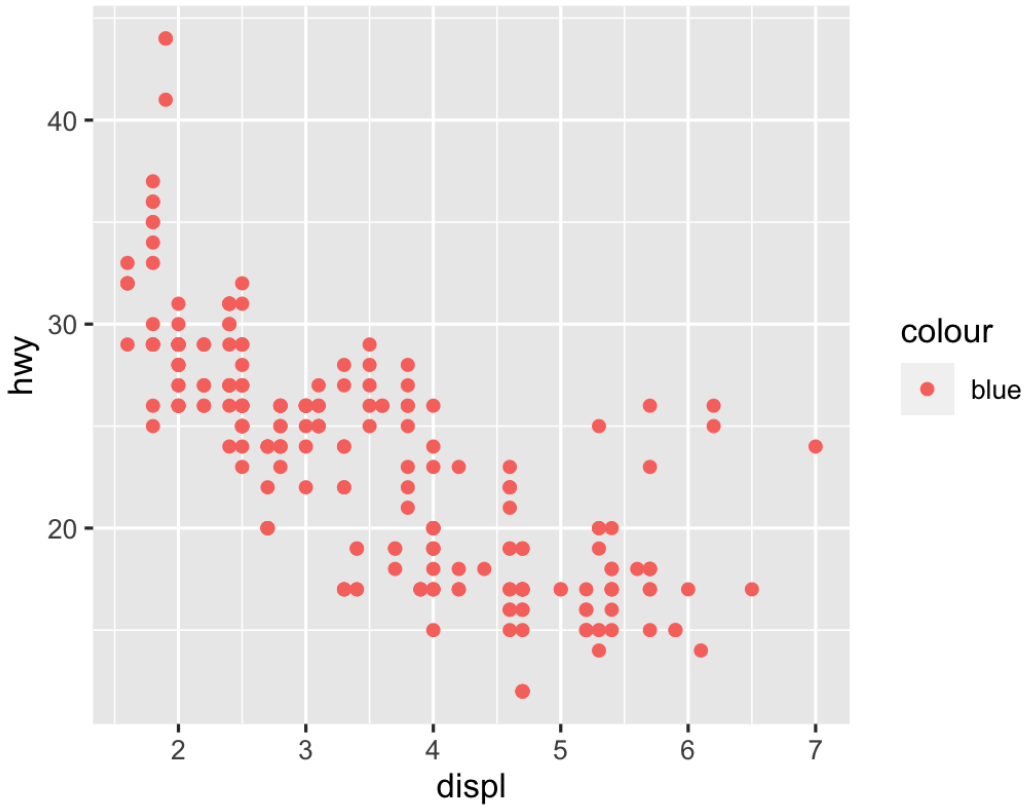
```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy,
                 color = class))
```



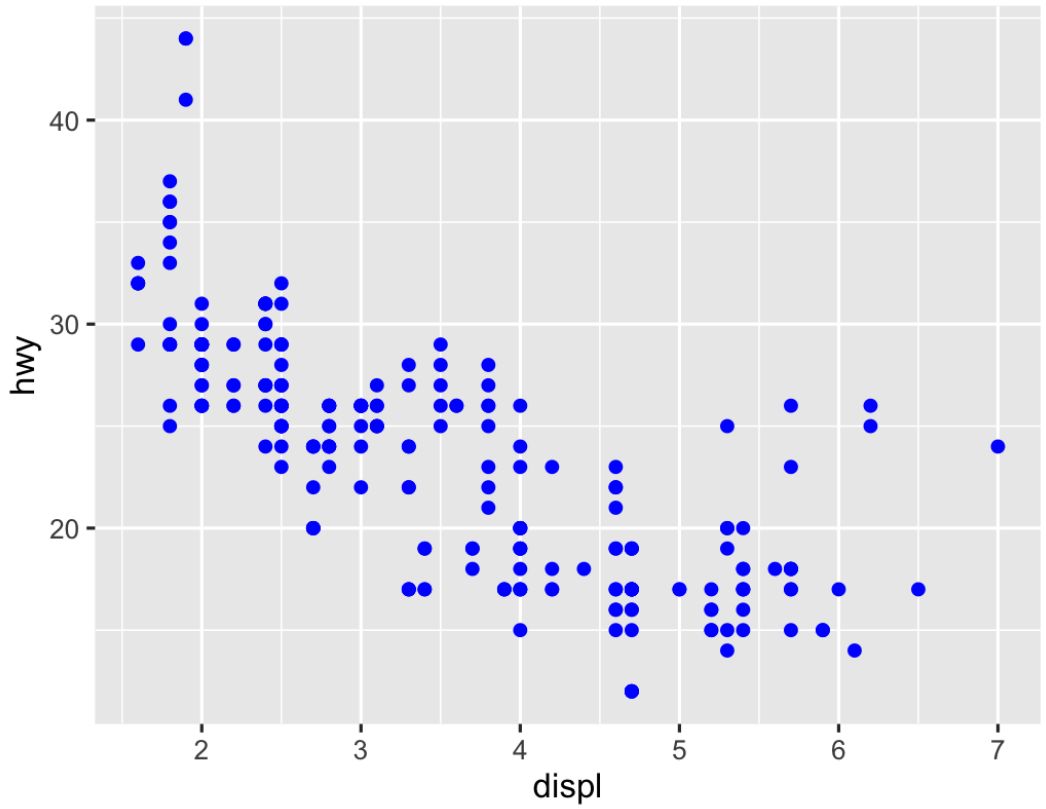
```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy),
             color = "blue")
```



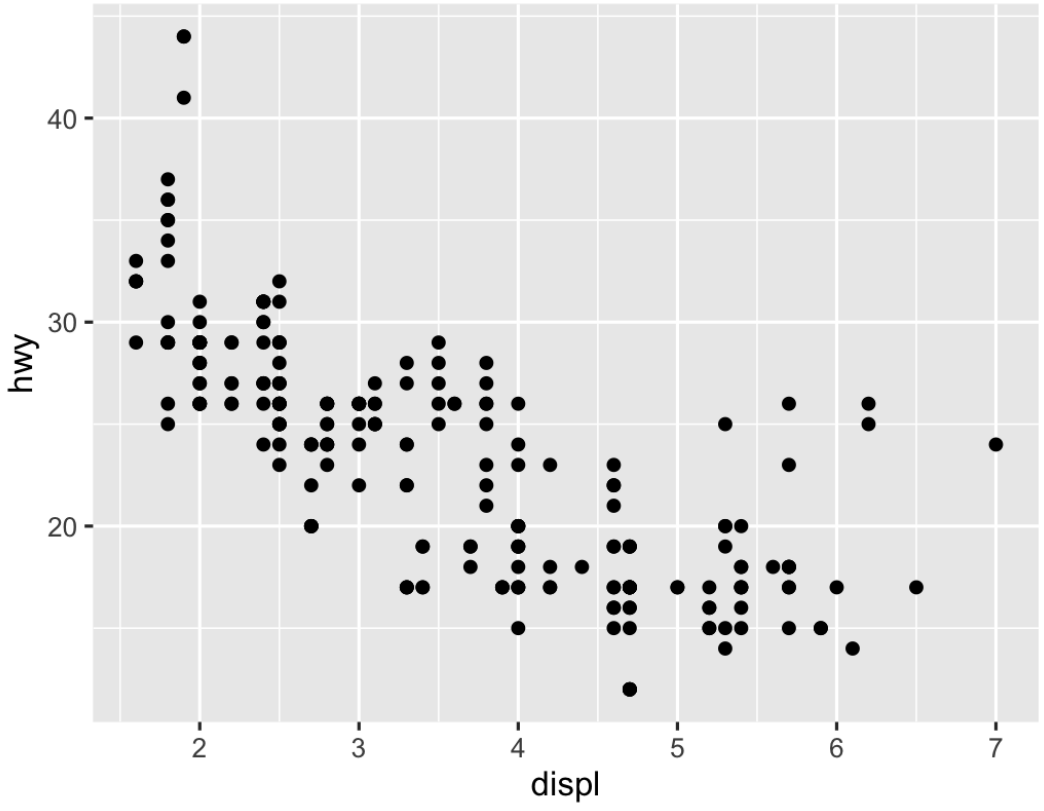
```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy,
                 color = "blue"))
```

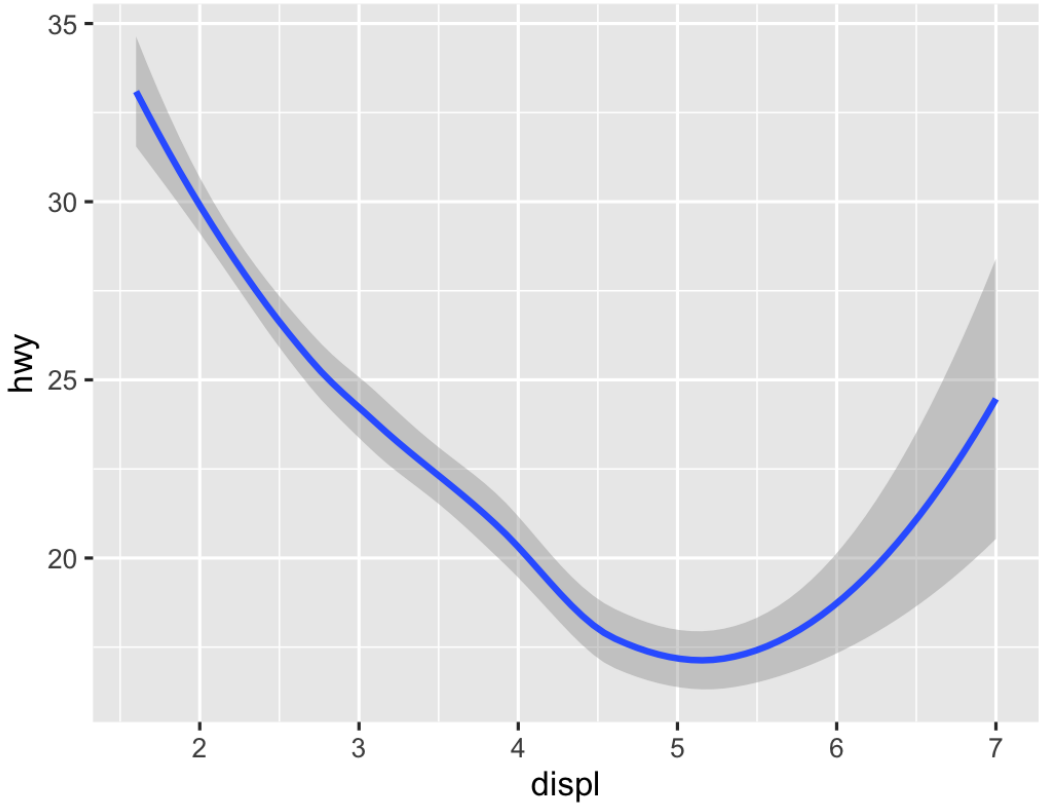



```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy)
             color = "blue")
```



What's the same? What's different?

