Mapping data to graphics

Week 5

AEM 2850: R for Business Analytics
Cornell Dyson
Spring 2022

Acknowledgements: Andrew Heiss, Allison Horst

Announcements

Next week is February break

- No Lab due next week
- Yes Reflection due Wednesday, March 2

We will provide details on the first project in the next 1-2 weeks

Questions before we get started?

Plan for today

Course progress

Prologue

Data, aesthetics, & the grammar of graphics

Grammatical layers

Aesthetics in extra dimensions

Tidy data revisited

Course progress

Course objectives reminder

1. Develop basic proficiency in R programming
2. Understand data structures and manipulation
3. Describe effective techniques for data visualization and communication
4. Construct effective data visualizations
5. Utilize course concepts and tools for business applications

Where we've been

1. **Develop basic proficiency in R programming**
2. **Understand data structures and manipulation**
3. Describe effective techniques for data visualization and communication
4. Construct effective data visualizations
5. Utilize course concepts and tools for business applications

Where we're going next

1. Develop basic proficiency in R programming
2. Understand data structures and manipulation
3. **Describe effective techniques for data visualization and communication**
4. **Construct effective data visualizations**
5. Utilize course concepts and tools for business applications

Schedule overview

Weeks 1-4: Programming Foundations

Weeks 5-10: Data Visualization Foundations

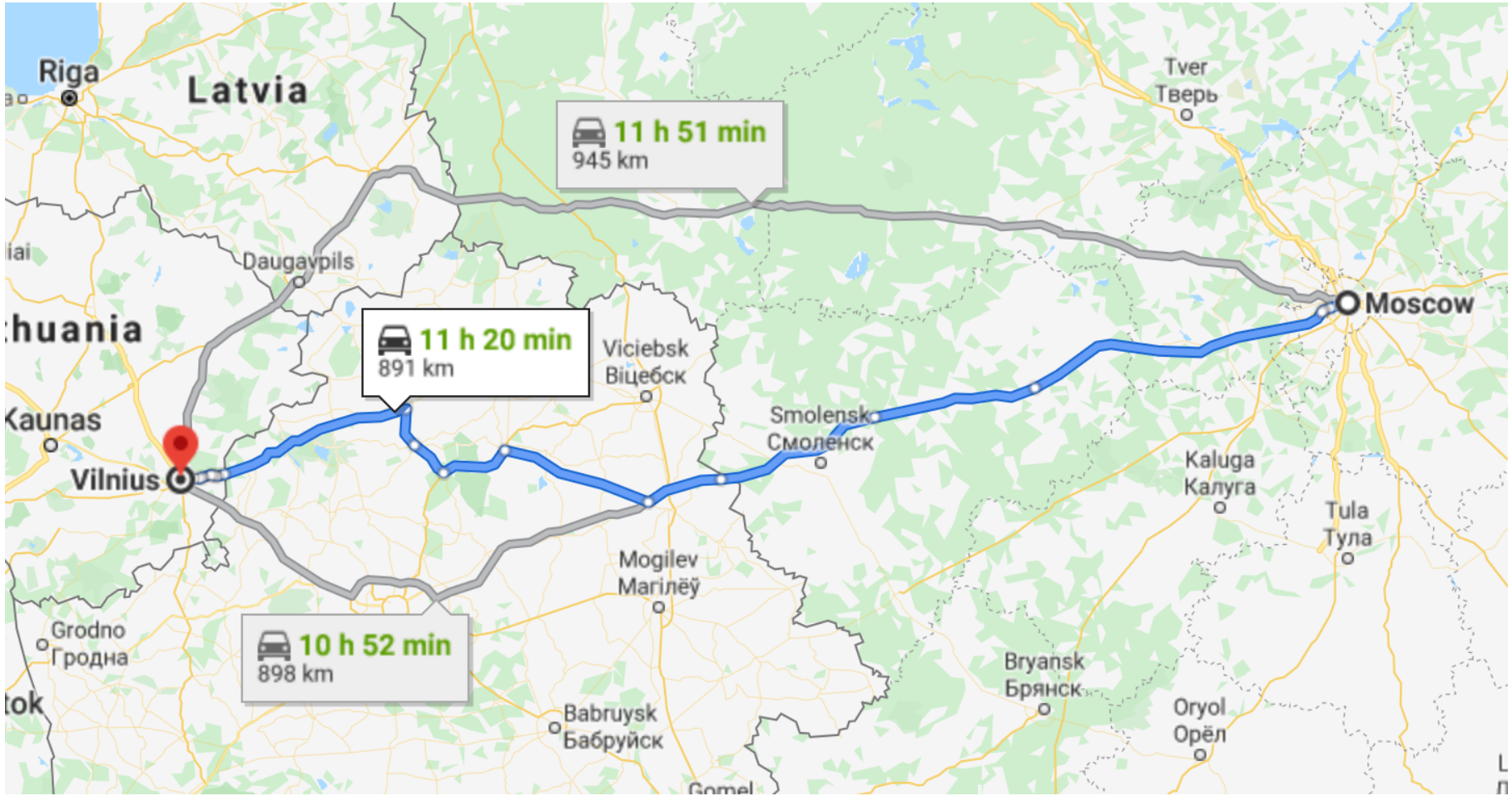
Weeks 11+: Special Topics (mix of programming and dataviz)

See aem2850.toddgerarden.com/schedule for details

Prologue

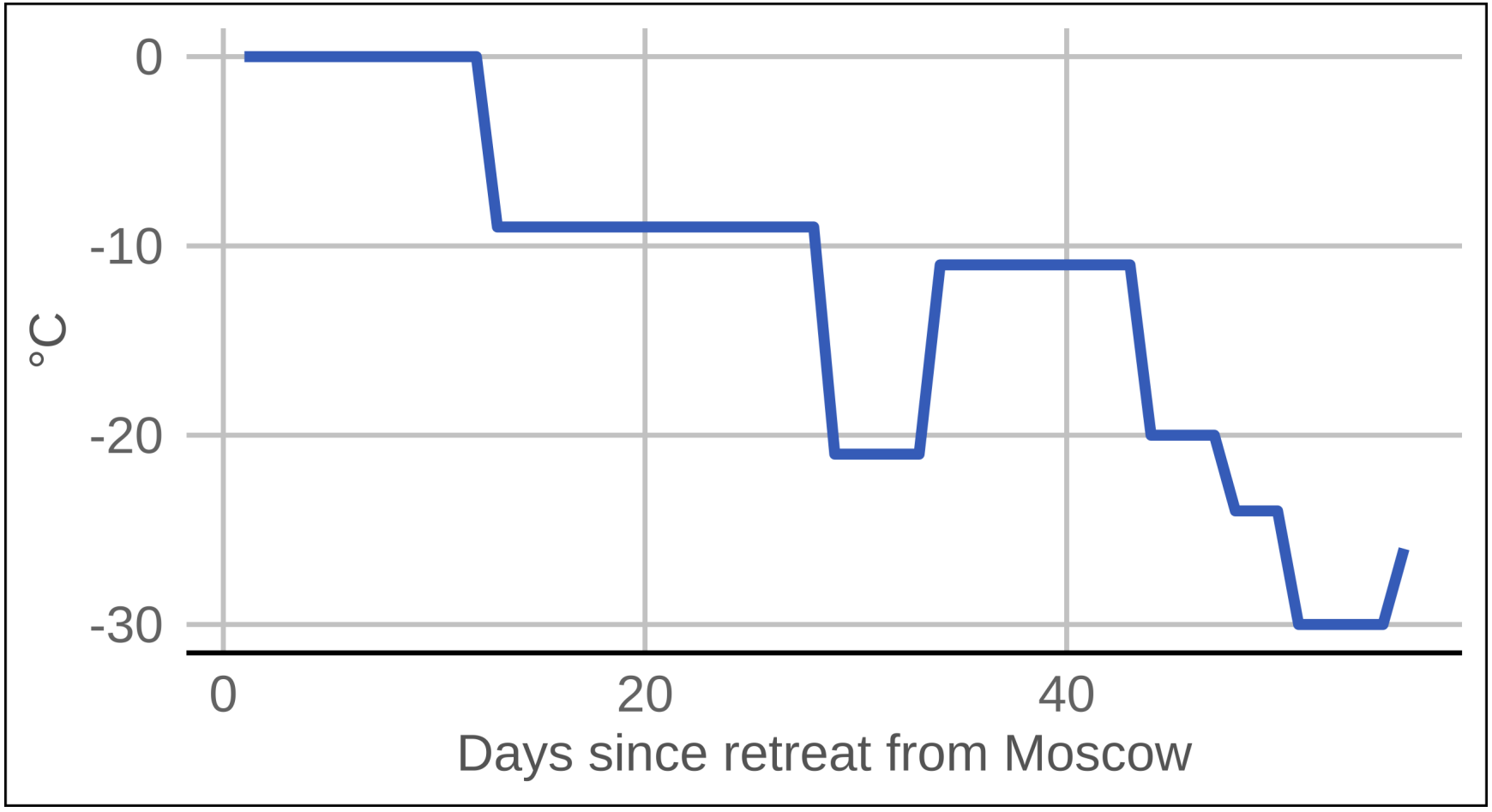
Do you know anything about Napoleon's retreat from Moscow?

Long distance!

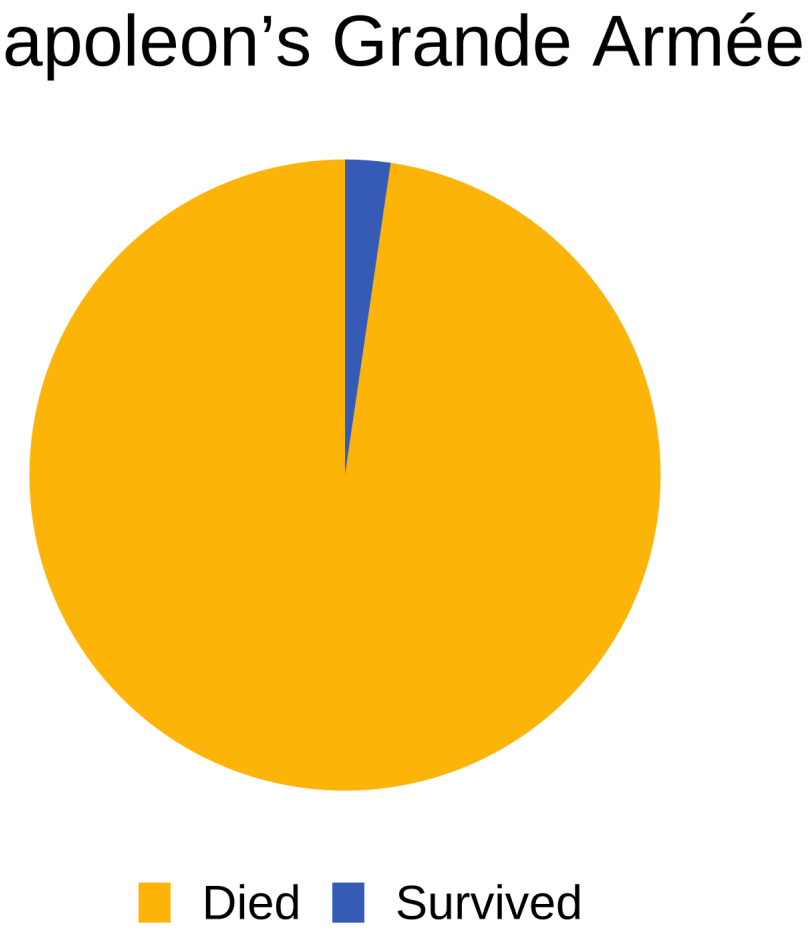


Moscow to Vilnius

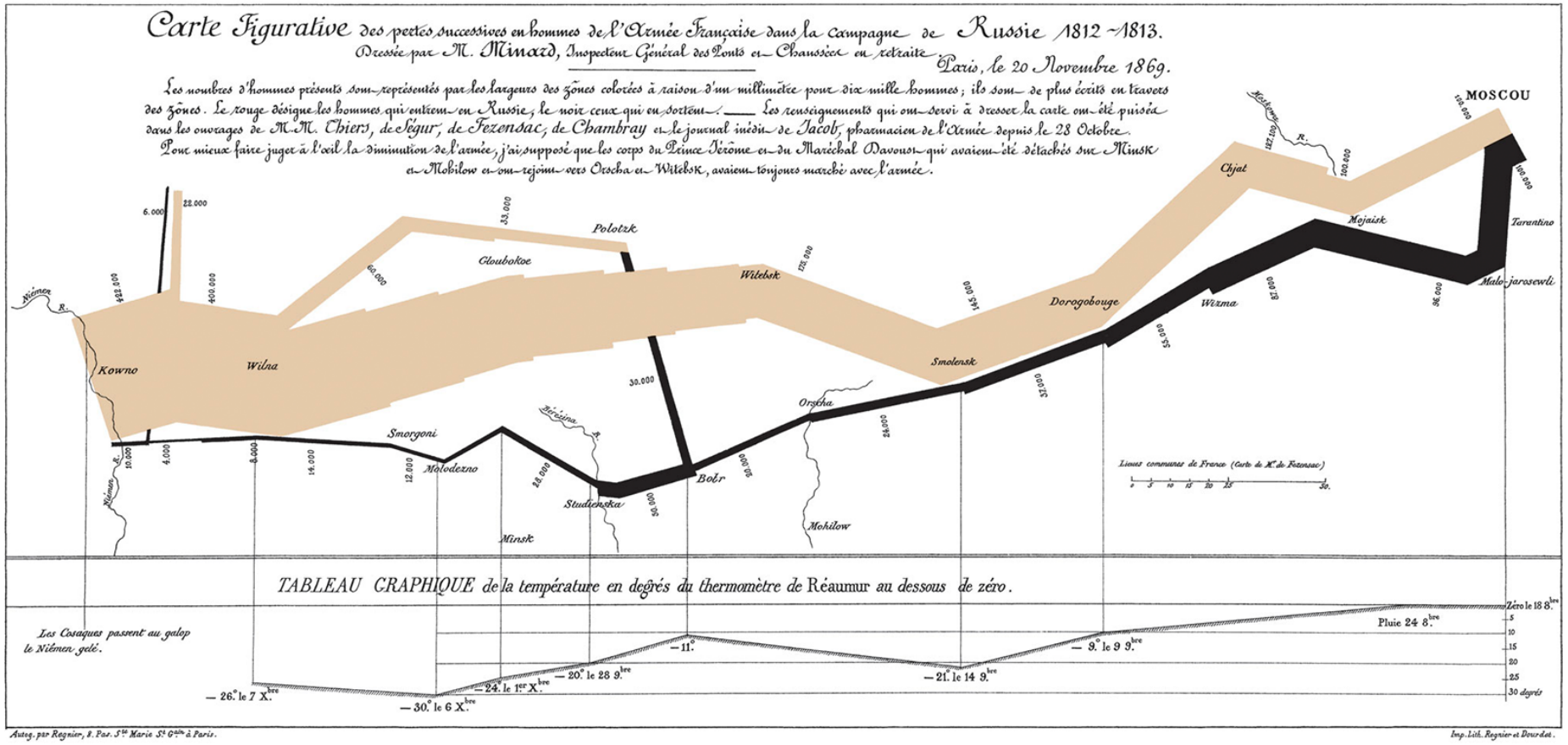
Very cold!



Lots of people died!

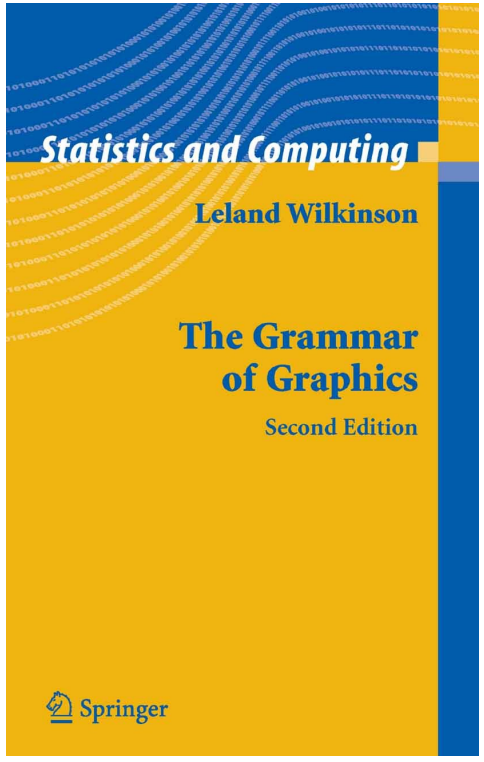


"Probably the best statistical graphic ever"



Data, aesthetics,
& the grammar of graphics

Mapping data to aesthetics



Data

A column in a dataset

Aesthetics

Visual properties of a graph

Position, shape, color, etc.