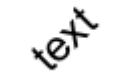




Geoms

```
ggplot(data = DATA) +
  GEOM_FUNCTION(mapping = aes(AESTHETIC MAPPINGS))
```

Possible geoms

	Example geom	What it makes
	`geom_col()`	Bar charts
	`geom_text()`	Text
	`geom_point()`	Points
	`geom_boxplot()`	Boxplots
	`geom_sf()`	Maps

Possible geoms

There are dozens of possible geoms!

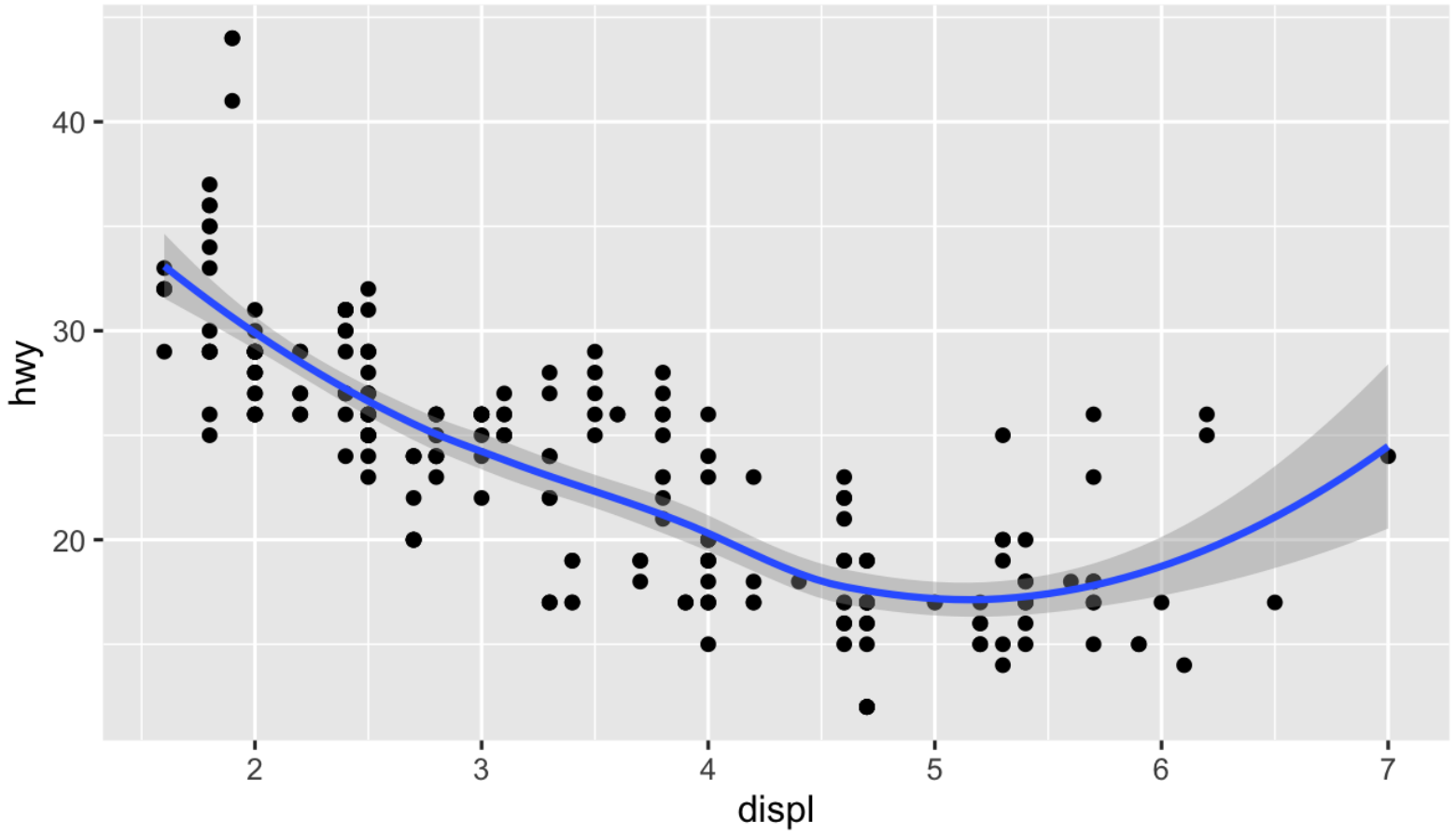
See the ggplot2 documentation for complete examples of all the different geom layers

Also see the ggplot cheatsheet

Complex graphs!

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

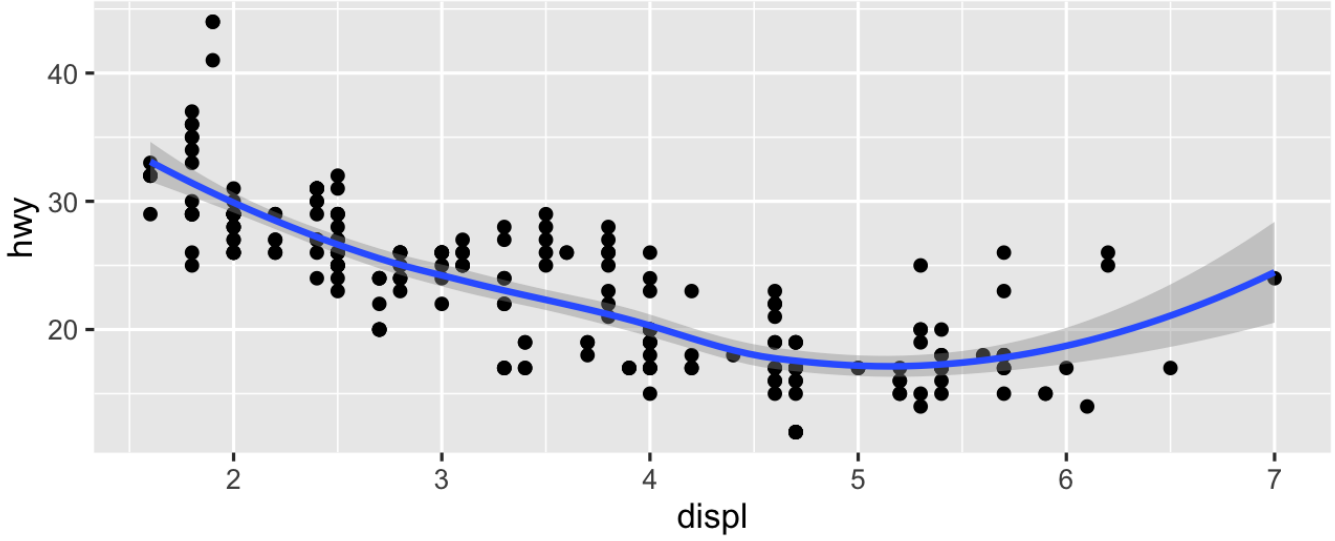
Complex graphs!



Global vs. local

Any aesthetics in `ggplot()` will show up in all `geom_` layers

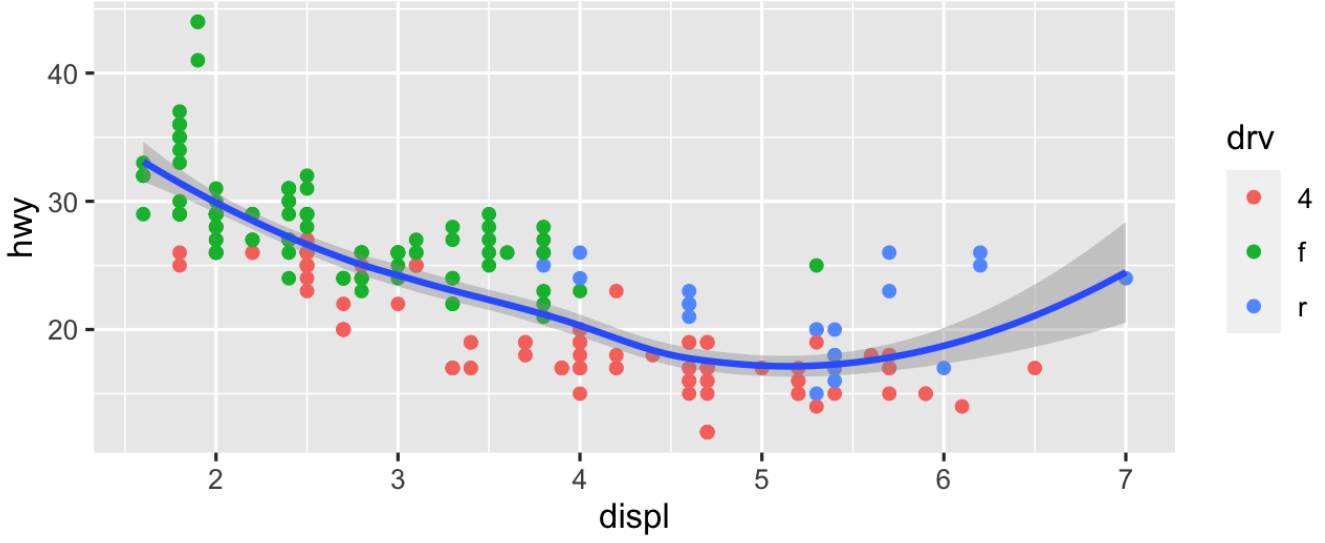
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```



Global vs. local

Any aesthetics in `geom_` layers only apply to that layer

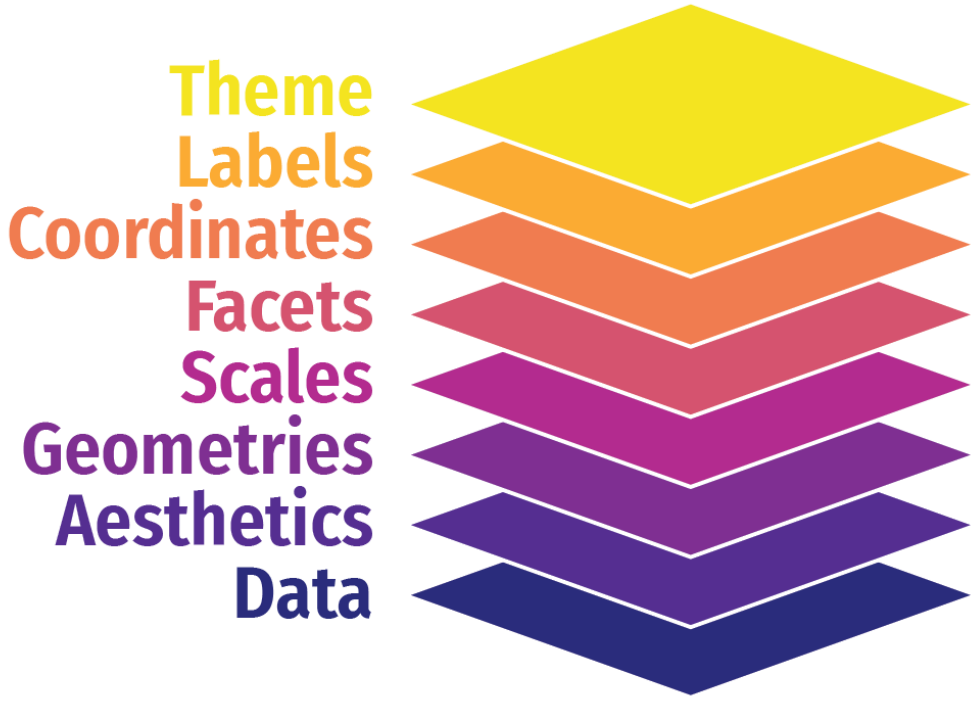
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = drv)) +
  geom_smooth()
```



So much more!

There are many other layers we can use to make and enhance graphs!

We sequentially add layers onto the foundational `ggplot()` plot to create complex figures

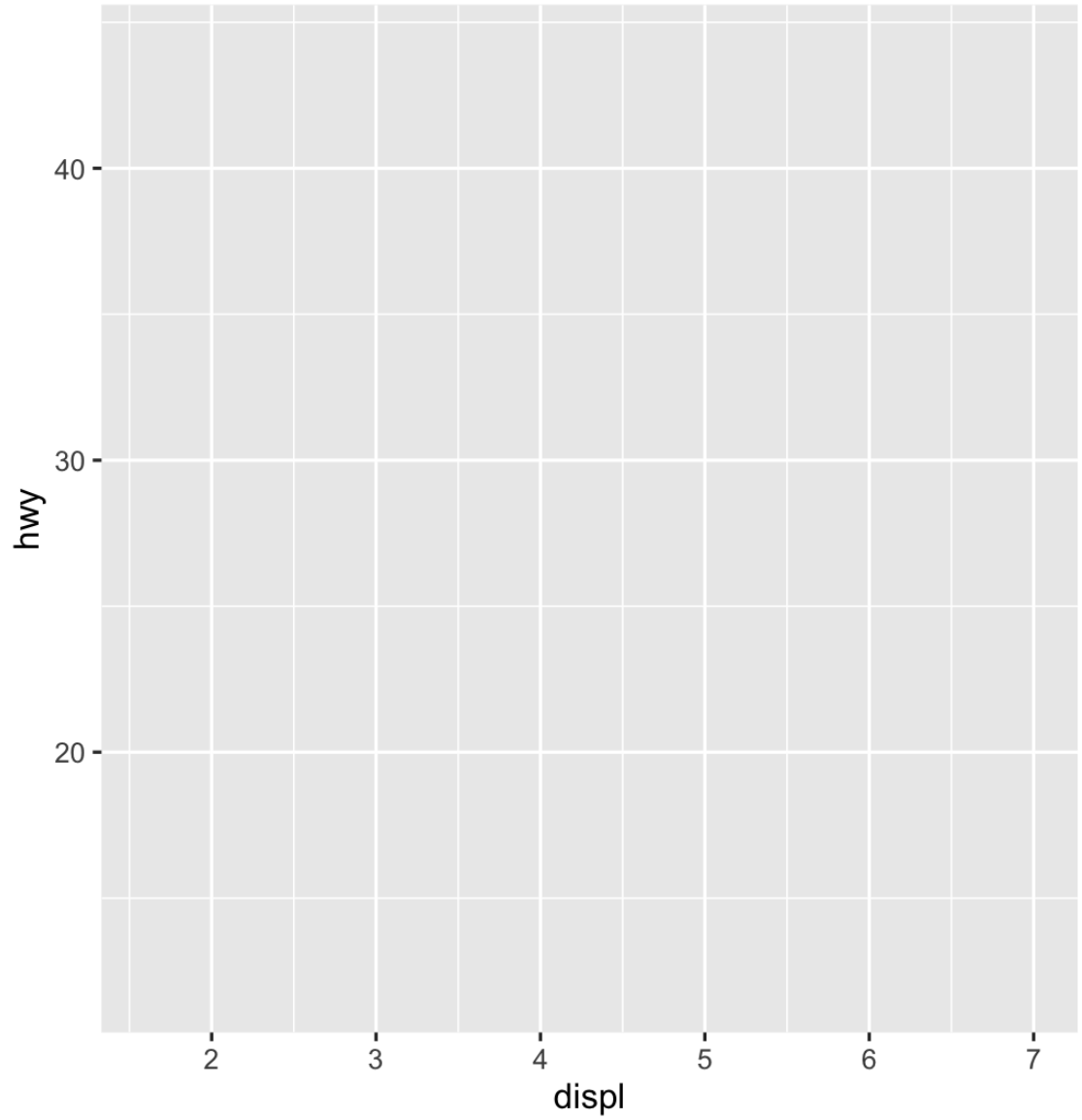


Putting it all together

We can build a plot sequentially to see how each grammatical layer changes the appearance

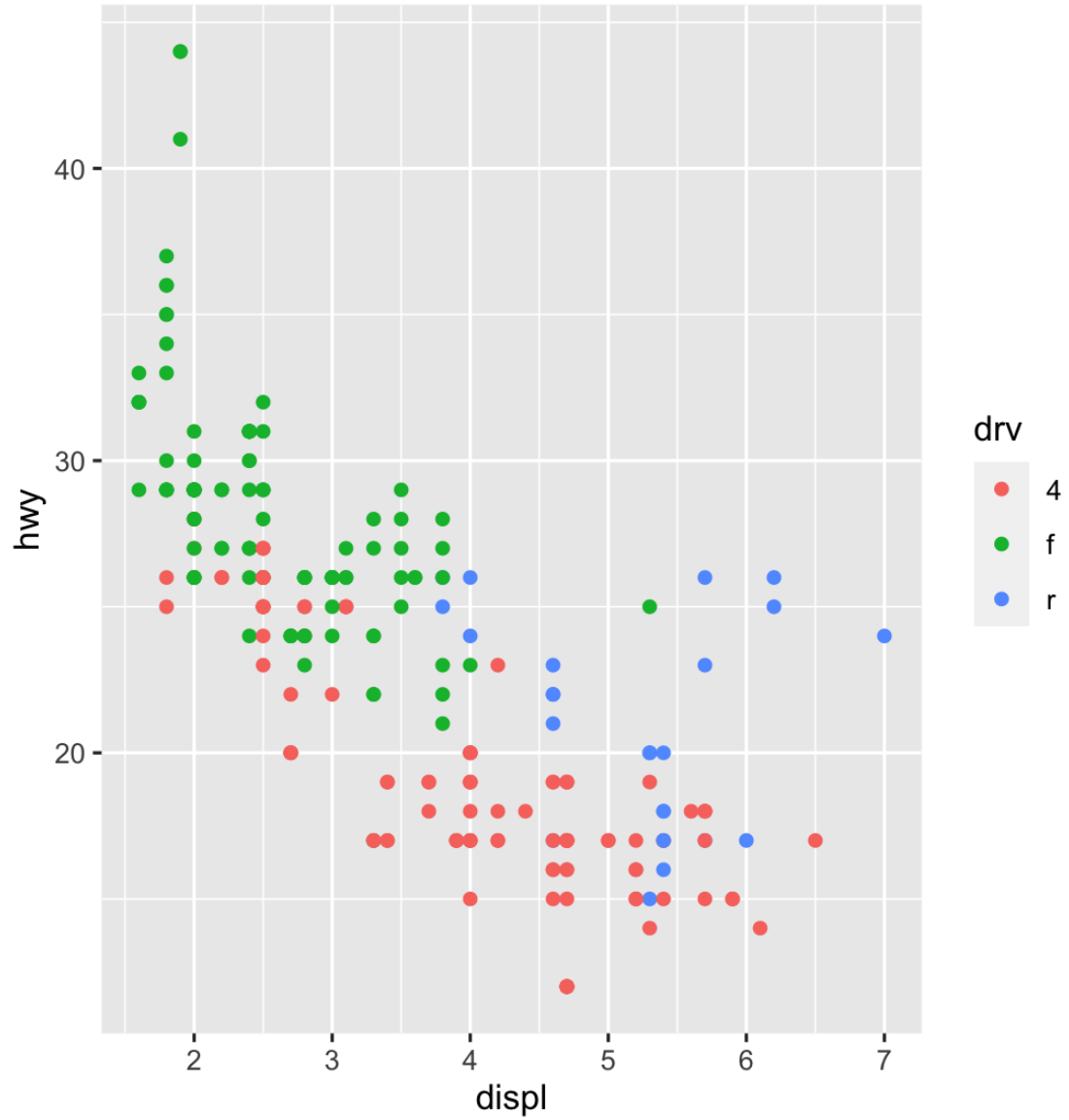
Start with data and aesthetics