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 using ggplot

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()`

Barebones `ggplot2::ggplot()` template

```
library(tidyverse) # tidyverse loads the package ggplot2

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

Mapping from `troops` to aesthetics

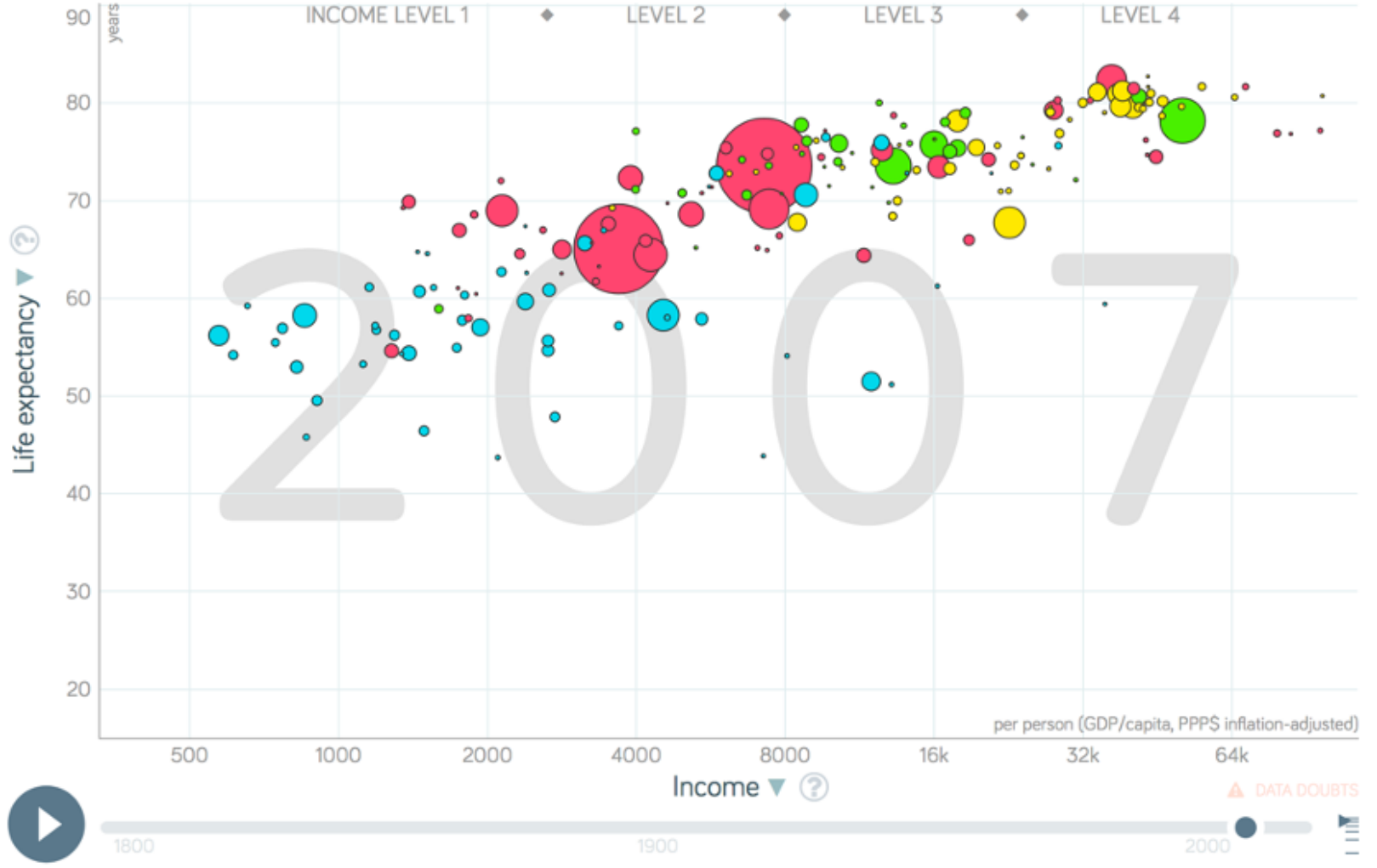
longitude	latitude	direction	survivors
24	54.9	A	340000
24.5	55	A	340000
…	…	…	…

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

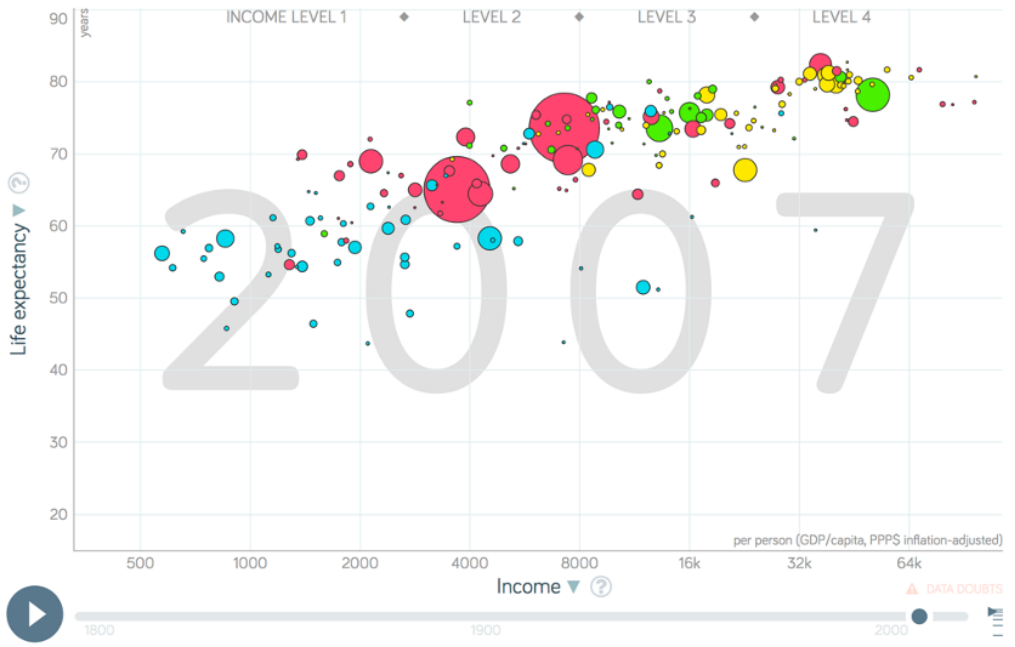
Probably not the best statistical graphic ever

But it's a solid start!

What did you think of the Hans Rosling video?

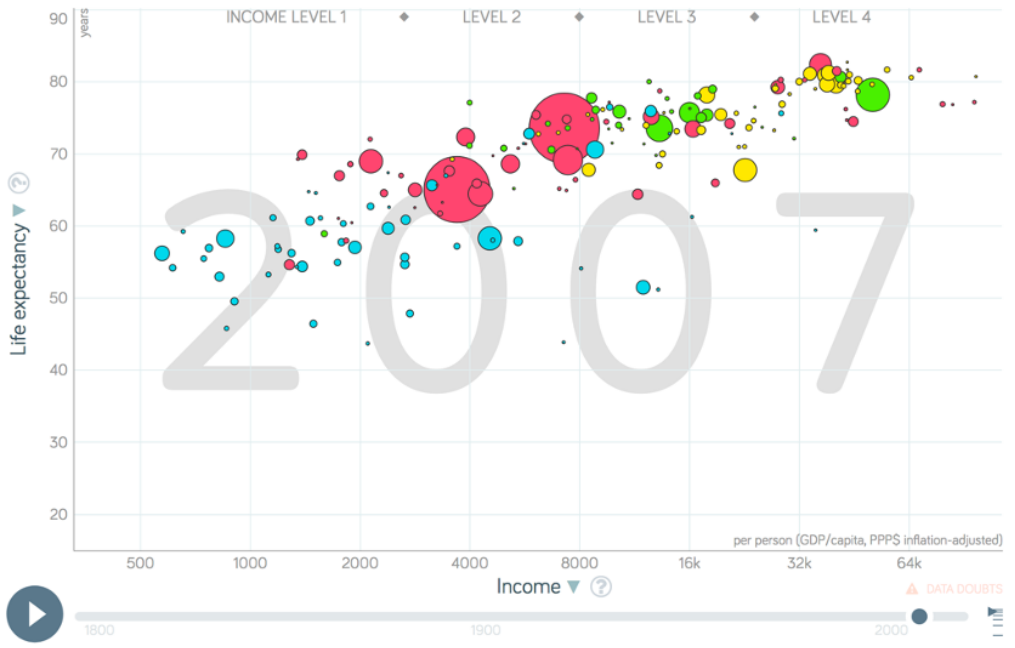


Mapping from health and wealth to aesthetics



Data	`aes()`	`geom`
Wealth	?	`geom_point()`
Health	?	?
Continent	`color`	?
Population	?	?

Mapping from health and wealth to aesthetics



Data	`aes()`	`geom`
Wealth	`x`	`geom_point()`
Health	`y`	`geom_point()`
Continent	`color`	`geom_point()`
Population	`size`	`geom_point()`

Mapping from `gapminder` to aesthetics

country	continent	gdpPercap	lifeExp	pop
Afghanistan	Asia	974.5803384	43.828	31889923
Albania	Europe	5937.029526	76.423	3600523
...	...	...	...	...

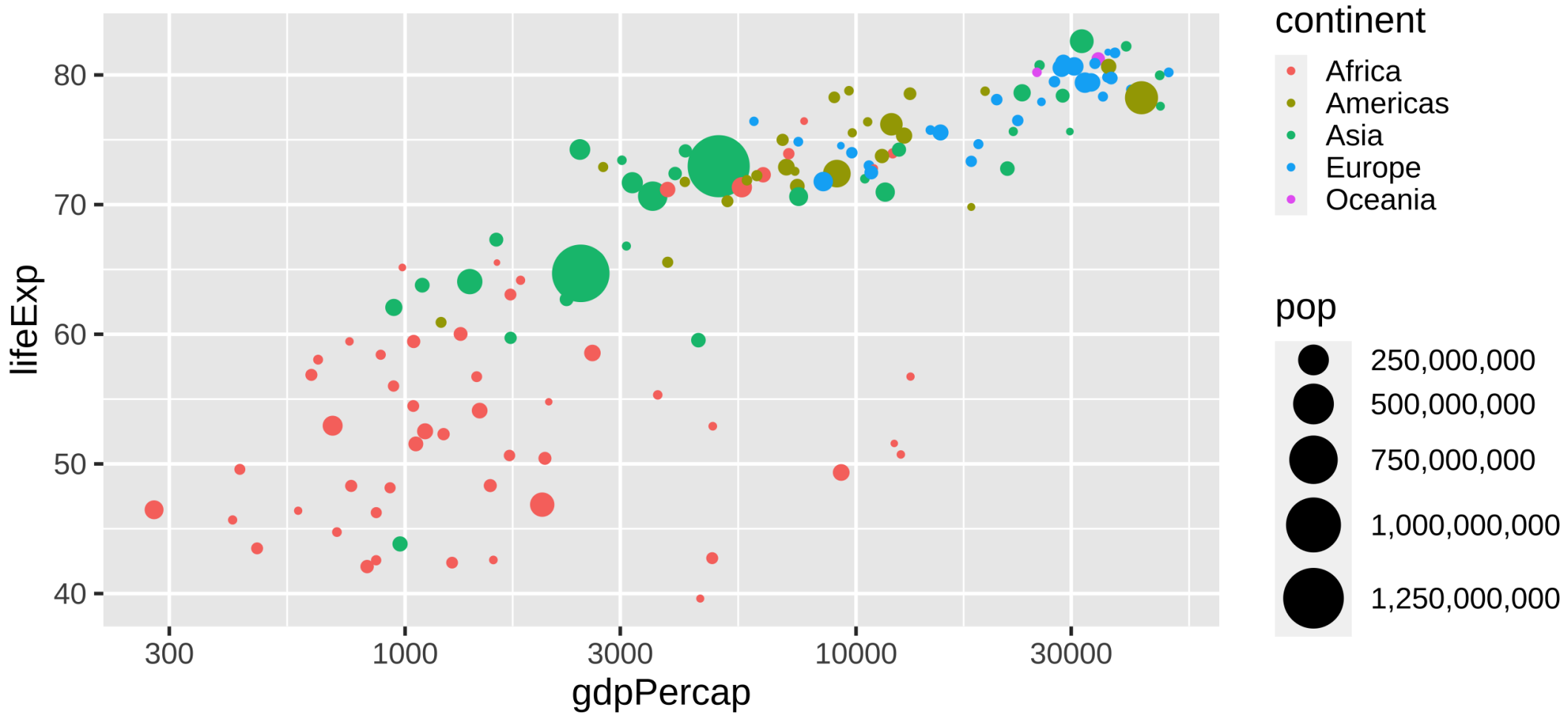
```
ggplot(data = gapminder,
       mapping = aes(x =          ,
                     y =          ,
                     color =          ,
                     size =    )) +
  geom_       () +
  scale_x_log10()
```

Mapping from `gapminder` to aesthetics

country	continent	gdpPercap	lifeExp	pop
Afghanistan	Asia	974.5803384	43.828	31889923
Albania	Europe	5937.029526	76.423	3600523
...	...	...	...	...