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



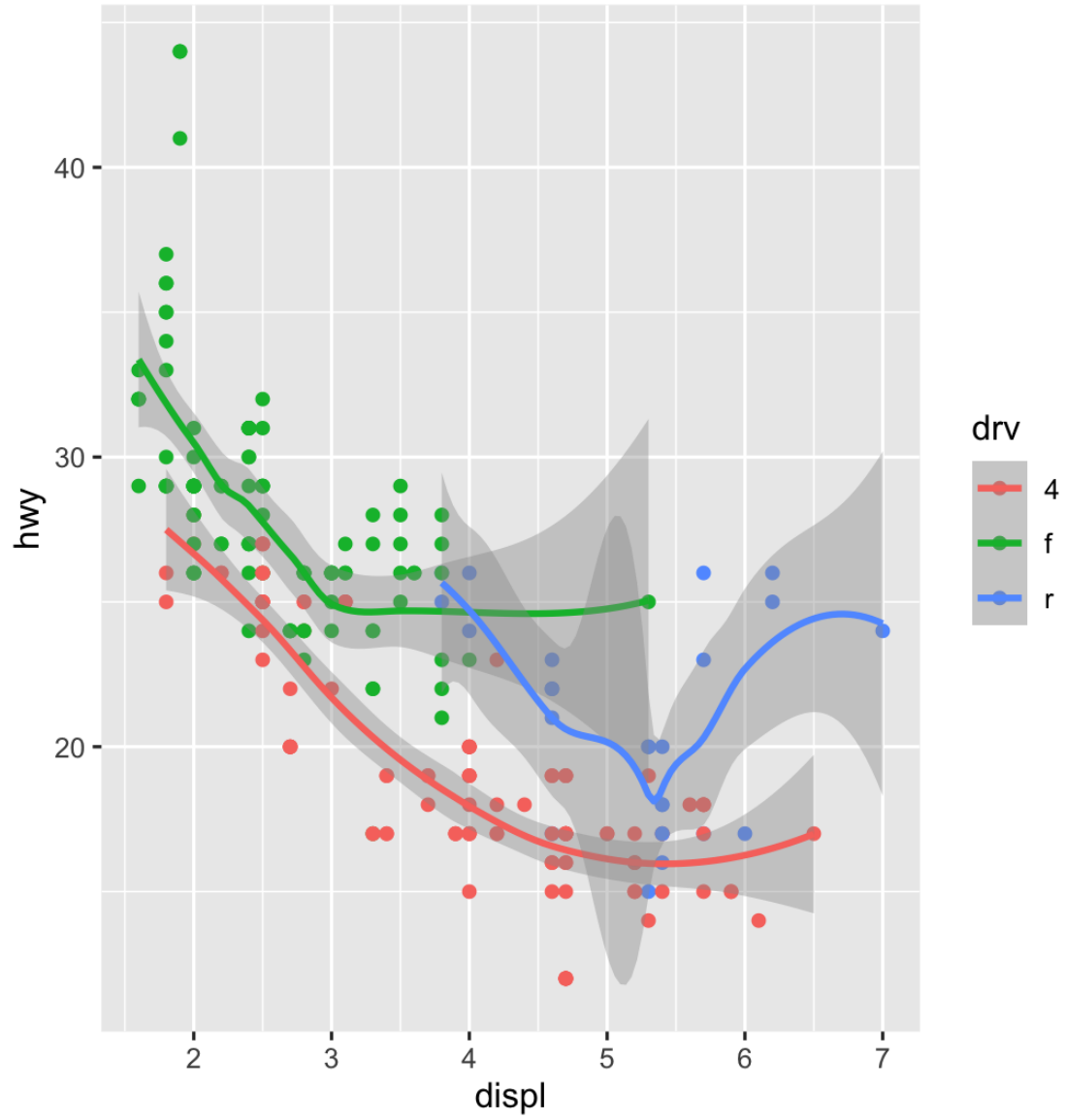
Add a point geom

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



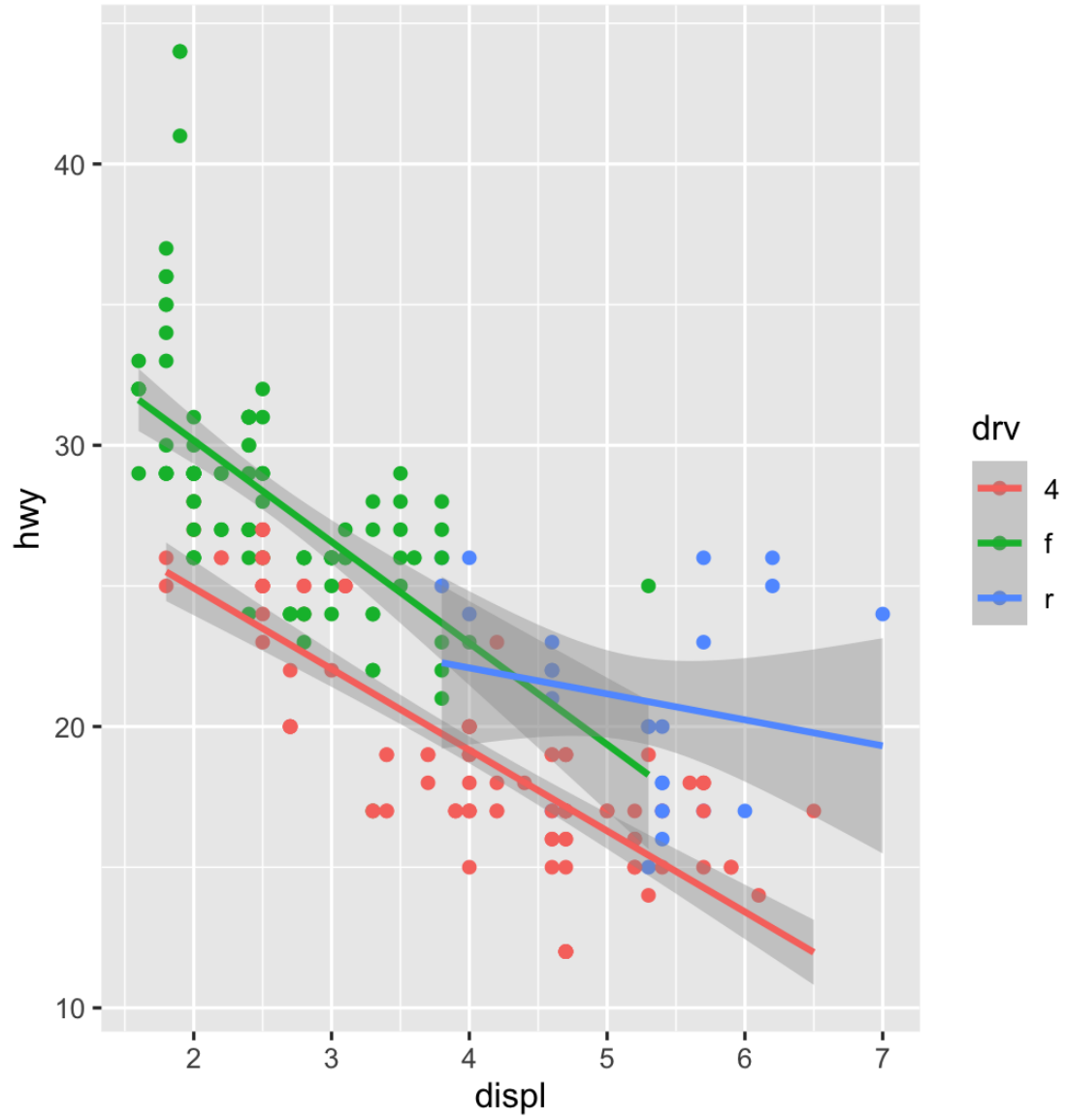
Add a smooth geom

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



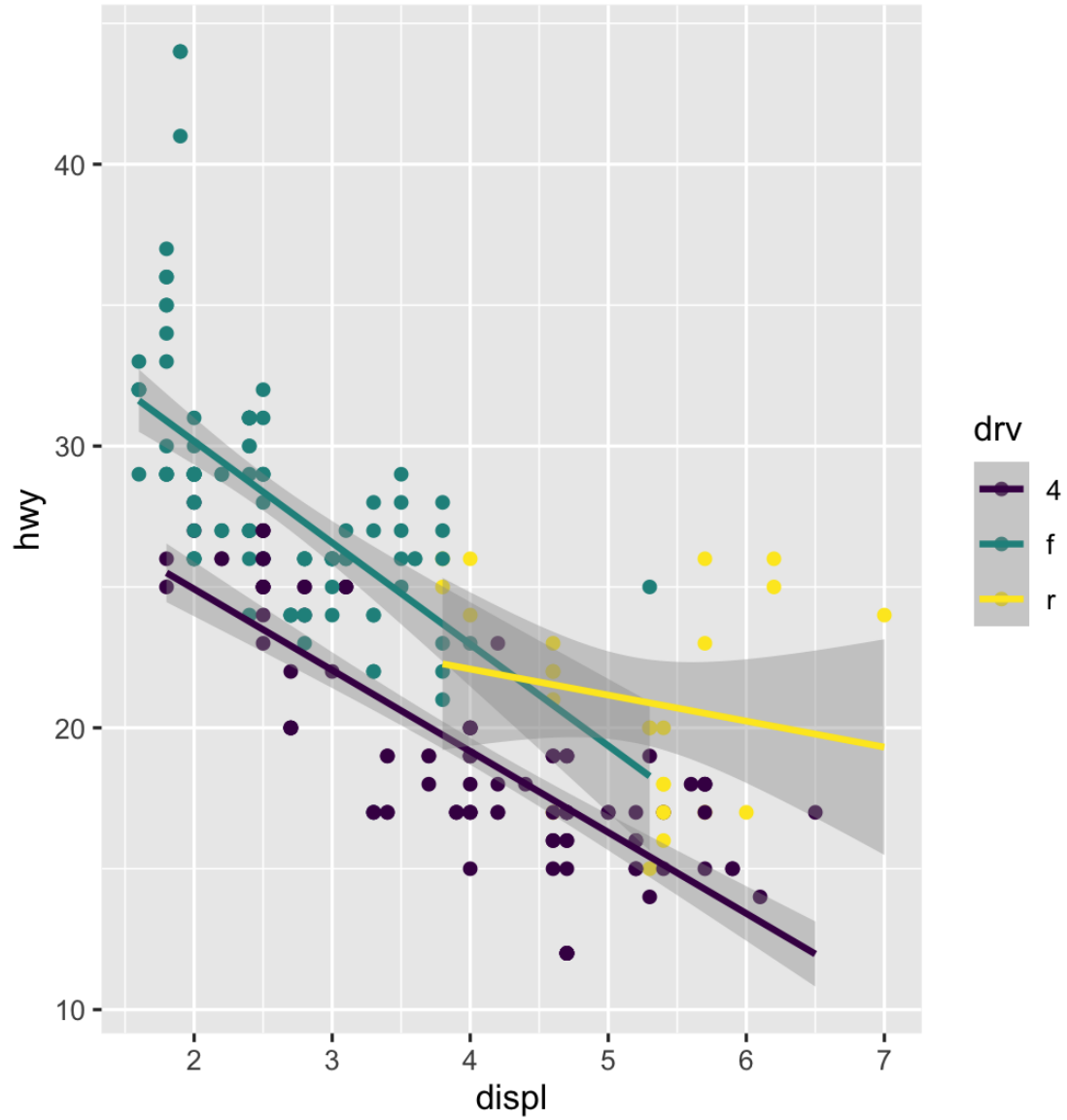
Make it straight

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



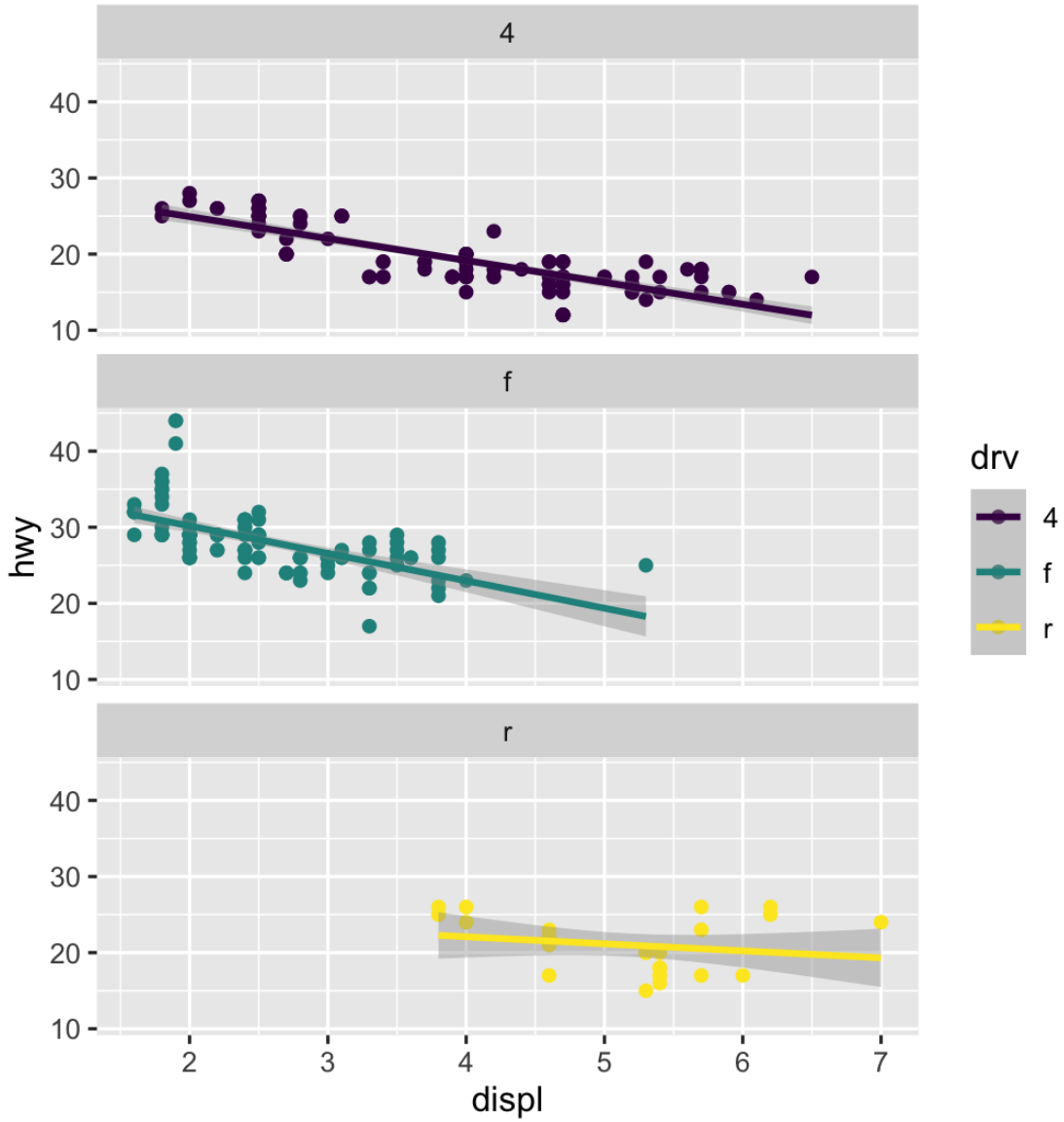
Use a viridis color scale

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



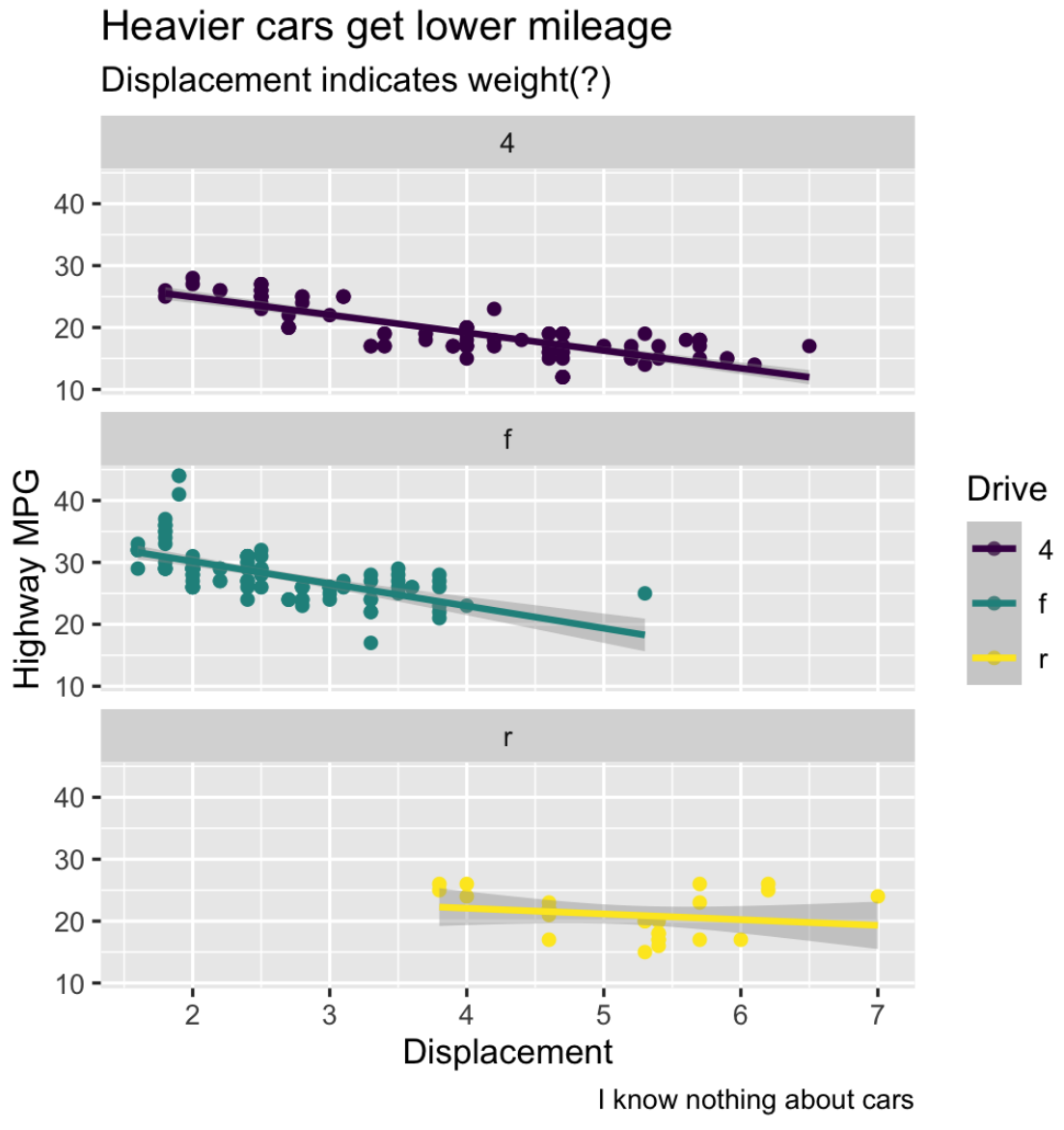
Facet by drive