```
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap,
                     y = lifeExp,
                     color = continent,
                     size = pop)) +
  geom_point() +
  scale_x_log10()
```

Health and wealth



Grammatical layers

Grammar components as layers

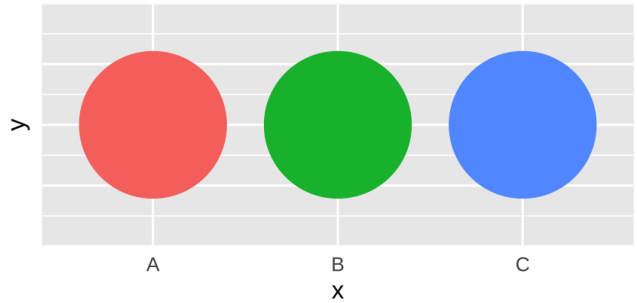
So far we know about data, aesthetics, and geometries

Think of these components as **layers**

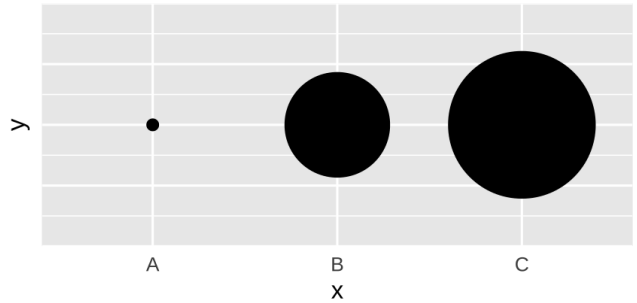
Add them to foundational `ggplot()` with `+`

Possible aesthetics

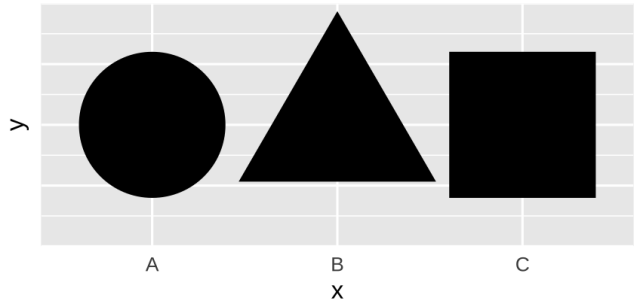
`color` (discrete)



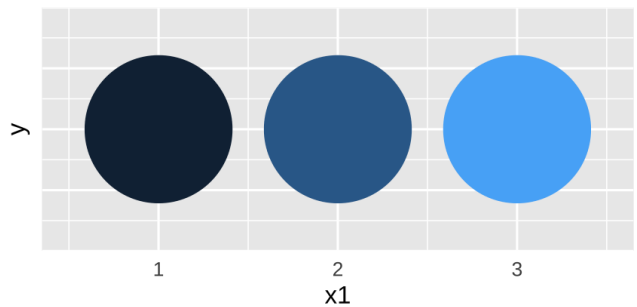
`size`



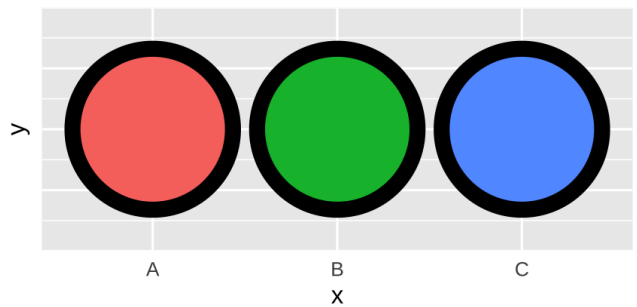
`shape`



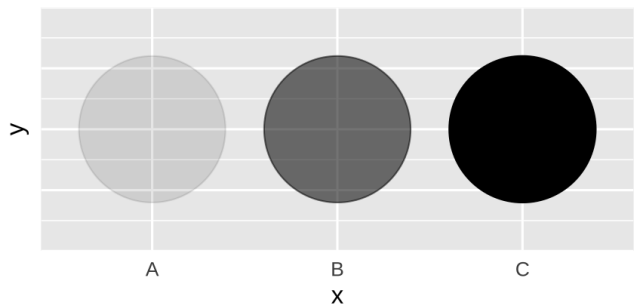
`color` (continuous)



`fill`



`alpha` (opacity)



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

Over the next several weeks we will cover a number of them

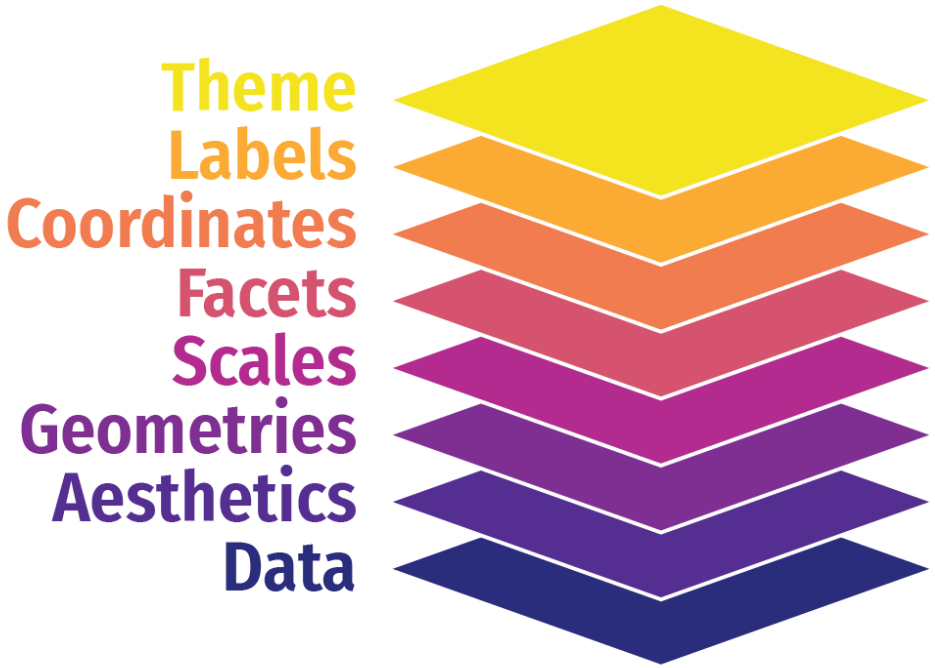
See the **ggplot2** documentation for examples of all the different geom layers

Additional layers

There are many other grammatical layers we can use to describe graphs!

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

I will give you a quick preview, don't worry about all the details for now



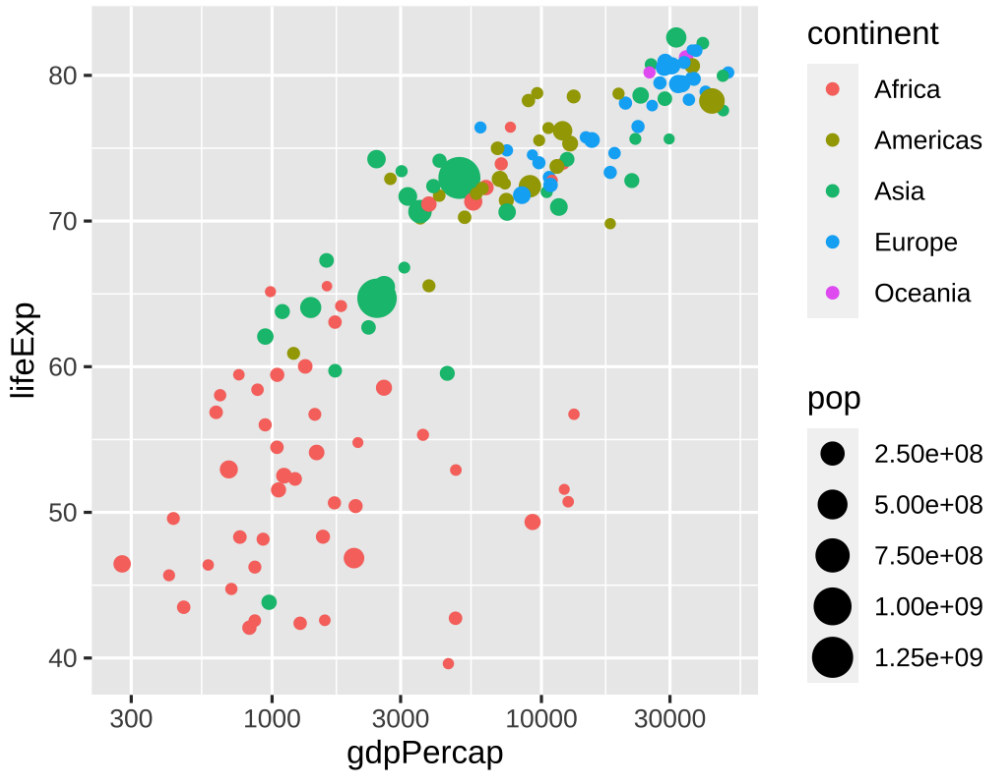
Scales

Scales change the properties of the variable mapping

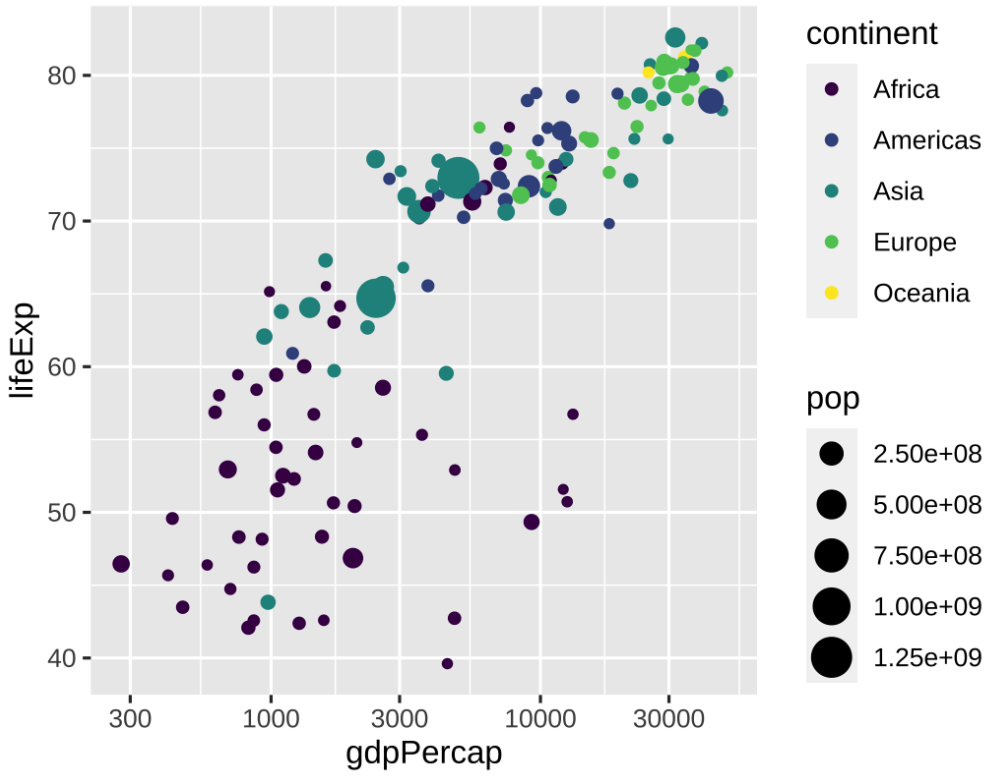
Example layer	What it does
`scale_x_continuous()`	Make the x-axis continuous
`scale_x_continuous(breaks = 1:5)`	Manually specify axis ticks
`scale_x_log10()`	Log the x-axis
`scale_color_gradient()`	Use a gradient
`scale_fill_viridis_d()`	Fill with discrete viridis colors

Scales

`scale_x_log10()`



`scale_color_viridis_d()`



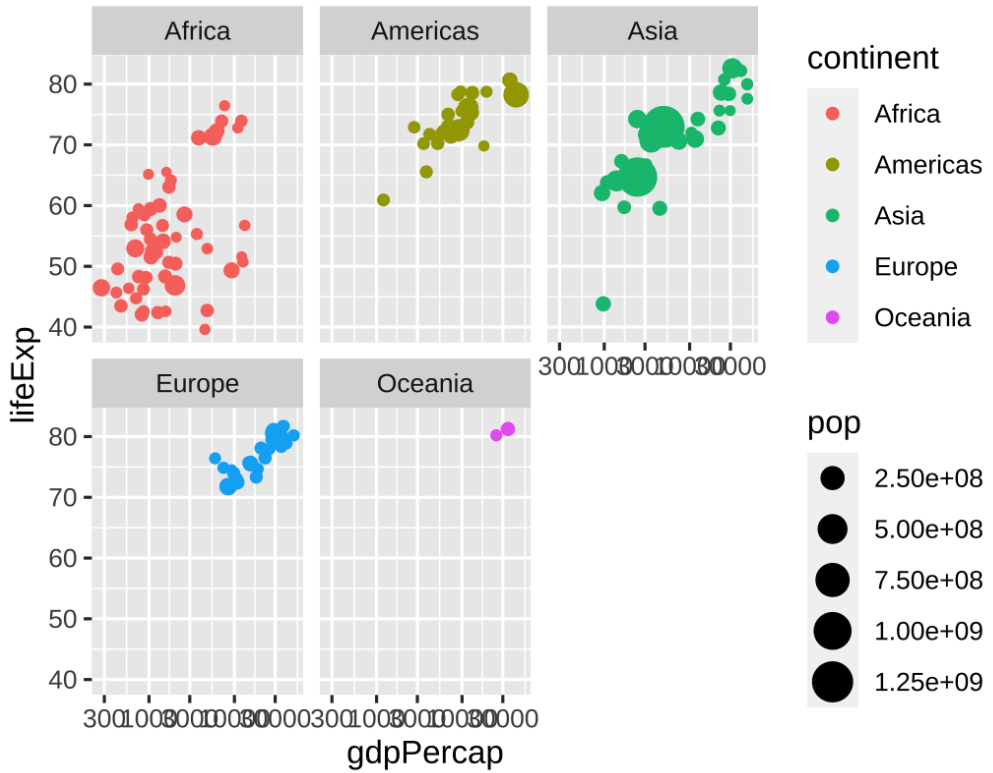
Facets

Facets show subplots for different subsets of data

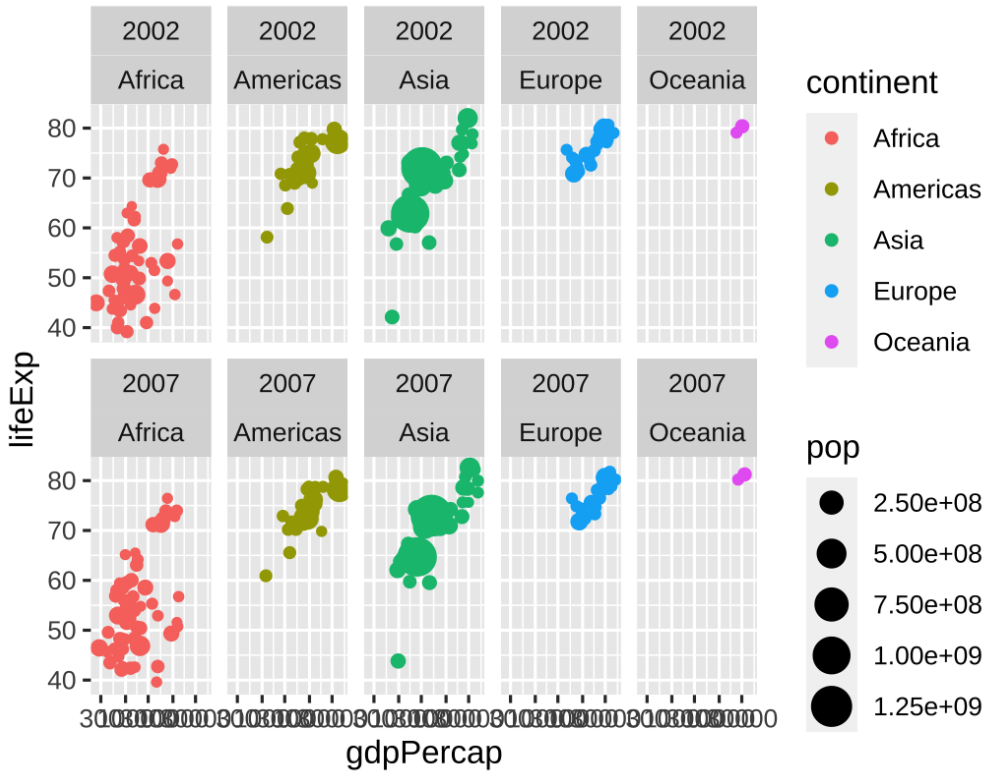
Example layer	What it does
`facet_wrap(vars(continent))`	Plot for each continent
`facet_wrap(vars(continent, year))`	Plot for each continent/year
`facet_wrap(..., ncol = 1)`	Put all facets in one column
`facet_wrap(..., nrow = 1)`	Put all facets in one row

Facets

`facet_wrap(vars(continent))`



`facet_wrap(vars(continent, year))`



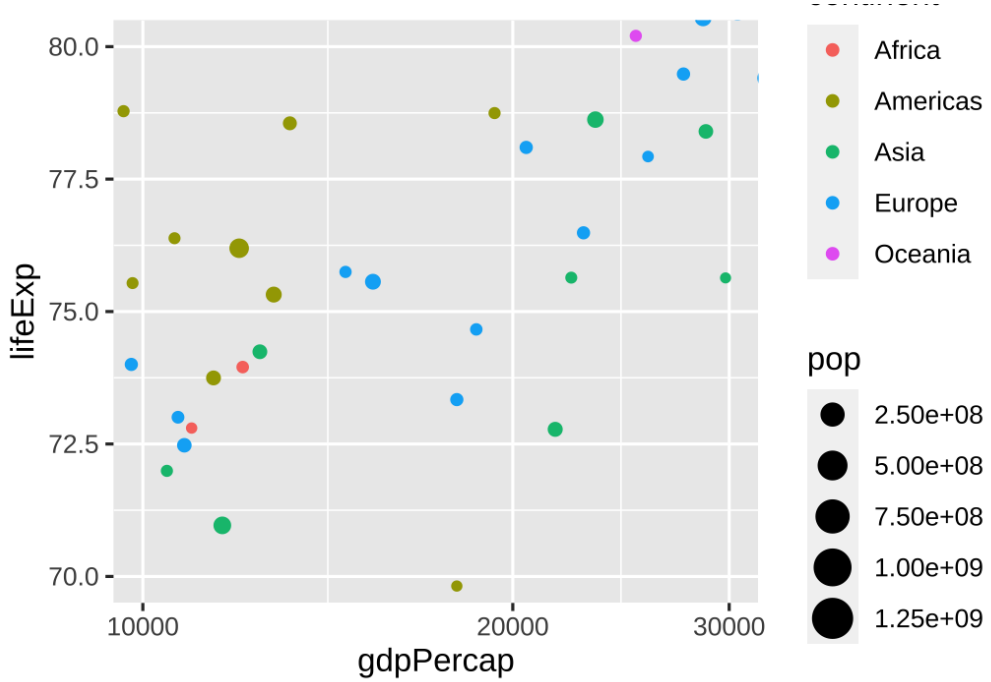
Coordinates

Change the coordinate system

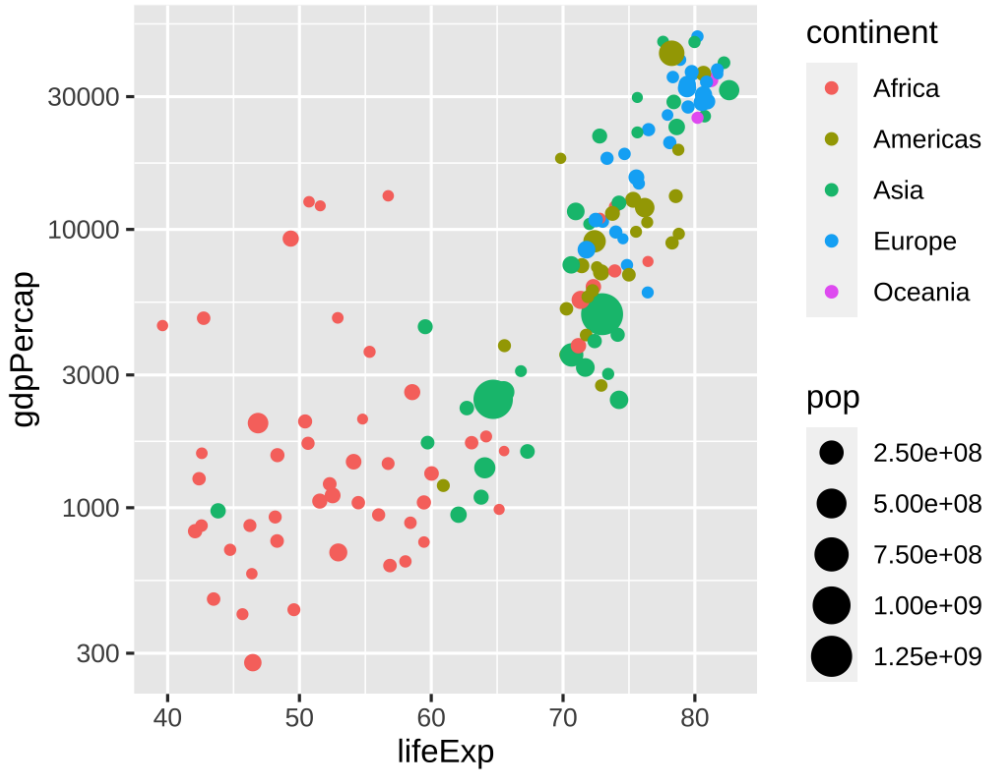
Example layer	What it does
`coord_cartesian(ylim = c(1, 10))`	Zoom in where y is 1–10
`coord_flip()`	Switch x and y
`coord_polar()`	Use polar coordinates

Coordinates