```
ggplot(data = mpg,
       mapping = aes(x = displ,
                     y = hwy,
                     color = drv)) +
  geom_point() +
  geom_smooth(method = "lm") +
  scale_color_viridis_d() +
  facet_wrap(vars(drv), ncol = 1)
```



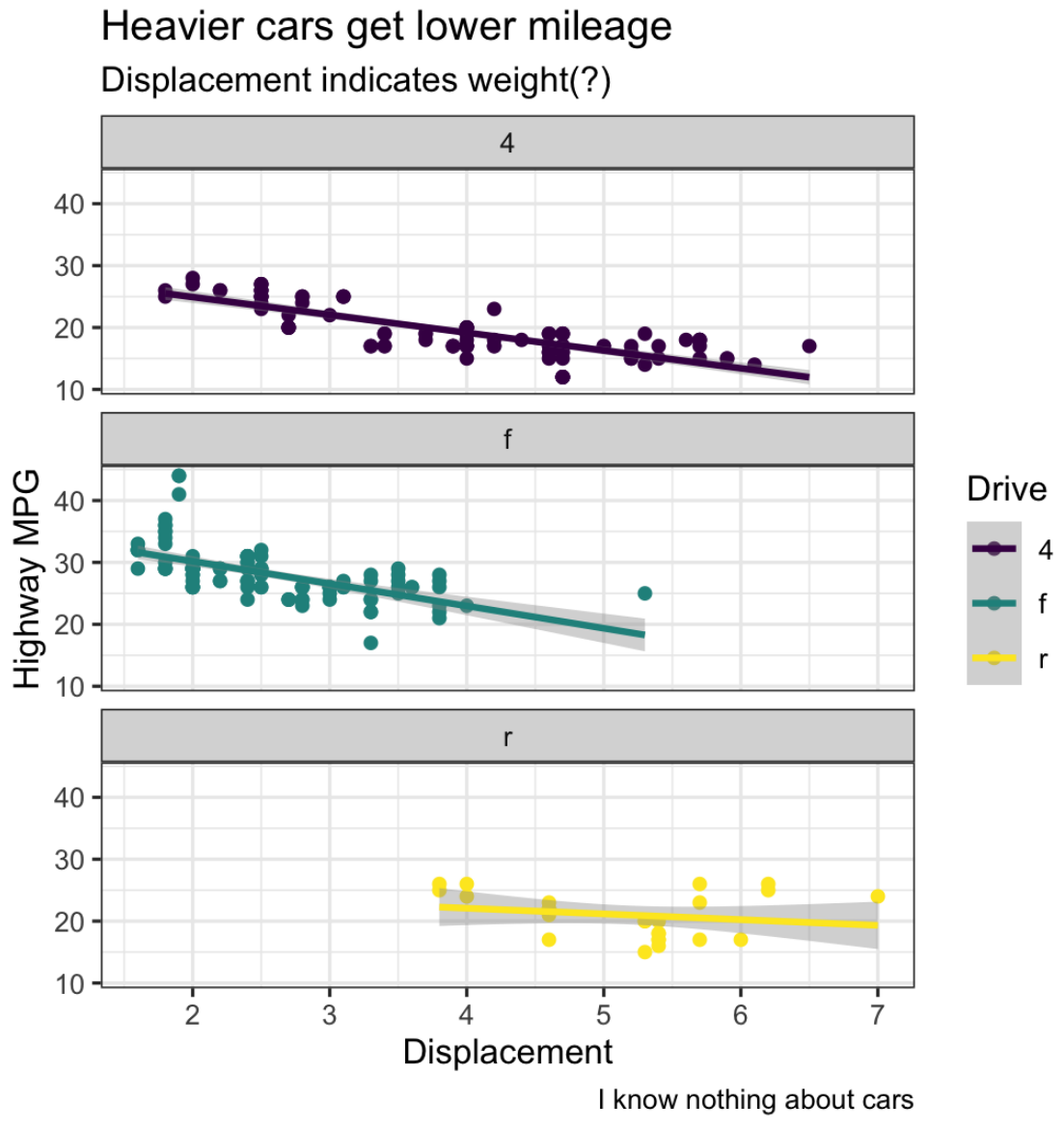
Add labels