```
coord_cartesian(ylim = c(70, 80),
    xlim = c(10000, 30000))
```




```
coord_flip()
```



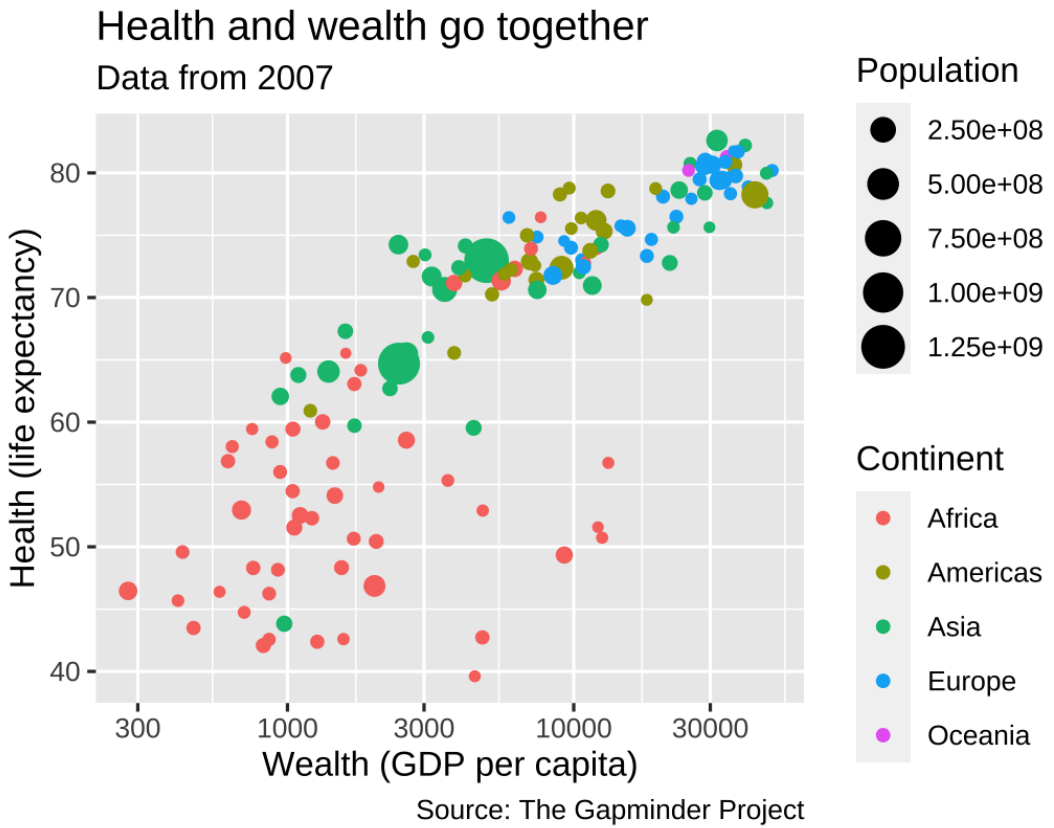
Labels

Add labels to the plot with a single `labs()` layer

Example layer	What it does
`labs(title = "Neat title")`	Title
`labs(caption = "Something")`	Caption
`labs(y = "Something")`	y-axis
`labs(size = "Population")`	Title of size legend

Labels

```
ggplot(gapminder_2007,
       aes(x = gdpPercap, y = lifeExp,
           color = continent, size = pop)) +
  geom_point() +
  scale_x_log10() +
  labs(title = "Health and wealth go together",
       subtitle = "Data from 2007",
       x = "Wealth (GDP per capita)",
       y = "Health (life expectancy)",
       color = "Continent",
       size = "Population",
       caption = "Source: The Gapminder Project")
```



Theme

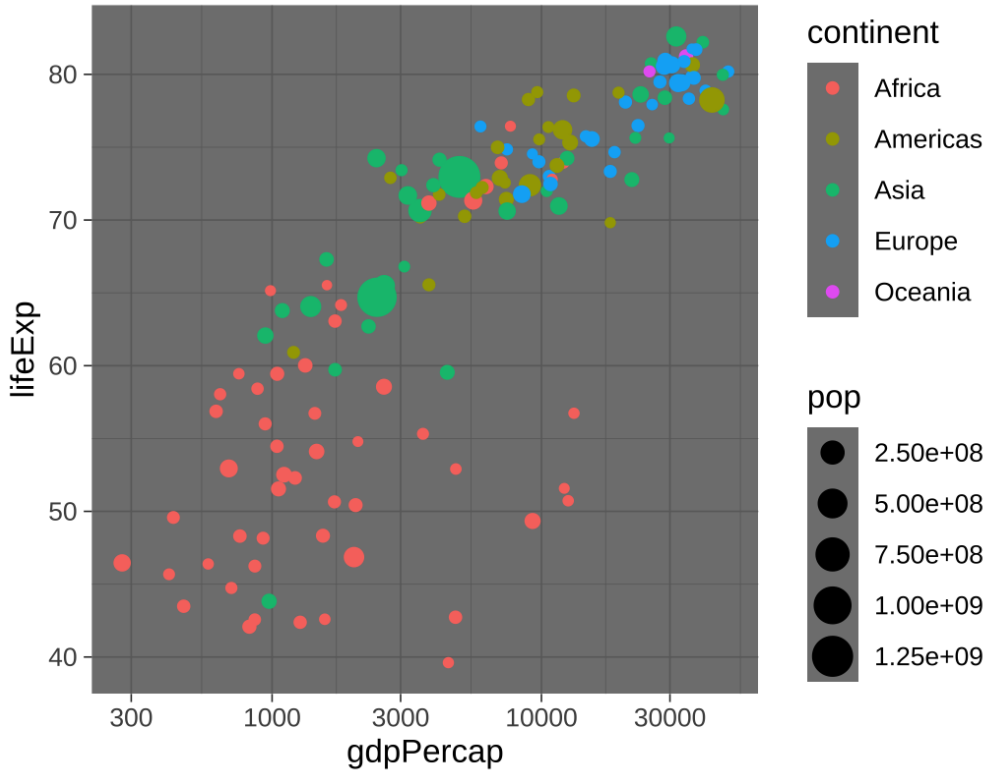
`theme()` can be used to change the appearance of anything in a plot

Lots of themes built in and available from other packages

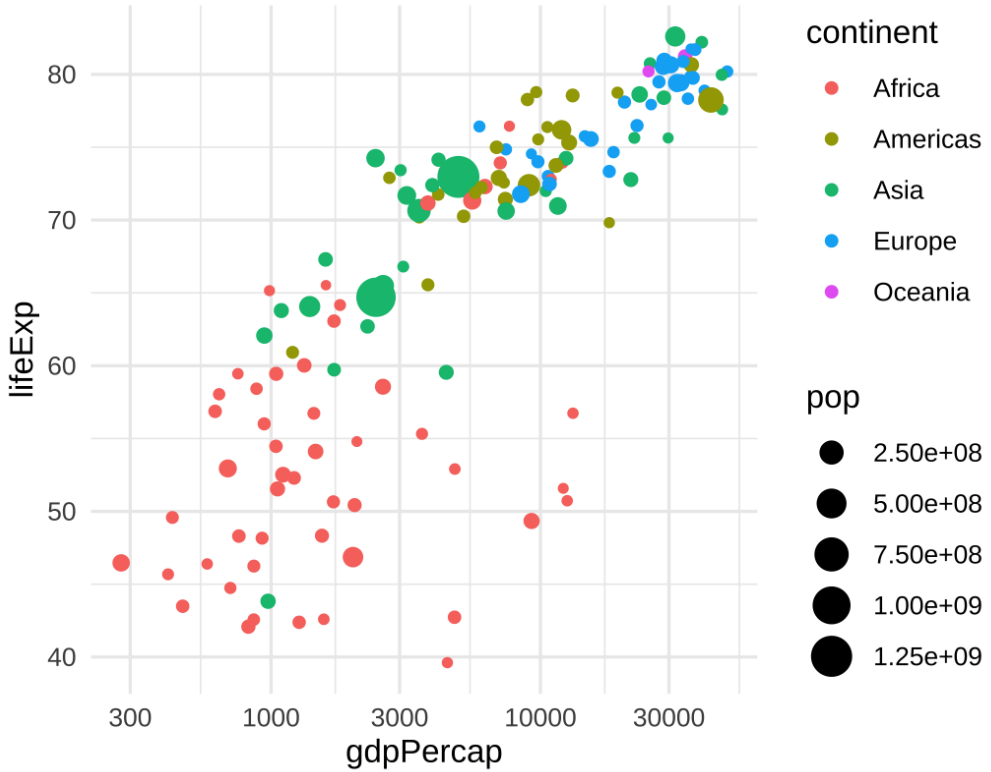
Example layer	What it does
`theme_grey()`	Default grey background
`theme_bw()`	Black and white
`theme_dark()`	Dark
`theme_minimal()`	Minimal

Theme

`theme_dark()`



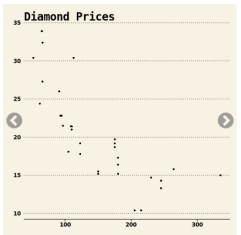
`theme_minimal()`



Theme

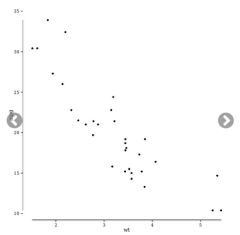
There are collections of pre-built themes online, like the **ggthemes** package

ggthemes



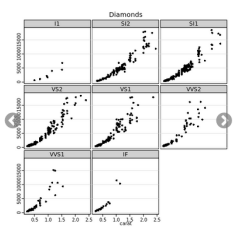
theme_wsj

Wall Street Journal theme



theme_tufte

Tufte Maximal Data, Minimal Ink Theme

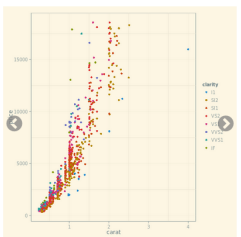


theme_stata

Themes based on Stata graph schemes

theme_solid

Theme with nothing other than a background color

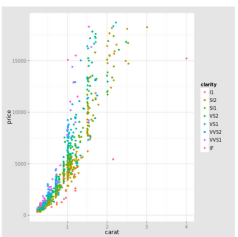


theme_solarized

ggplot color themes based on the Solarized palette

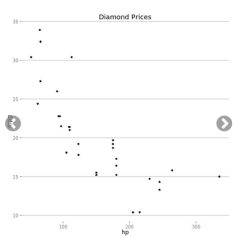
theme_map

Clean theme for maps



theme_igray

Inverse gray theme

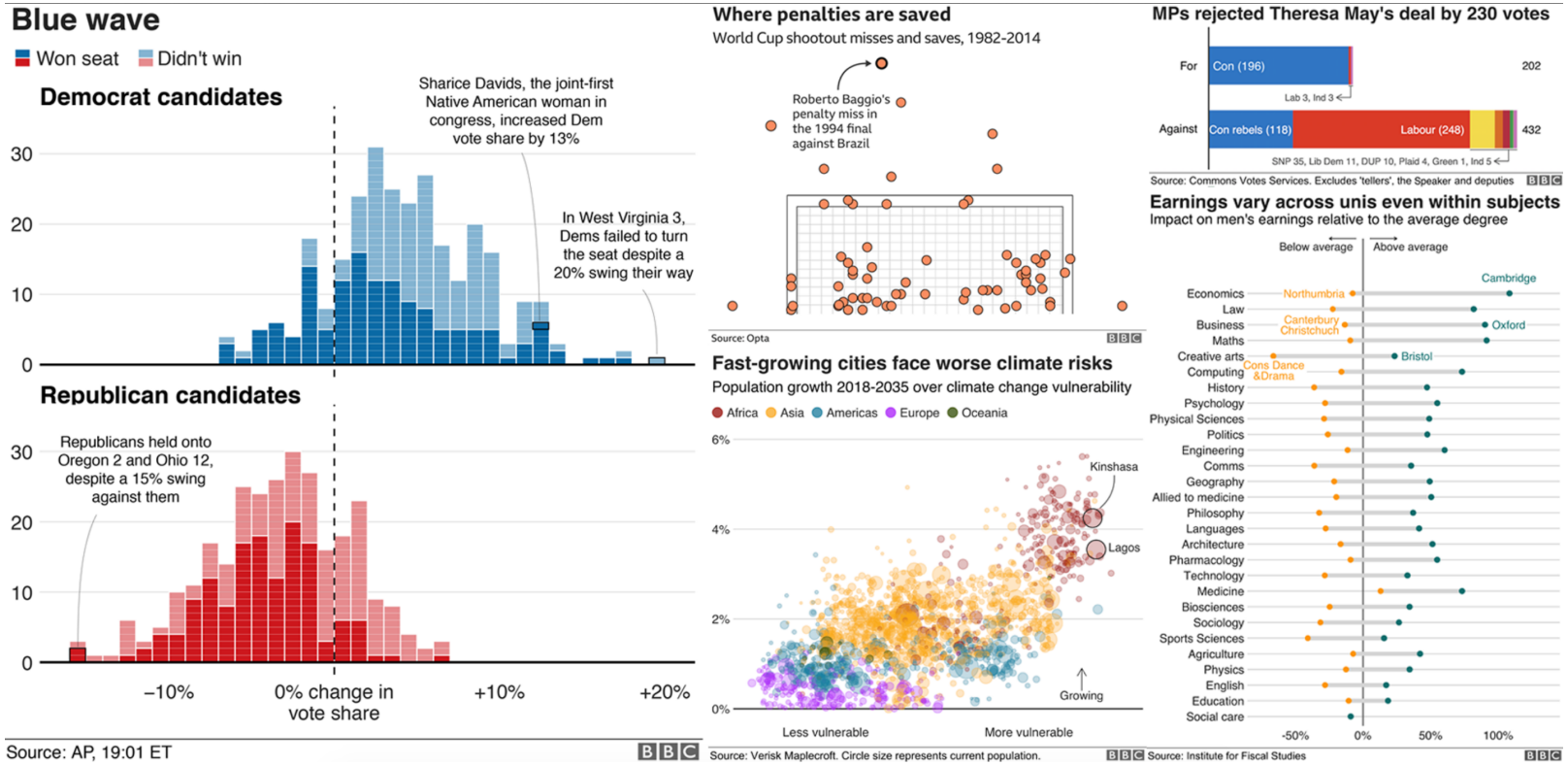


theme_hc

Highcharts JS theme

Theme

Organizations often make their own custom themes, like the BBC

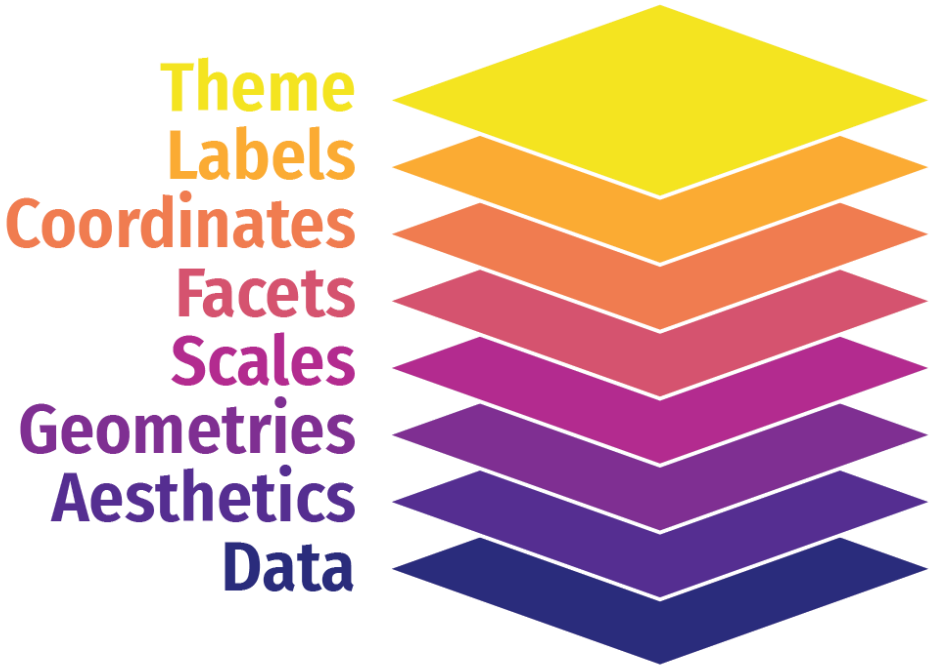


Theme

Make individual theme adjustments with `theme()`

```
theme_bw() +
theme(legend.position = "bottom",
      plot.title = element_text(face = "bold"),
      panel.grid = element_blank(),
      axis.title.y = element_text(face = "italic"))
```

So many possibilities!



These were just a few examples

See the **ggplot2** documentation for examples of everything you can do

Putting it all together

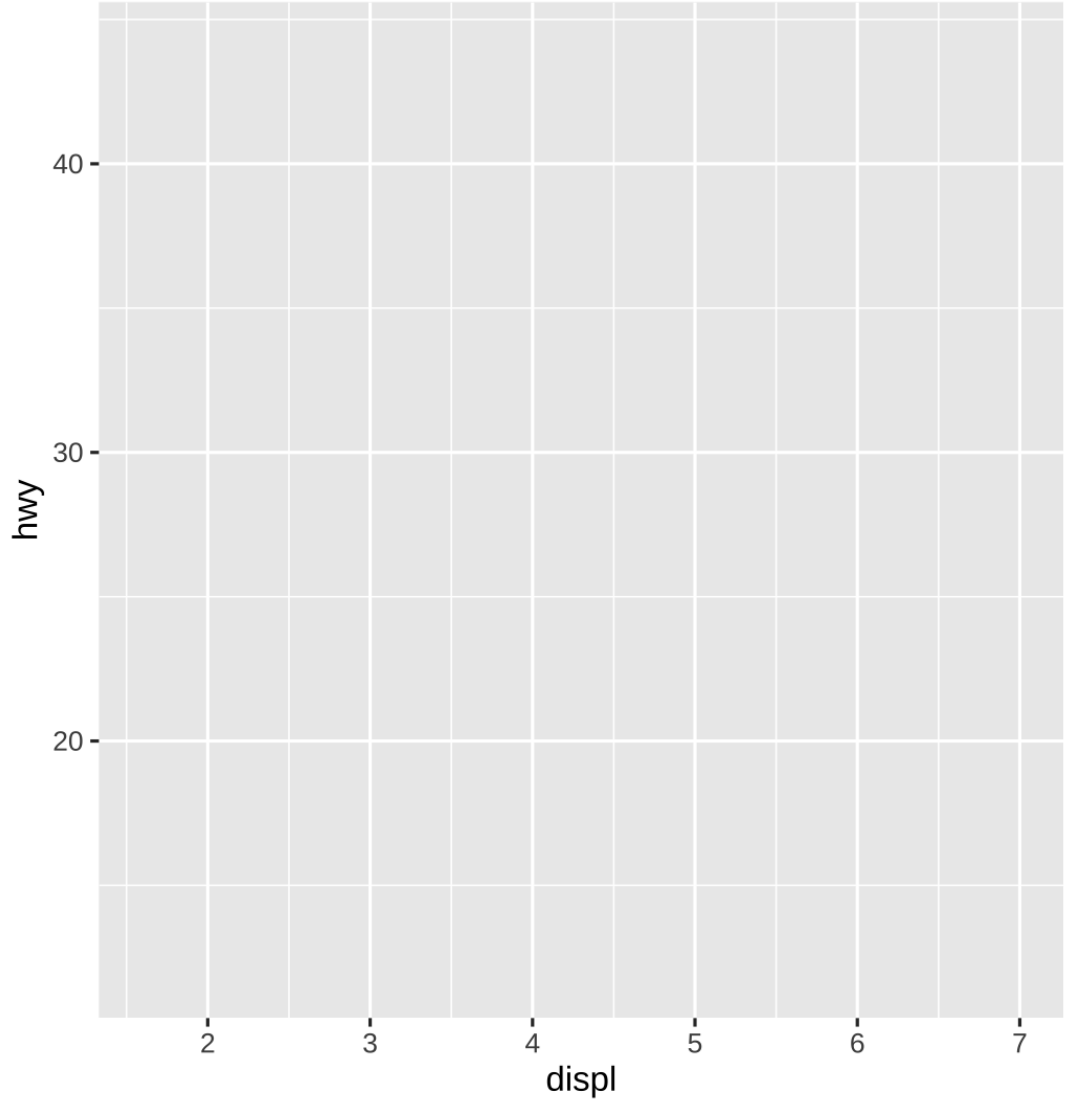
We can build a plot sequentially to see how each grammatical layer changes its appearance

For illustrative purposes, let's use data on cars from the data frame `ggplot2::mpg`

manufacturer	model	year	trans	displ	hwy	drv
volkswagen	gti	1999	manual(m5)	2	29	f
volkswagen	gti	1999	auto(l4)	2	26	f
...	...	...	...	...	...	...

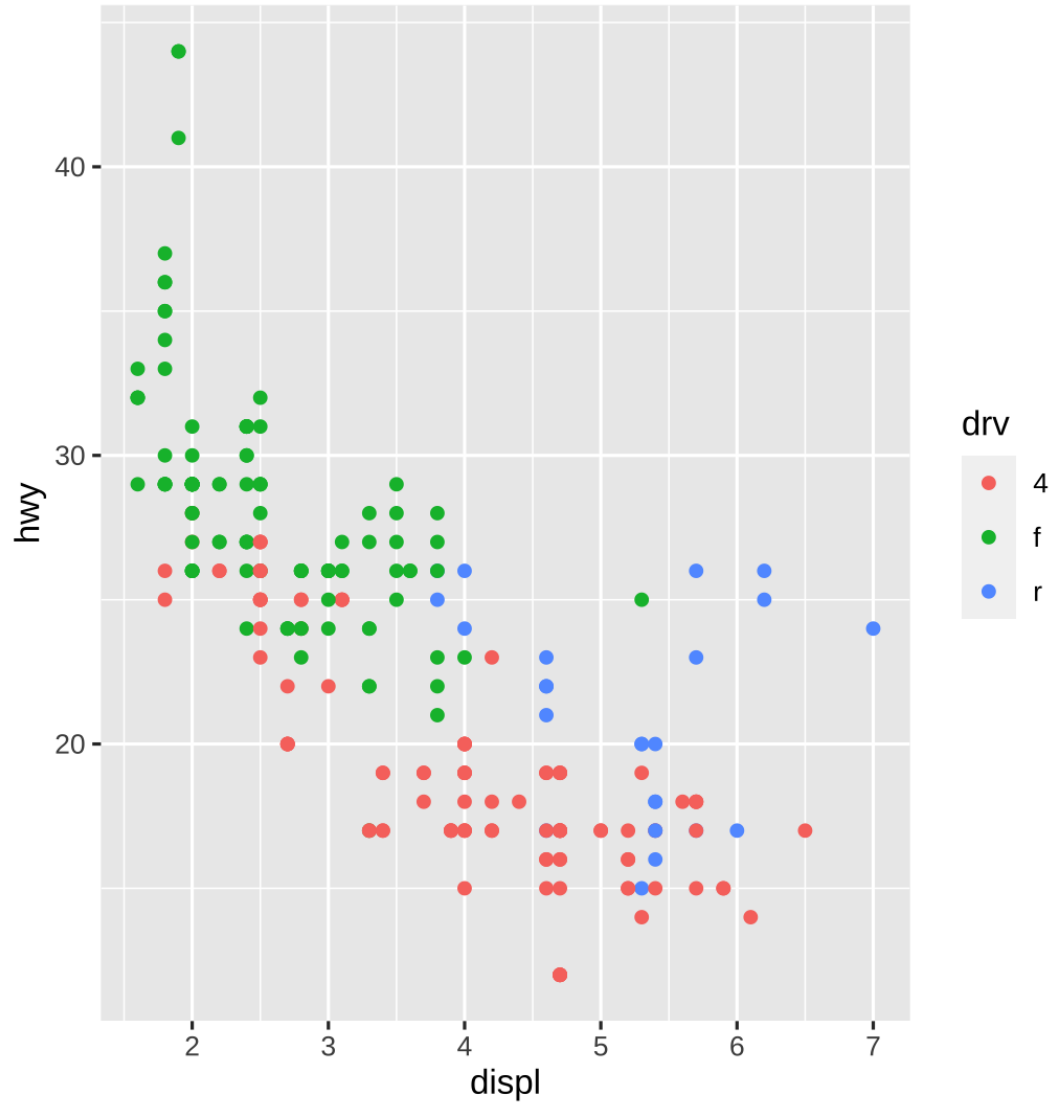
Start with data and aesthetics