```
ggplot(data = mpg,
       mapping = aes(x = displ,
                     y = hwy,
                     color = drv)) +
  geom_point() +
  geom_smooth(method = "lm") +
  scale_color_viridis_d() +
  facet_wrap(vars(drv), ncol = 1) +
  labs(x = "Displacement", y = "Highway MPG",
       color = "Drive",
       title = "Heavier cars get lower mileag
       subtitle = "Displacement indicates we
       caption = "I know nothing about cars")
```



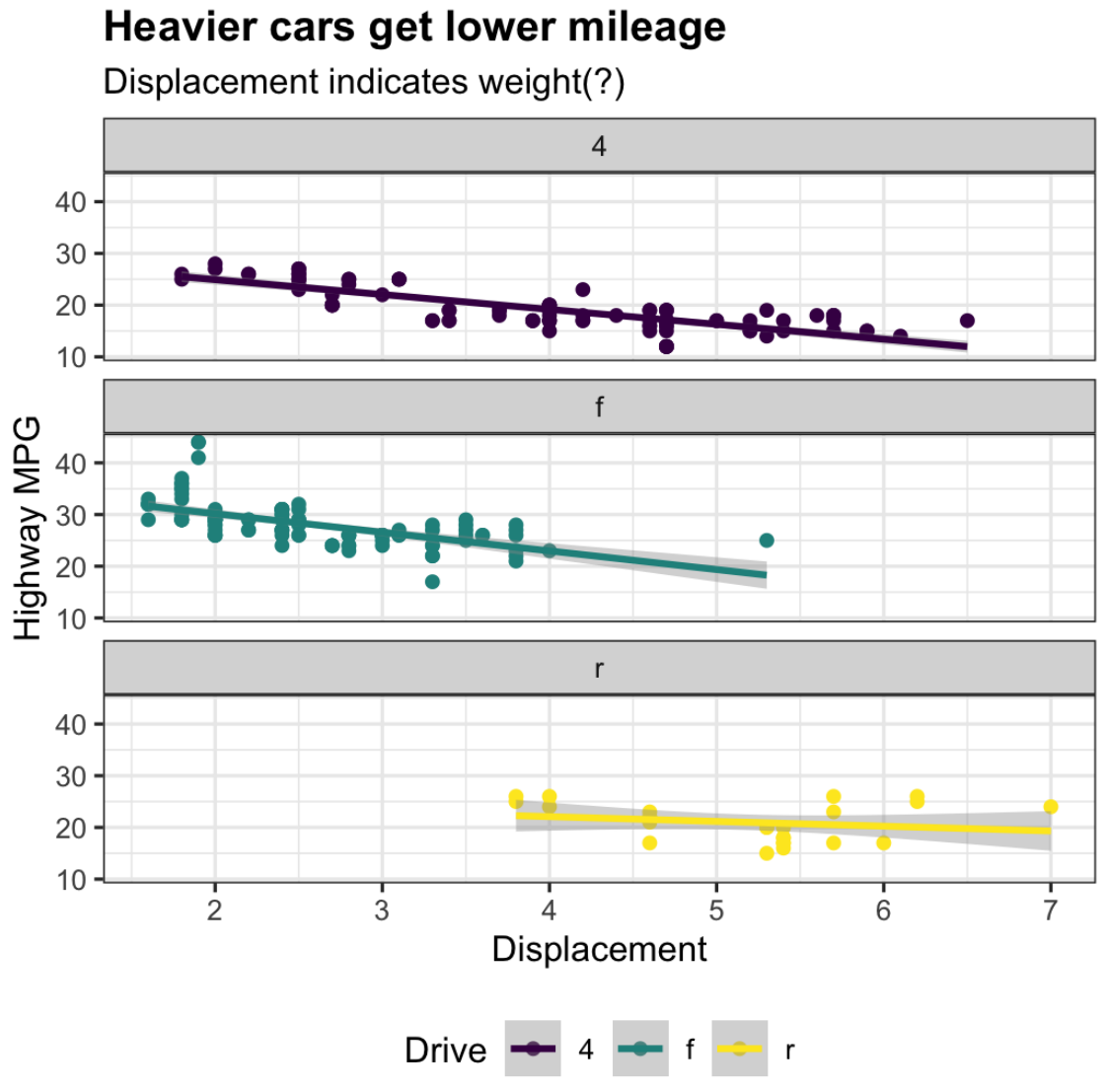
Add a theme

```
ggplot(data = mpg,
       mapping = aes(x = displ,
                     y = hwy,
                     color = drv)) +
  geom_point() +
  geom_smooth(method = "lm") +
  scale_color_viridis_d() +
  facet_wrap(vars(drv), ncol = 1) +
  labs(x = "Displacement", y = "Highway MPG",
       color = "Drive",
       title = "Heavier cars get lower mileag
       subtitle = "Displacement indicates wei
       caption = "I know nothing about cars")
  theme_bw()
```



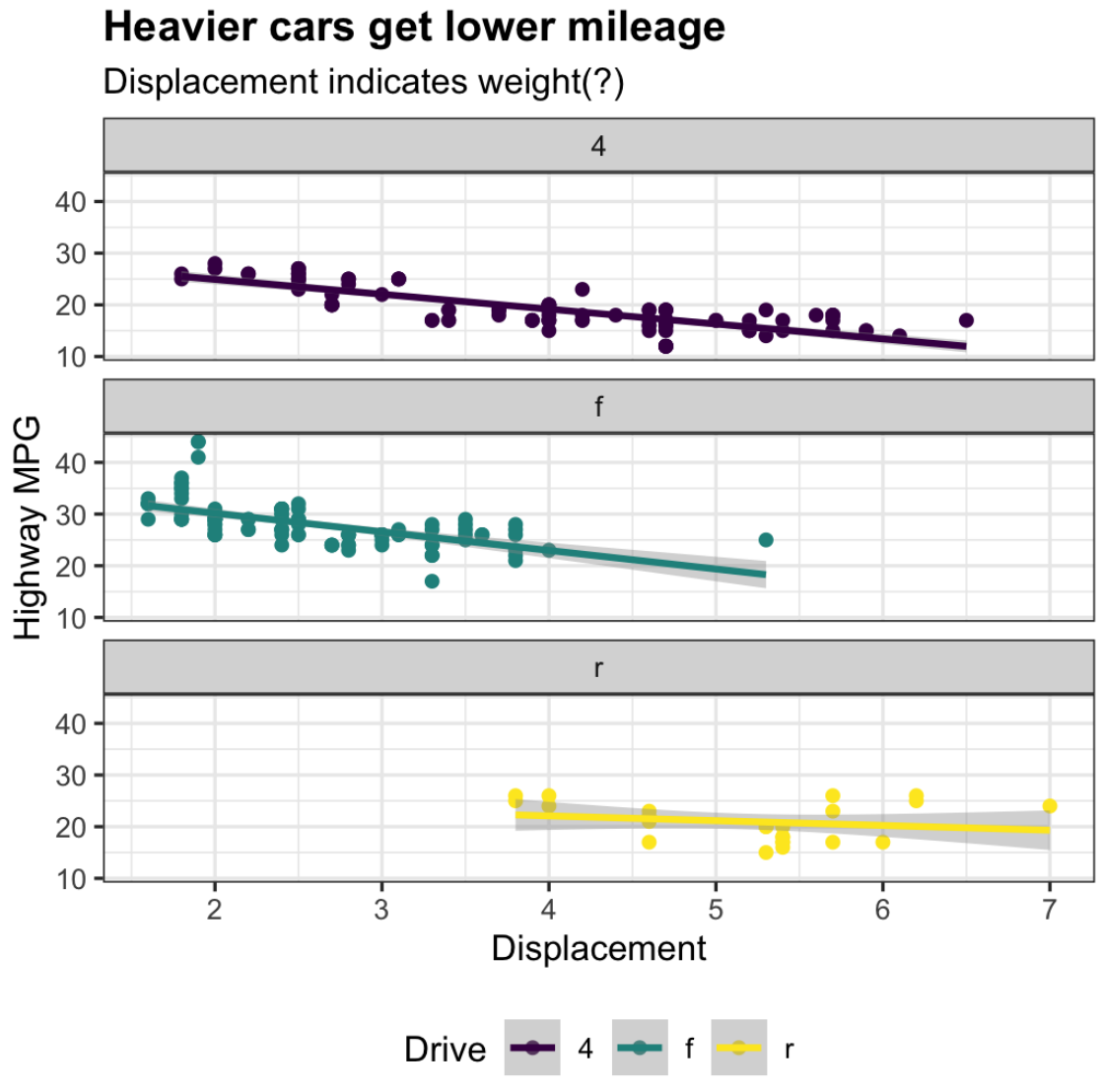
Modify the theme

```
ggplot(data = mpg,
       mapping = aes(x = displ,
                     y = hwy,
                     color = drv)) +
  geom_point() +
  geom_smooth(method = "lm") +
  scale_color_viridis_d() +
  facet_wrap(vars(drv), ncol = 1) +
  labs(x = "Displacement", y = "Highway MPG",
       color = "Drive",
       title = "Heavier cars get lower mileag
       subtitle = "Displacement indicates we
       caption = "I know nothing about cars")
  theme_bw() +
  theme(legend.position = "bottom",
        plot.title = element_text(face = "bo
```



Finished!

```
ggplot(data = mpg,
       mapping = aes(x = displ,
                     y = hwy,
                     color = drv)) +
  geom_point() +
  geom_smooth(method = "lm") +
  scale_color_viridis_d() +
  facet_wrap(vars(drv), ncol = 1) +
  labs(x = "Displacement", y = "Highway MPG",
       color = "Drive",
       title = "Heavier cars get lower mileag
       subtitle = "Displacement indicates we
       caption = "I know nothing about cars")
  theme_bw() +
  theme(legend.position = "bottom",
        plot.title = element_text(face = "bo
```

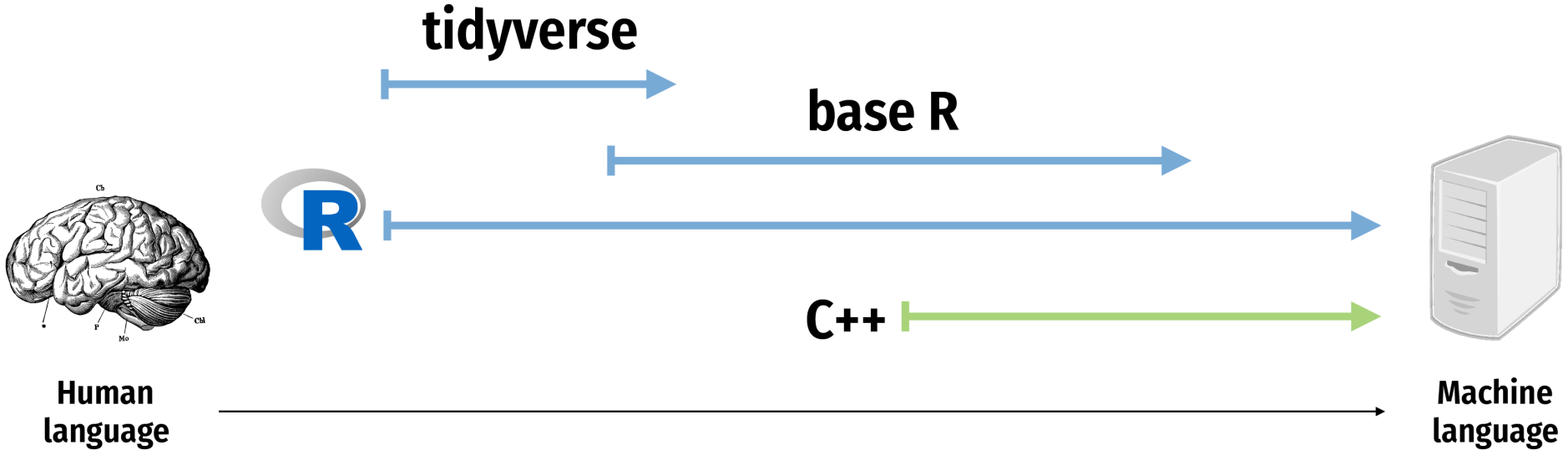


Transform data with dplyr

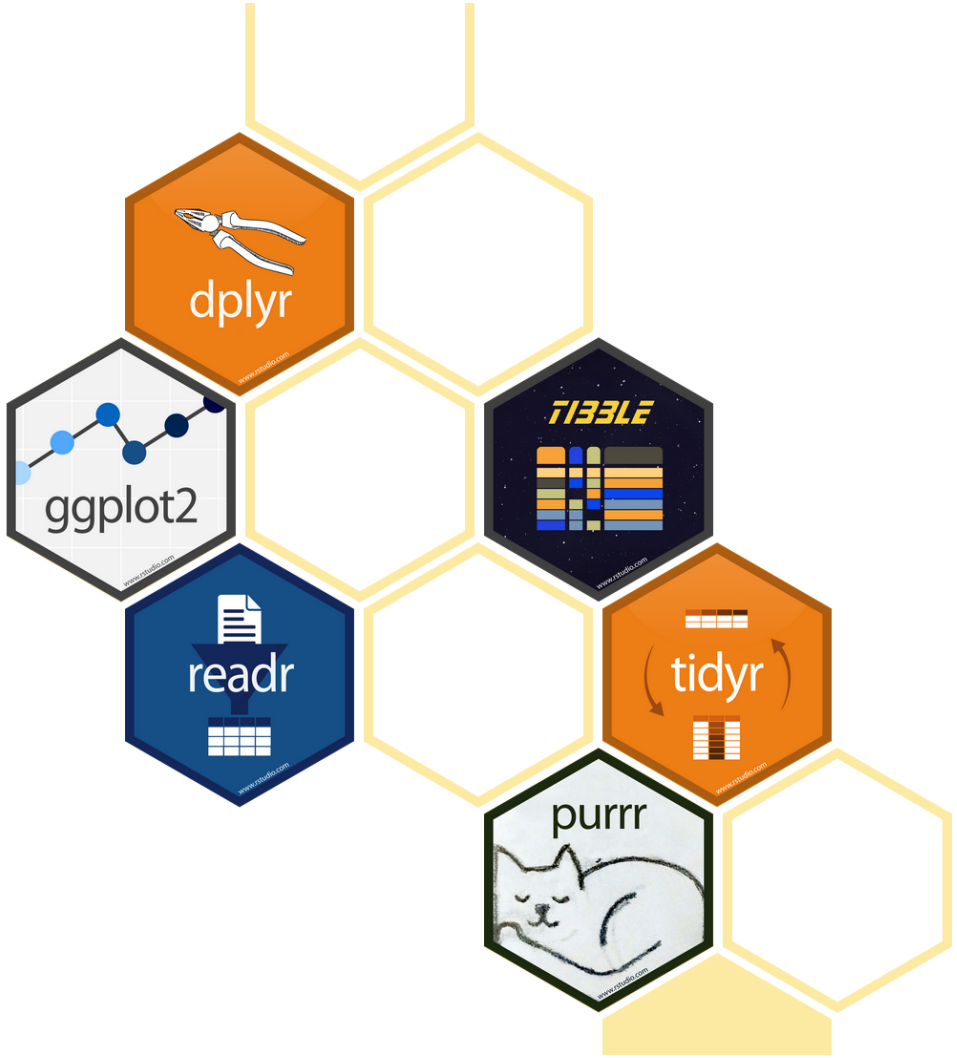
```
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # … with 1,694 more rows
```

The tidyverse

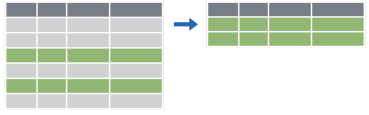


The tidyverse

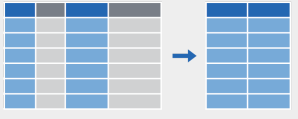


dplyr: verbs for manipulating data

Extract rows with `filter()`



Extract columns with `select()`

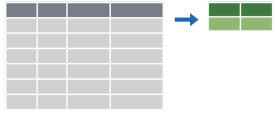


Arrange/sort rows with `arrange()`

Make new columns with `mutate()`

Make group summaries with

`group_by() %>% summarize()`



`filter()`

filter()

Extract rows that meet some sort of test

```
filter(.data = DATA, ...)
```

- **`DATA`** = Data frame to transform
- **`...`** = One or more tests

`filter()` returns each row for which the test is TRUE

```
filter(.data = gapminder, country == "Denmark")
```

country	continent	year
Afghanistan	Asia	1952
Afghanistan	Asia	1957
Afghanistan	Asia	1962
Afghanistan	Asia	1967
Afghanistan	Asia	1972
...	...	...

country	continent	year
Denmark	Europe	1952
Denmark	Europe	1957
Denmark	Europe	1962
Denmark	Europe	1967
Denmark	Europe	1972
Denmark	Europe	1977

filter()

```
filter(.data = gapminder,
       country ==
"Denmark")
```

One = sets an argument

Two == tests if equal

returns TRUE or FALSE)

Logical tests

Test	Meaning	Test	Meaning
`x < y`	Less than	`x %in% y`	In (group membership)
`x > y`	Greater than	`is.na(x)`	Is missing
`==`	Equal to	`!is.na(x)`	Is not missing
`x <= y`	Less than or equal to		
`x >= y`	Greater than or equal to		
`x != y`	Not equal to		

Your turn #1: Filtering

Use `filter()` and logical tests to show...

1. The data for Canada
2. All data for countries in Oceania
3. Rows where the life expectancy is greater than 82

```
filter(gapminder, country == "Canada")
```

```
filter(gapminder, continent == "Oceania")
```

```
filter(gapminder, lifeExp > 82)
```

Common mistakes

Using = instead of ==

```
filter(gapminder,
       country = "Canada")
```

```
filter(gapminder,
       country == "Canada")
```

Quote use

```
filter(gapminder,
       country == Canada)
```

```
filter(gapminder,
       country == "Canada")
```

`filter()` with multiple conditions

Extract rows that meet *every* test