```
ggplot(data = mpg,
       aes(x = displ, # engine displacement
           y = hwy, # highway mpg
           color = drv))  # type of drive train
```



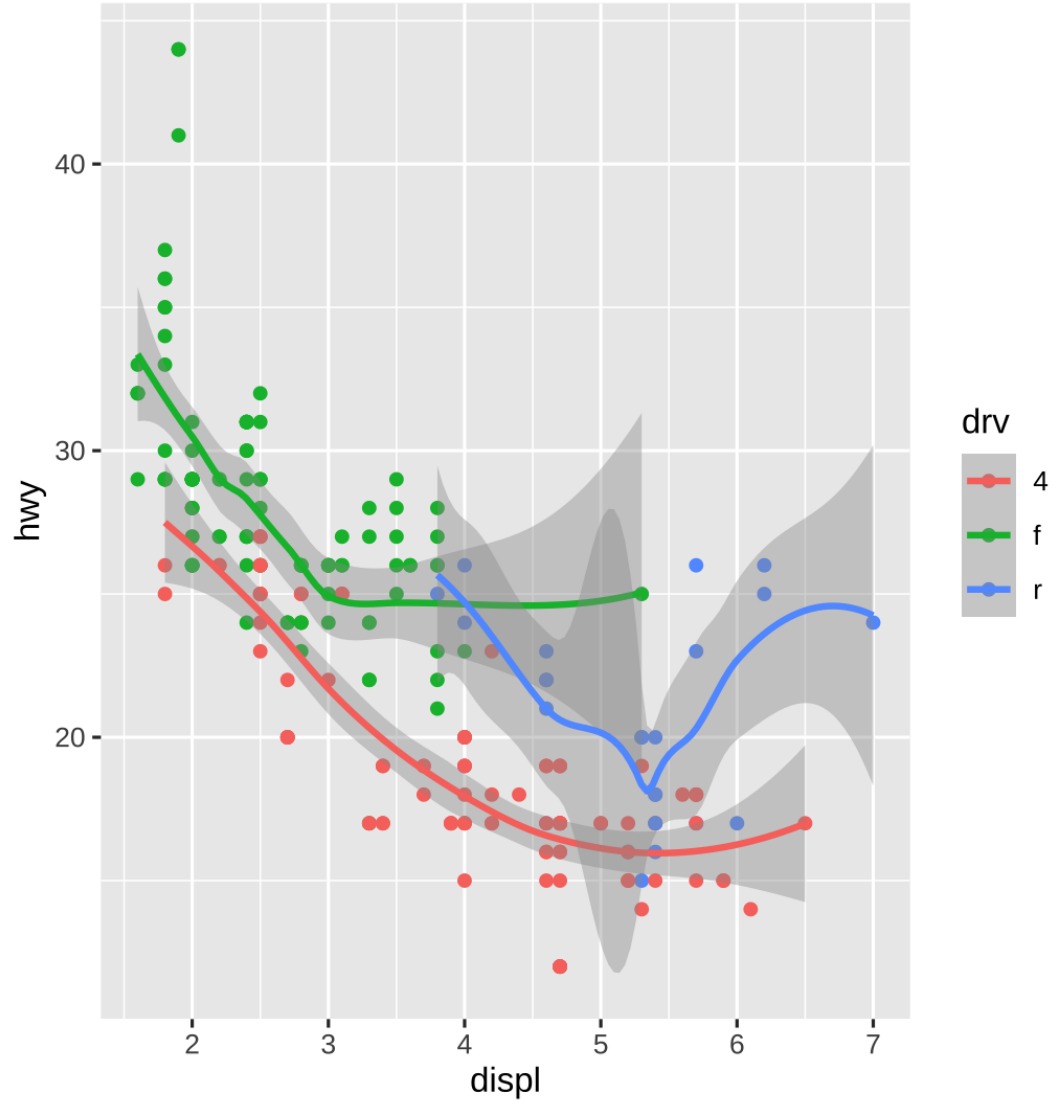
Add a point geom

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



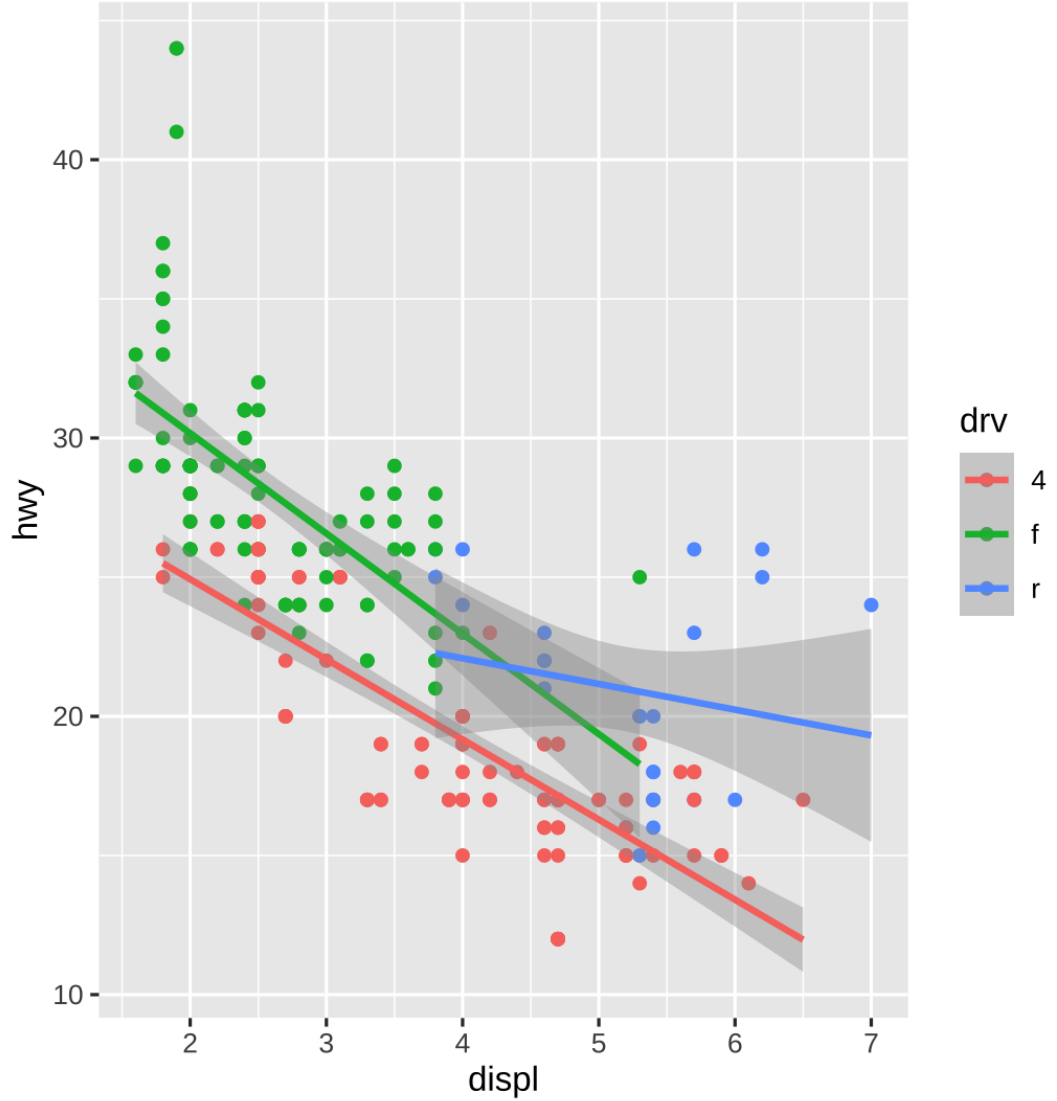
Add a smooth geom

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



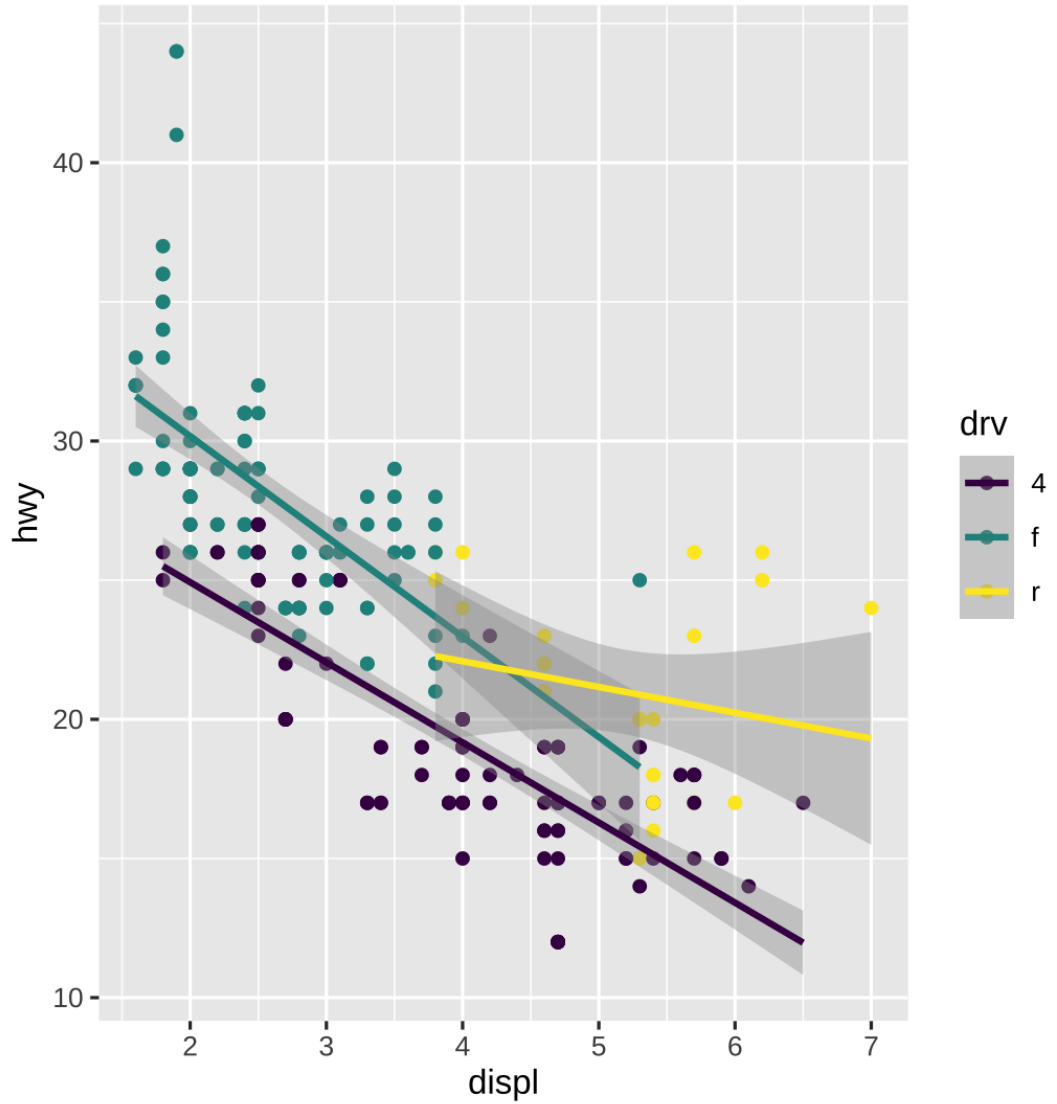
Make it straight

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



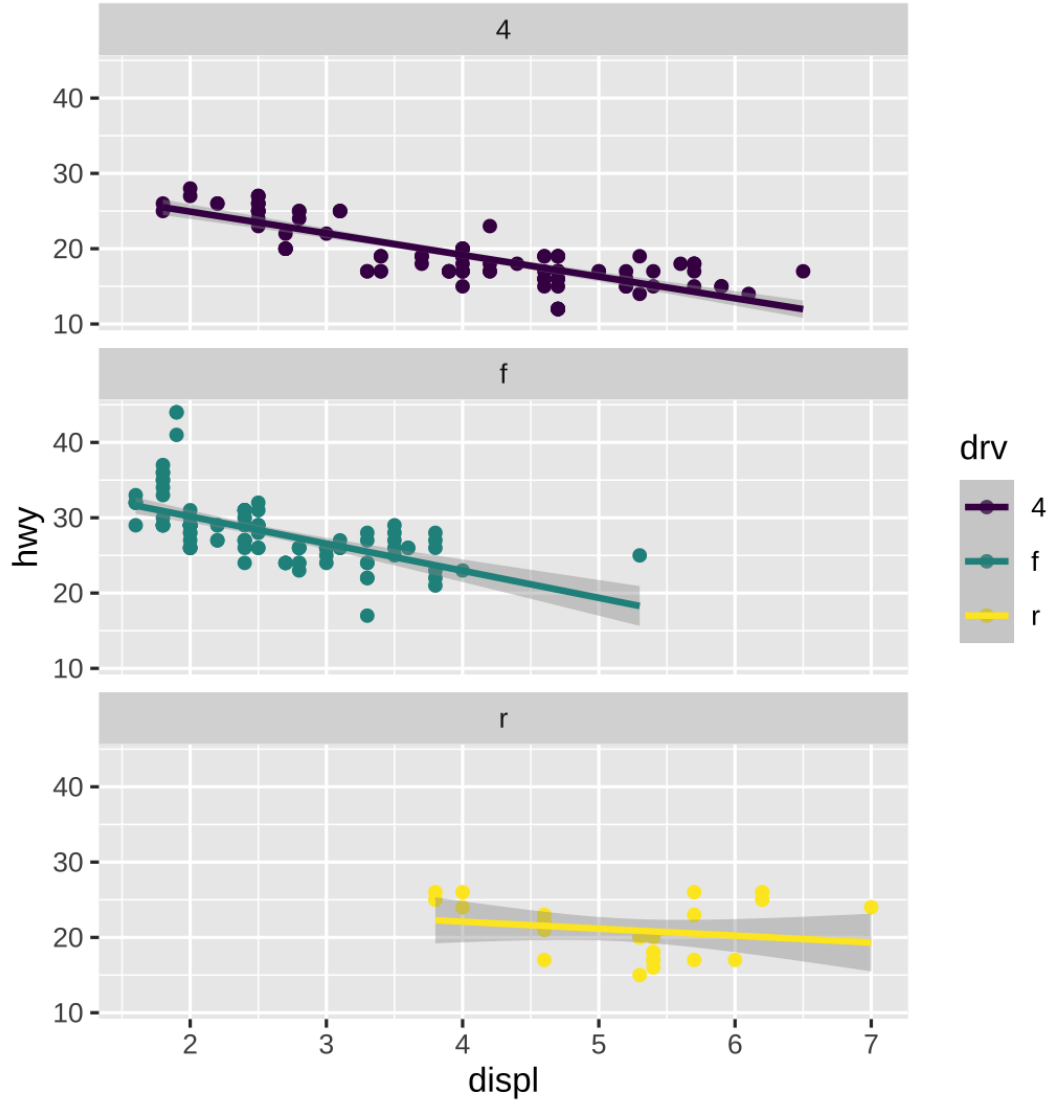
Use a viridis color scale

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



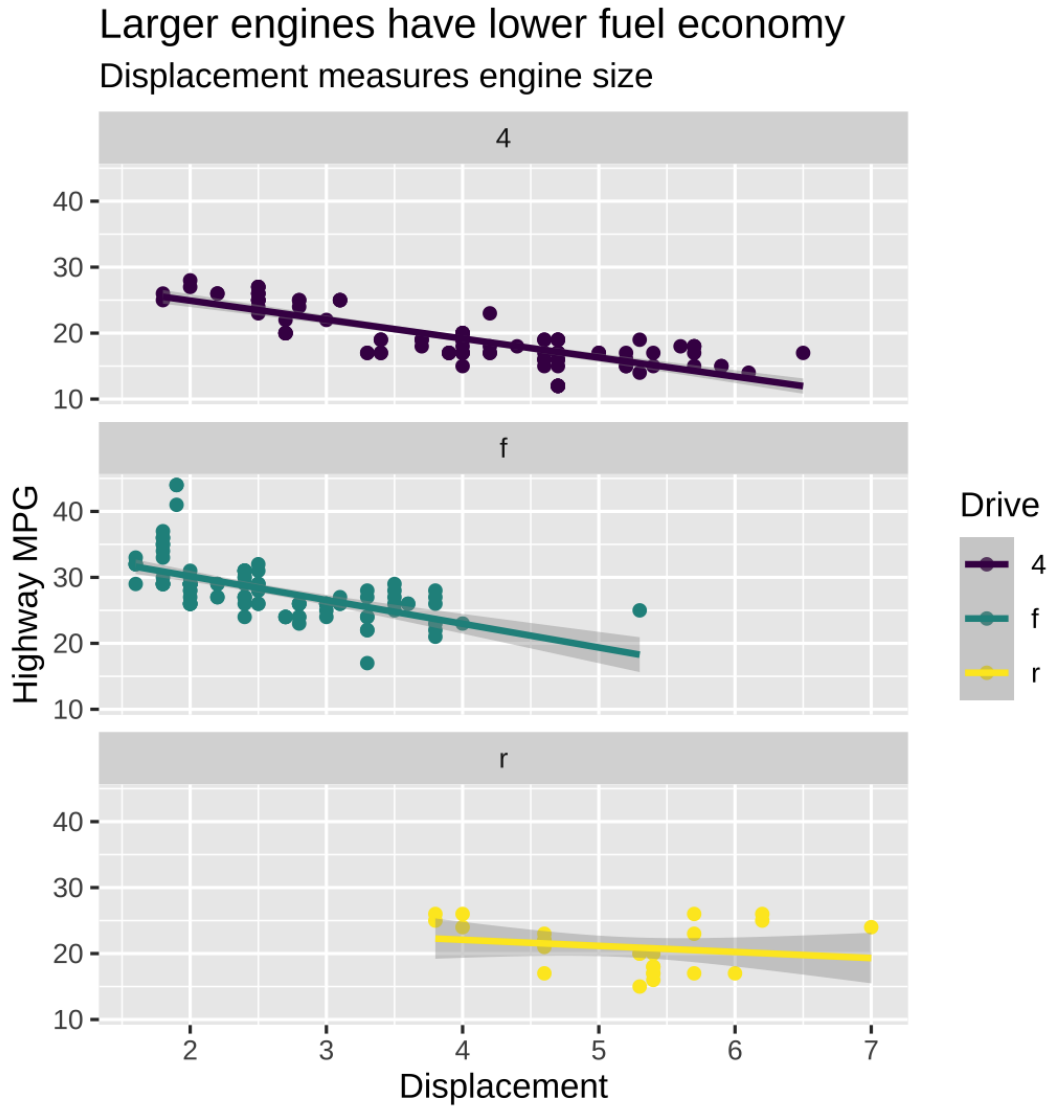
Facet by drive train type

```
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)
```



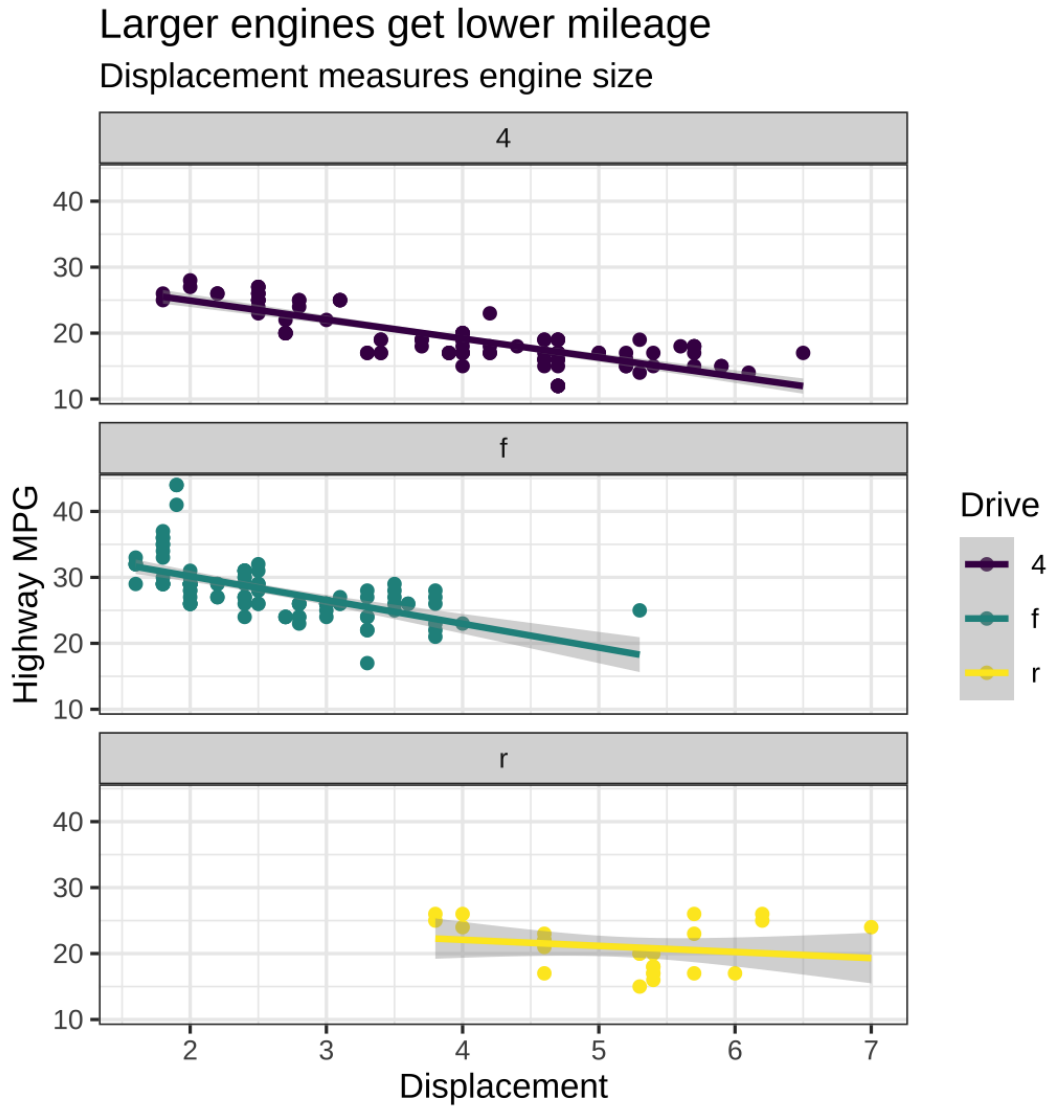
Add labels

```
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 = "Larger engines have lower fuel ecor
       subtitle = "Displacement measures engine siz
```



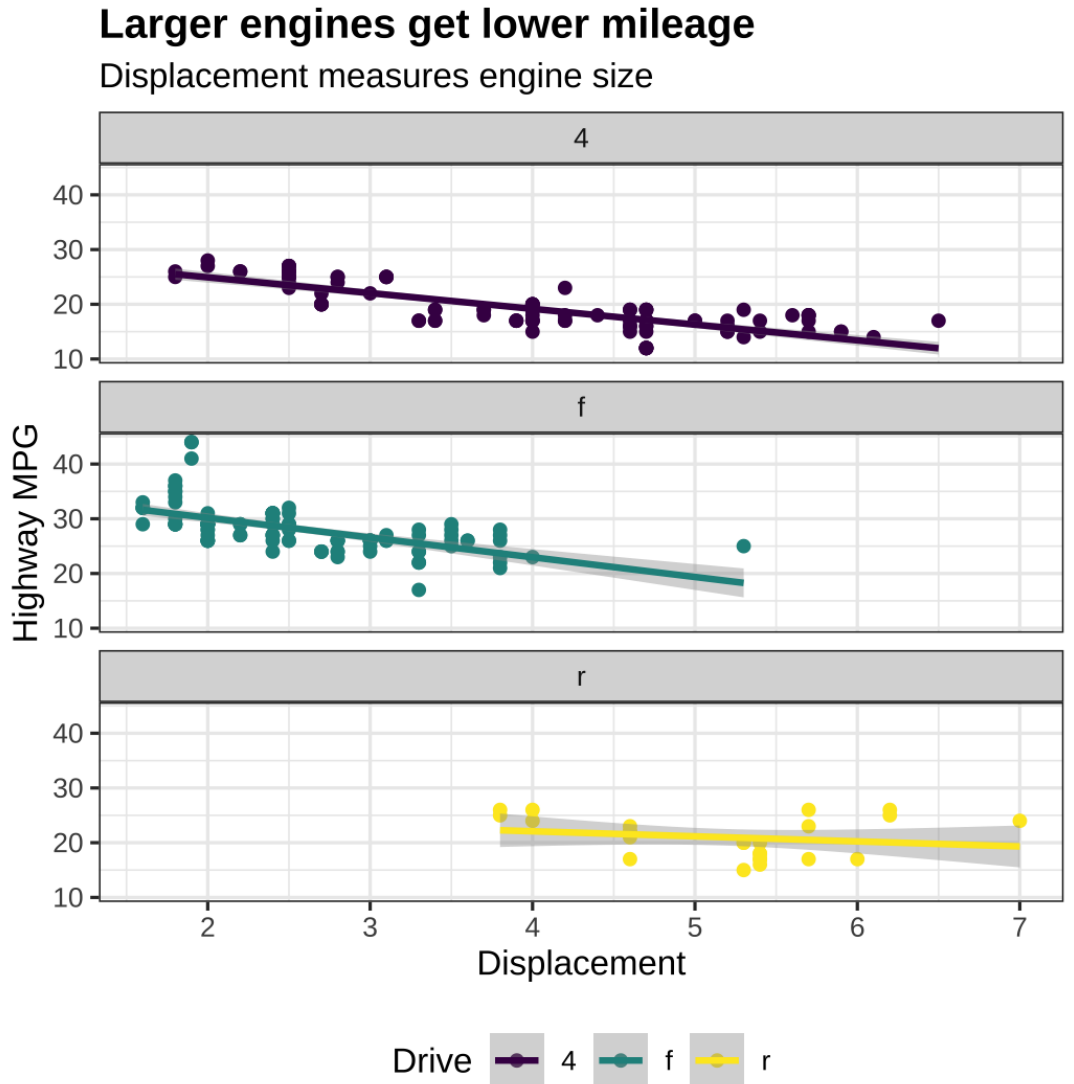
Add a theme