```
filter(gapminder, country == "Denmark", year > 2000)
```

```
filter(gapminder, country == "Denmark", year > 2000)
```

country	continent	year
Afghanistan	Asia	1952
Afghanistan	Asia	1957
Afghanistan	Asia	1962
Afghanistan	Asia	1967
Afghanistan	Asia	1972
…	…	…

country	continent	year
Denmark	Europe	2002
Denmark	Europe	2007

Boolean operators

Operator	Meaning
`a & b`	and
`a | b`	or
`!a`	not

Default is "and"

These do the same thing:

```
filter(gapminder, country == "Denmark", year > 2000)
```

```
filter(gapminder, country == "Denmark" & year > 2000)
```

Your turn #2: Filtering

Use `filter()` and Boolean logical tests to show...

1. Canada before 1970
2. Countries where life expectancy in 2007 is below 50
3. Countries where life expectancy in 2007 is below 50 and are not in Africa

```
filter(gapminder, country == "Canada", year < 1970)
```

```
filter(gapminder, year == 2007, lifeExp < 50)
```

```
filter(gapminder, year == 2007, lifeExp < 50,
       continent != "Africa")
```

Common mistakes

Collapsing multiple tests into one

```
filter(gapminder, 1960 < year < 1980)
```

```
filter(gapminder,
       year > 1960, year < 1980)
```

Using multiple tests instead of `%in%`

```
filter(gapminder,
       country == "Mexico",
       country == "Canada",
       country == "United States")
```

```
filter(gapminder,
       country %in% c("Mexico", "Canada",
                      "United States"))
```

Common syntax

Every dplyr verb function follows the same pattern

First argument is a data frame; returns a data frame

```
VERB(DATA, ...)
```

- `VERB` = dplyr function/verb
- `DATA` = Data frame to transform
- `...` = Stuff the verb does

mutate()

Create new columns

```
mutate(.data, ...)
```

- **`.data`** = Data frame to transform
- **`...`** = Columns to make

```
mutate(gapminder, gdp = gdpPercap * pop)
```

country	year	gdpPercap	pop
Afghanistan	1952	779.4453145	8425333
Afghanistan	1957	820.8530296	9240934
Afghanistan	1962	853.10071	10267083
Afghanistan	1967	836.1971382	11537966
Afghanistan	1972	739.9811058	13079460
...	...	...	...

country	year	...	gdp
Afghanistan	1952	...	6567086330
Afghanistan	1957	...	7585448670
Afghanistan	1962	...	8758855797
Afghanistan	1967	...	9648014150
Afghanistan	1972	...	9678553274
Afghanistan	1977	...	11697659231

```
mutate(gapminder, gdp = gdpPercap * pop,
                  pop_mil = round(pop / 1000000))
```

country	year	gdpPercap	pop
Afghanistan	1952	779.4453145	8425333
Afghanistan	1957	820.8530296	9240934
Afghanistan	1962	853.10071	10267083
Afghanistan	1967	836.1971382	11537966
Afghanistan	1972	739.9811058	13079460
...	...	...	...

country	year	...	gdp	pop_mil
Afghanistan	1952	...	6567086330	8
Afghanistan	1957	...	7585448670	9
Afghanistan	1962	...	8758855797	10
Afghanistan	1967	...	9648014150	12
Afghanistan	1972	...	9678553274	13
Afghanistan	1977	...	11697659231	15

ifelse()

Do conditional tests within `mutate()`

```
ifelse(TEST,
       VALUE_IF_TRUE,
       VALUE_IF_FALSE)
```

- **`TEST`** = A logical test
- **`VALUE_IF_TRUE`** = What happens if test is true
- **`VALUE_IF_FALSE`** = What happens if test is false

```
mutate(gapminder,
       after_1960 = ifelse(year > 1960, TRUE, FALSE))
```

```
mutate(gapminder,
       after_1960 = ifelse(year > 1960,
                           "After 1960",
                           "Before 1960"))
```

Your turn #3: Mutating

Use `mutate()` to...

1. Add an `africa` column that is TRUE if the country is on the African continent
2. Add a column for logged GDP per capita (hint: use `log()`)
3. Add an `africa_asia` column that says “Africa or Asia” if the country is in Africa or Asia, and “Not Africa or Asia” if it’s not

```
mutate(gapminder, africa = ifelse(continent == "Africa",
                                  TRUE, FALSE))
```

```
mutate(gapminder, log_gdpPercap = log(gdpPercap))
```

```
mutate(gapminder,
       africa_asia =
         ifelse(continent %in% c("Africa", "Asia"),
                "Africa or Asia",
                "Not Africa or Asia"))
```

What if you have multiple verbs?

Make a dataset for just 2002 *and* calculate logged GDP per capita

Solution 1: Intermediate variables

```
gapminder_2002 <- filter(gapminder, year == 2002)

gapminder_2002_log <- mutate(gapminder_2002,
                             log_gdpPercap = log(gdpPercap))
```

What if you have multiple verbs?

Make a dataset for just 2002 *and* calculate logged GDP per capita

Solution 2: Nested functions

```
filter(mutate(gapminder_2002,
              log_gdpPercap = log(gdpPercap)),
       year == 2002)
```

What if you have multiple verbs?

Make a dataset for just 2002 *and* calculate logged GDP per capita

Solution 3: Pipes!

The `%>%` operator (pipe) takes an object on the left and passes it as the first argument of the function on the right

```
gapminder %>% filter(_, country == "Canada")
```

What if you have multiple verbs?

These do the same thing!

```
filter(gapminder, country == "Canada")
```

```
gapminder %>% filter(country == "Canada")
```

What if you have multiple verbs?

Make a dataset for just 2002 *and* calculate logged GDP per capita

Solution 3: Pipes!

```
gapminder %>%
  filter(year == 2002) %>%
  mutate(log_gdpPercap = log(gdpPercap))
```

%>%

```
leave_house(get_dressed(get_out_of_bed(wake_up(me, time =
"8:00"), side = "correct"), pants = TRUE, shirt = TRUE), car
= TRUE, bike = FALSE)
```

```
me %>%
  wake_up(time = "8:00") %>%
  get_out_of_bed(side = "correct") %>%
  get_dressed(pants = TRUE, shirt = TRUE) %>%
  leave_house(car = TRUE, bike = FALSE)
```

summarize()

Compute a table of summaries

```
gapminder %>% summarize(mean_life = mean(lifeExp))
```

country	continent	year	lifeExp
Afghanistan	Asia	1952	28.801
Afghanistan	Asia	1957	30.332
Afghanistan	Asia	1962	31.997
Afghanistan	Asia	1967	34.02
…	…	…	…

mean_life
59.47444

summarize()

```
gapminder %>% summarize(mean_life = mean(lifeExp),
                        min_life = min(lifeExp))
```

country	continent	year	lifeExp
Afghanistan	Asia	1952	28.801
Afghanistan	Asia	1957	30.332
Afghanistan	Asia	1962	31.997
Afghanistan	Asia	1967	34.02
Afghanistan	Asia	1972	36.088
...	...	...	...

mean_life	min_life
59.47444	23.599

Your turn #4: Summarizing

Use `summarize()` to calculate...

1. The first (minimum) year in the dataset
2. The last (maximum) year in the dataset
3. The number of rows in the dataset (use the cheatsheet)
4. The number of distinct countries in the dataset (use the cheatsheet)

```
gapminder %>%
  summarize(first = min(year),
            last = max(year),
            num_rows = n(),
            num_unique = n_distinct(country))
```

first	last	num_rows	num_unique
1952	2007	1704	142

Your turn #5: Summarizing

Use `filter()` and `summarize()` to calculate (1) the number of unique countries and (2) the median life expectancy on the African continent in 2007

```
gapminder %>%
  filter(continent == "Africa", year == 2007) %>%
  summarise(n_countries = n_distinct(country),
            med_le = median(lifeExp))
```

n_countries	med_le
52	52.9265

group_by()

Put rows into groups based on values in a column

```
gapminder %>% group_by(continent)
```

Nothing happens by itself!

Powerful when combined with `summarize()`

```
gapminder %>%
  group_by(continent) %>%
  summarize(n_countries = n_distinct(country))
```

continent	n_countries
Africa	52
Americas	25
Asia	33
Europe	30
Oceania	2

```
pollution %>%
  summarize(mean = mean(amount), sum = sum(amount), n = n())
```

city	particle_size	amount
New York	Large	23
New York	Small	14
London	Large	22
London	Small	16
Beijing	Large	121
Beijing	Small	56

mean	sum	n
42	252	6

```
pollution %>%
  group_by(city) %>%
  summarize(mean = mean(amount), sum = sum(amount), n = n())
```

city	particle_size	amount
New York	Large	23
New York	Small	14
London	Large	22
London	Small	16
Beijing	Large	121
Beijing	Small	56

city	mean	sum	n
Beijing	88.5	177	2
London	19.0	38	2
New York	18.5	37	2

```
pollution %>%
  group_by(particle_size) %>%
  summarize(mean = mean(amount), sum = sum(amount), n = n())
```

city	particle_size	amount
New York	Large	23
New York	Small	14
London	Large	22
London	Small	16
Beijing	Large	121
Beijing	Small	56

particle_size	mean	sum	n
Large	55.33333	166	3
Small	28.66667	86	3

Your turn #6: Grouping and summarizing

Find the minimum, maximum, and median life expectancy for each continent

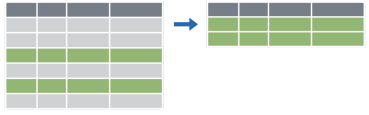
Find the minimum, maximum, and median life expectancy for each continent in 2007 only

```
gapminder %>%
  group_by(continent) %>%
  summarize(min_le = min(lifeExp),
            max_le = max(lifeExp),
            med_le = median(lifeExp))
```

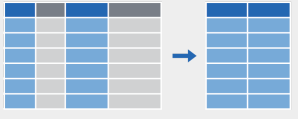
```
gapminder %>%
  filter(year == 2007) %>%
  group_by(continent) %>%
  summarize(min_le = min(lifeExp),
            max_le = max(lifeExp),
            med_le = median(lifeExp))
```

dplyr: verbs for manipulating data

Extract rows with `filter()`



Extract columns with `select()`



Arrange/sort rows with `arrange()`

Make new columns with `mutate()`

Make group summaries with

`group_by() %>% summarize()`