```
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 = "Larger engines get lower mileage",
       subtitle = "Displacement measures engine siz
  theme_bw()
```



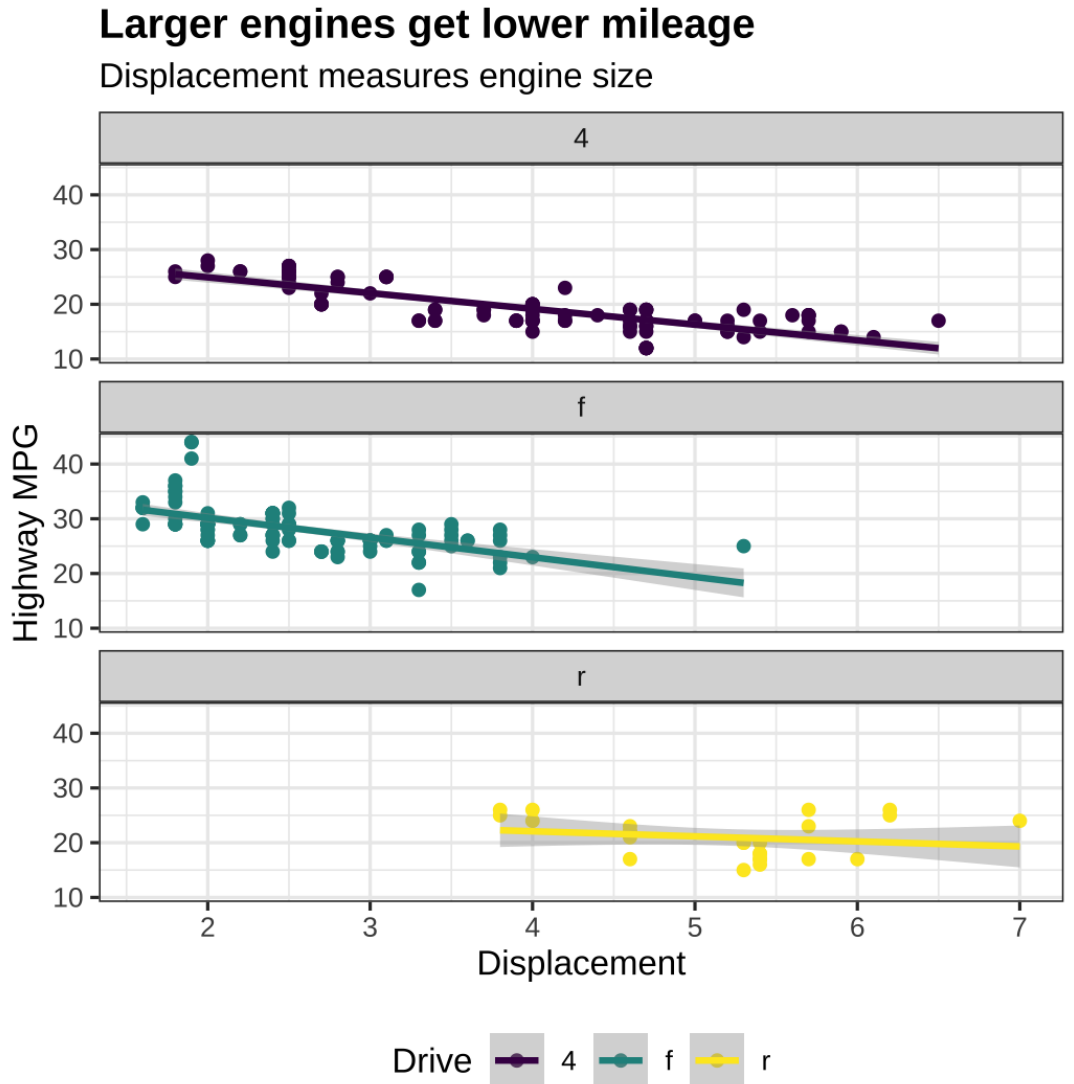
Modify the theme

```
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 = "Larger engines get lower mileage",
       subtitle = "Displacement measures engine siz
  theme_bw() +
  theme(legend.position = "bottom",
        plot.title = element_text(face = "bold"))
```



Finished!

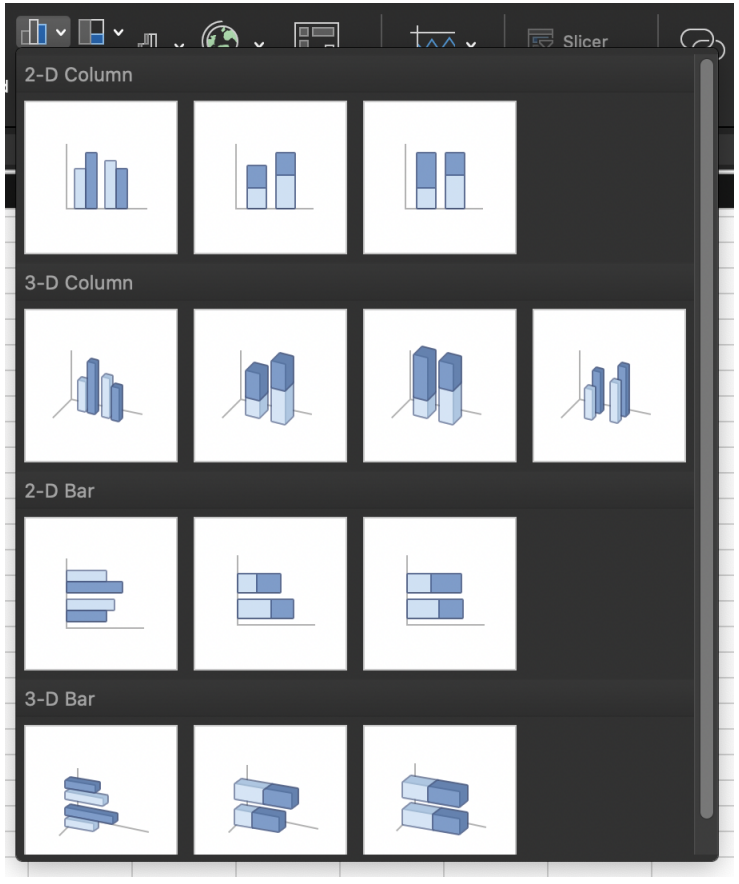
```
ggplot(data = mpg,
       mapping = aes(x = displ,
                     y = hwy,
                     color = drv)) +
  geom_point() +
  geom_smooth(method = "lm") +
  scale_color_viridis_d() +
  facet_wrap(vars(drv), ncol = 1) +
  labs(x = "Displacement", y = "Highway MPG",
       color = "Drive",
       title = "Larger engines get lower mileage",
       subtitle = "Displacement measures engine siz
  theme_bw() +
  theme(legend.position = "bottom",
        plot.title = element_text(face = "bold"))
```



A true grammar

You have probably made graphics in Excel by searching through menus for a specific chart type, and then reshaping your data to work with it

With the grammar of graphics, we don't talk about specific chart **types**

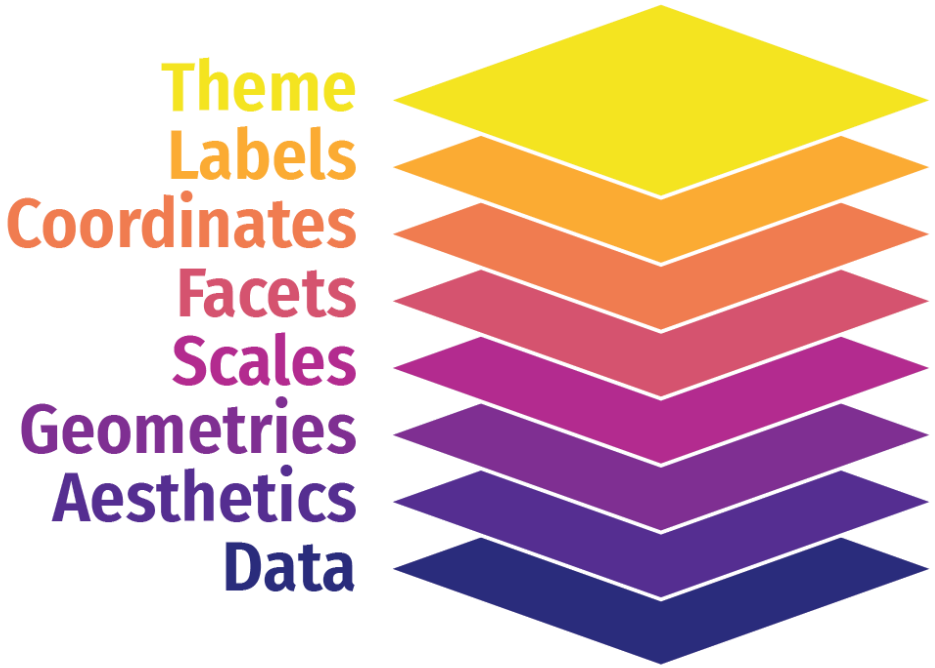


A true grammar

With the grammar of graphics, we talk about specific chart **elements**

- Map a column to the x-axis, fill by a variable, `geom_col()` to get stacked bars
- Geoms can be interchangable (e.g., `geom_violin()` and `geom_boxplot()`)

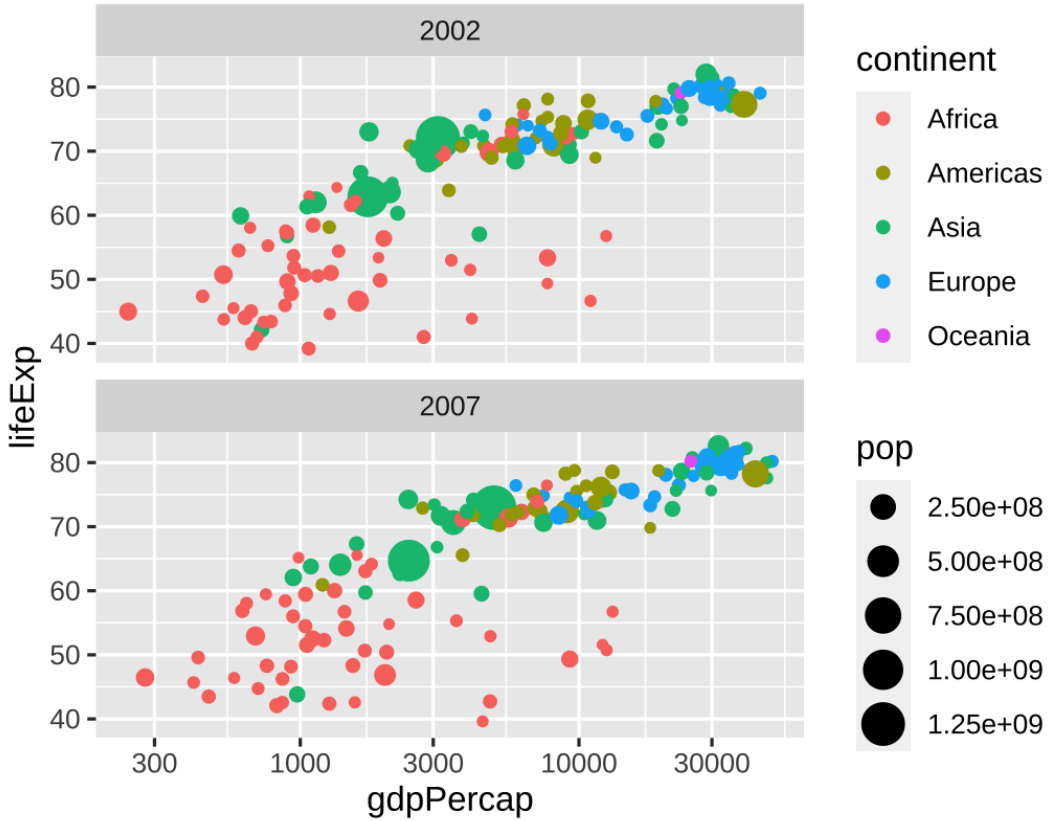
This grammar is portable to other software like Tableau, which is built on the grammar of graphics



Describing graphs with the grammar

Map wealth to the x-axis, health to the y-axis, add points, color by continent, size by population, scale the y-axis with a log, and facet by year

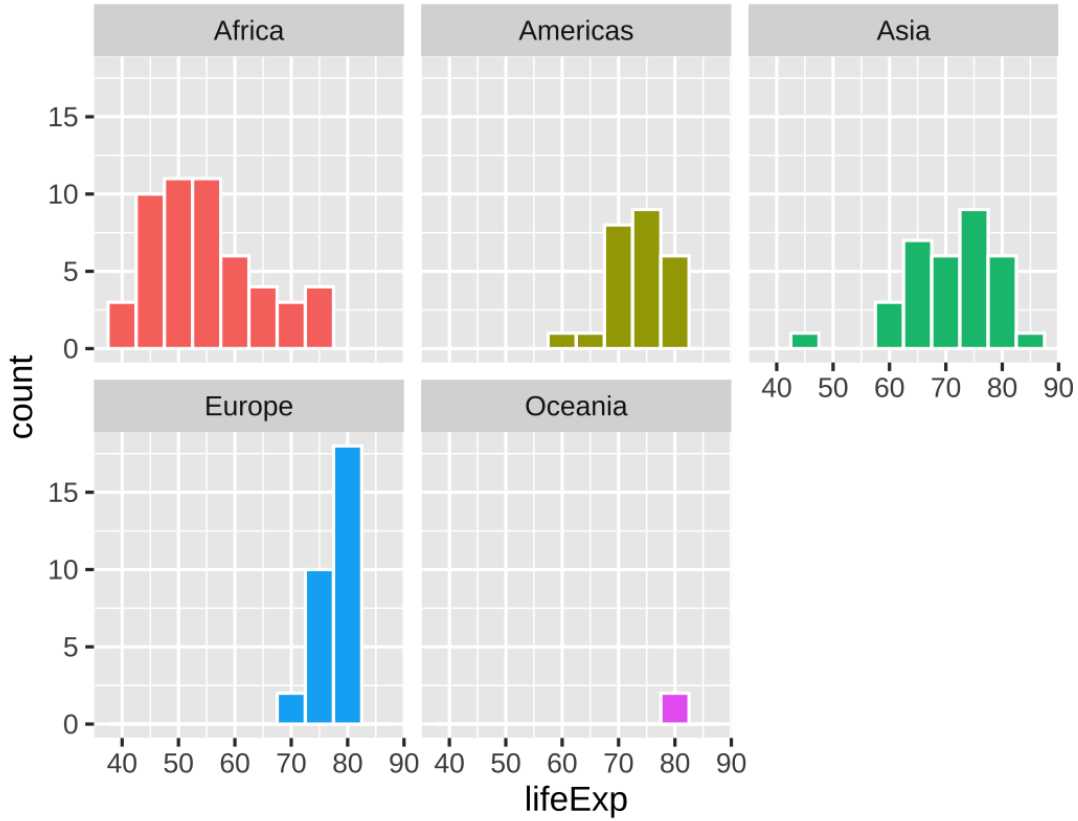
```
ggplot(data = filter(gapminder, year %in% c(2002, 2
       mapping = aes(x = gdpPercap,
                     y = lifeExp,
                     color = continent,
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  facet_wrap(vars(year), ncol = 1)
```



Describing graphs with the grammar

Map health to the x-axis, add a histogram with bins for every 5 years, fill and facet by continent

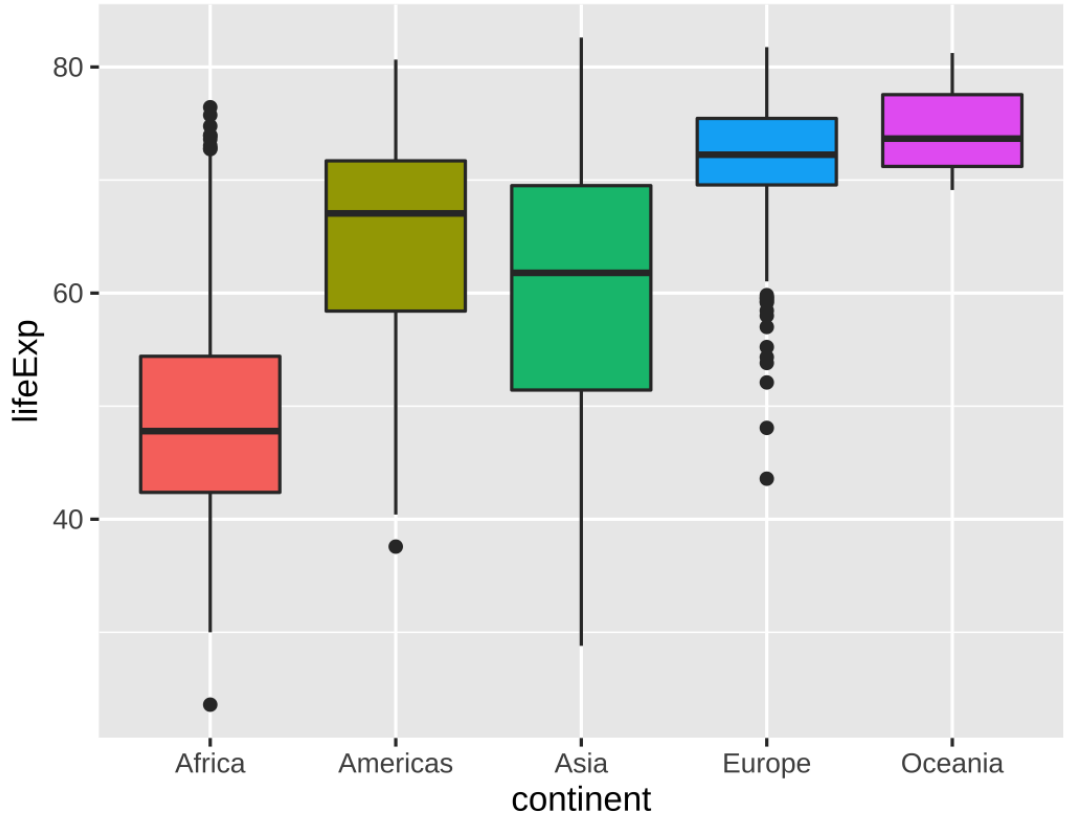
```
ggplot(data = gapminder_2007,
       mapping = aes(x = lifeExp,
                     fill = continent)) +
  geom_histogram(binwidth = 5,
                 color = "white") +
  guides(fill = "none") +  # Turn off legend
  facet_wrap(vars(continent))
```



Describing graphs with the grammar

Map continent to the x-axis, health to the y-axis, add boxplots, fill by continent

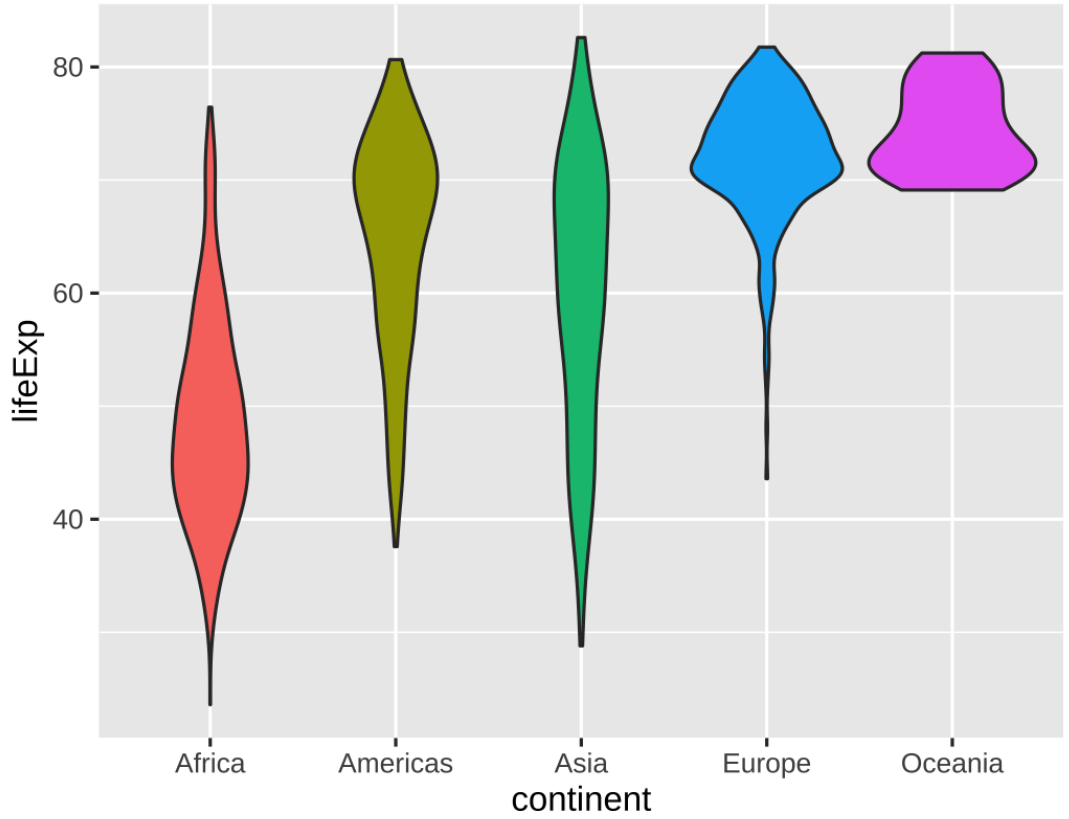
```
ggplot(data = gapminder,
       mapping = aes(x = continent,
                     y = lifeExp,
                     fill = continent)) +
  geom_boxplot() +
  guides(fill = "none")  # Turn off legend
```



Describing graphs with the grammar

Map continent to the x-axis, health to the y-axis, add violin plots, fill by continent

```
ggplot(data = gapminder,
       mapping = aes(x = continent,
                     y = lifeExp,
                     fill = continent)) +
  geom_violin() + # we only had to change this!
  guides(fill = "none")  # Turn off legend
```



Tidy data revisited

Tidy data revisited

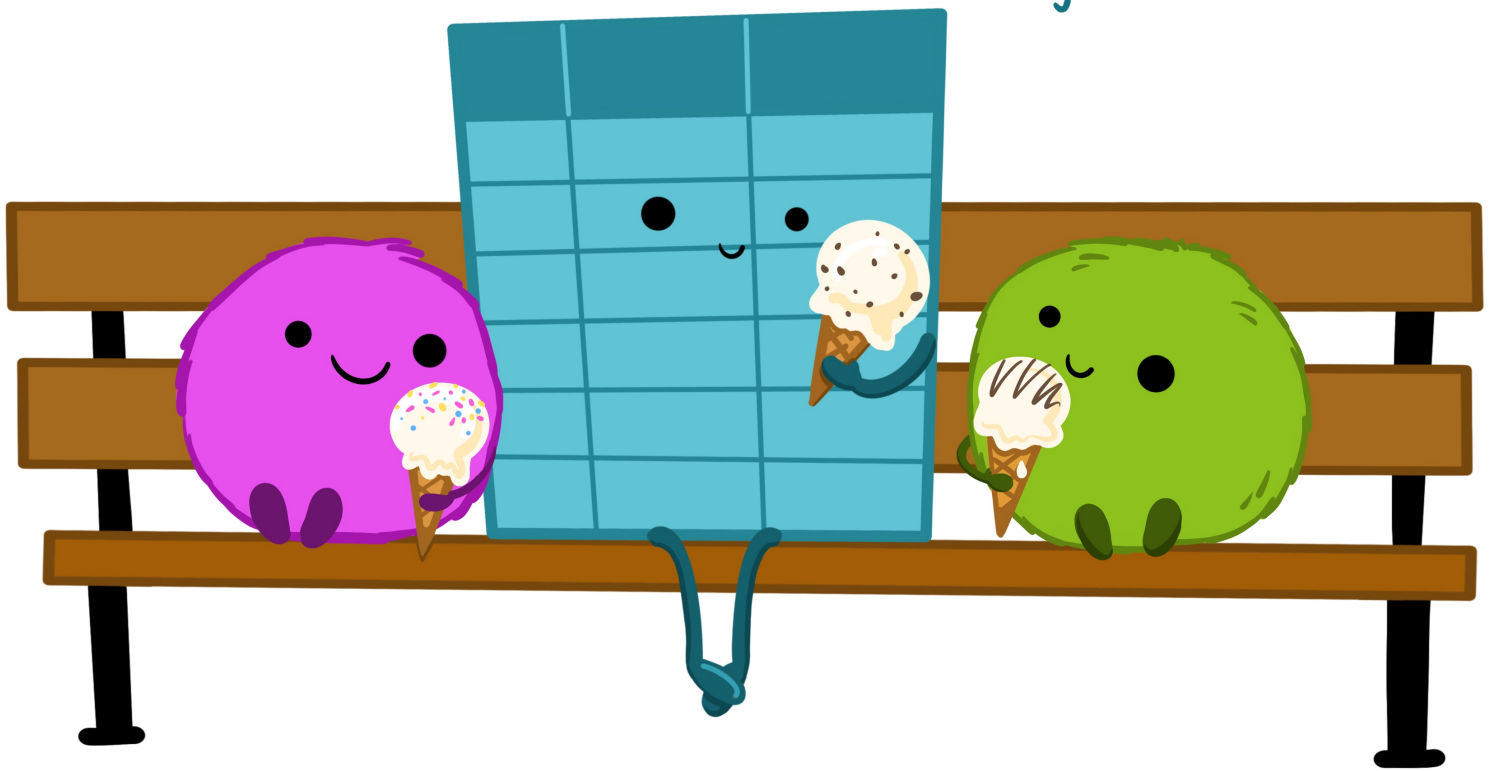
`ggplot()` is incredibly flexible

For `ggplot()` to work its magic, your data frame needs to be **tidy**

This requires a bit of upfront work, but it pays off

Note that we never modified `mpg` or `gapminder` in the process of making a dozen different plots

make friends with tidy data.

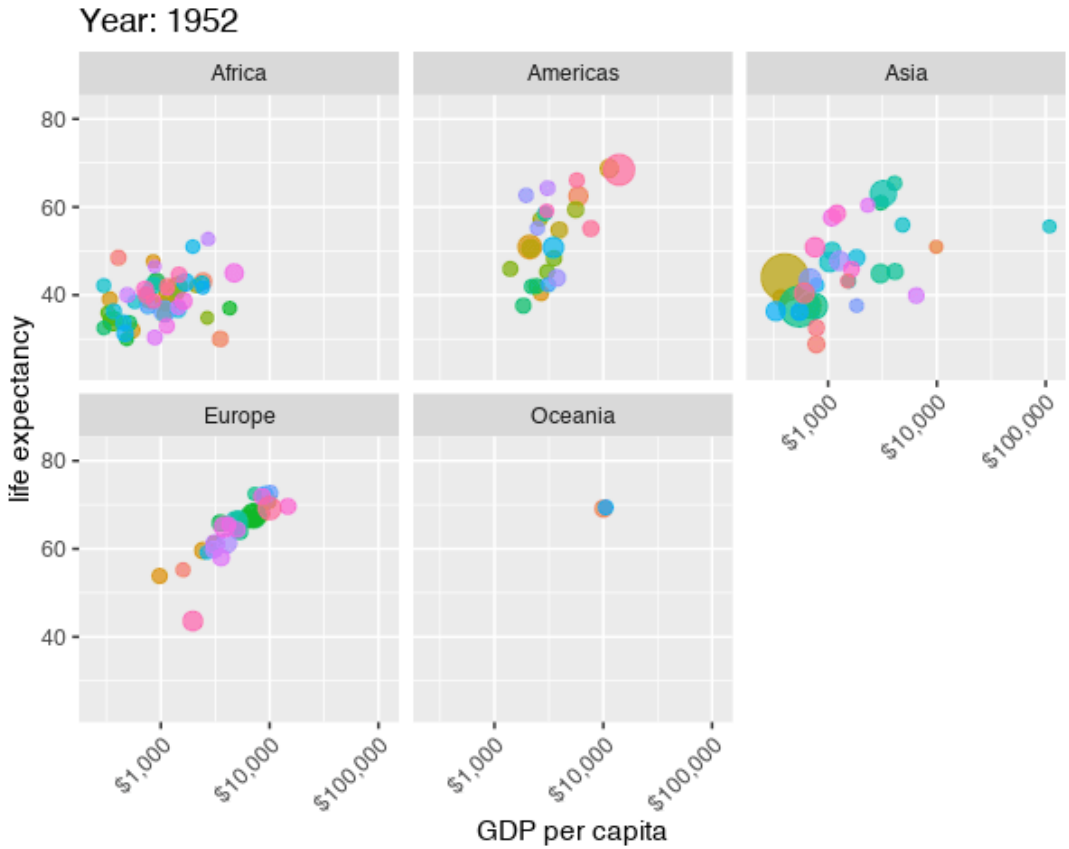


Aesthetics in extra dimensions

Time

Use **gganimate** to map variables to a time aesthetic

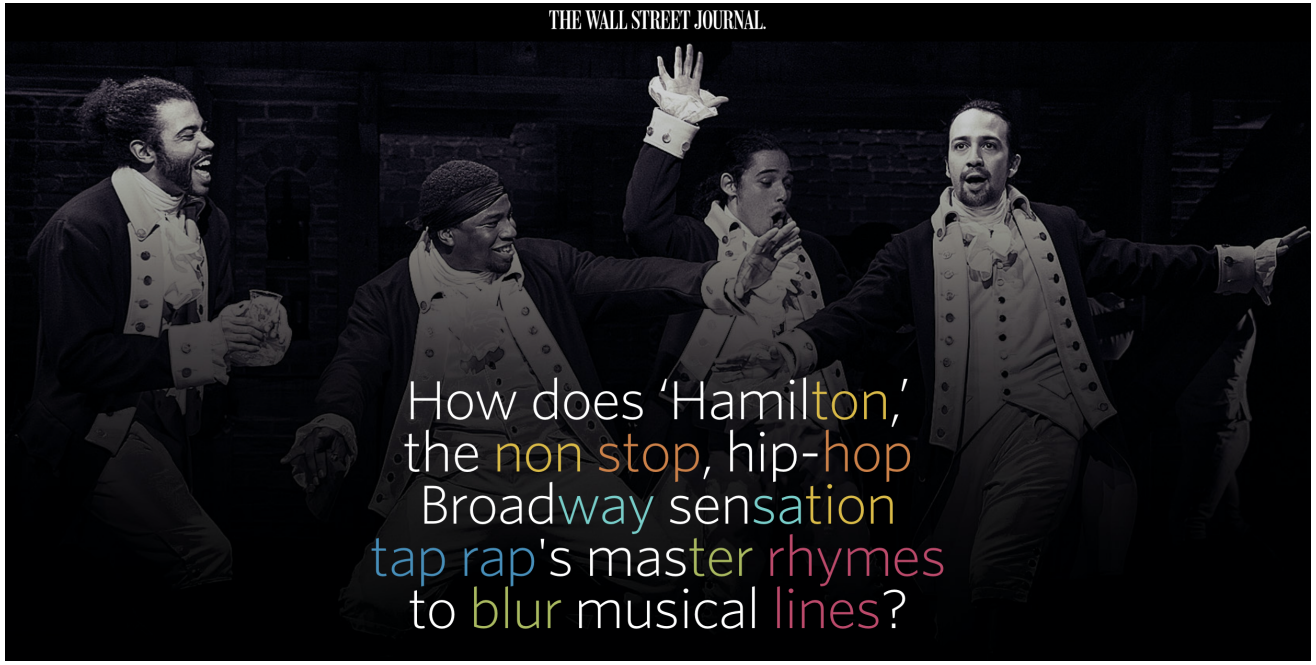
```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp,
                      size = pop, color = country))
  geom_point(alpha = 0.7) +
  scale_size(range = c(2, 12)) +
  scale_x_log10(labels = scales::dollar) +
  guides(size = "none", color = "none") +
  facet_wrap(~continent) +
  # Special gganimate stuff
  labs(title = 'Year: {frame_time}',
       x = 'GDP per capita',
       y = 'life expectancy') +
  transition_time(year) +
  ease_aes('linear')
```



Sound

Combine audio and visual to highlight internal rhyming schemes in music

http://graphics.wsj.com/hamilton/



Sound

Daveed Diggs

I'm **in** the **cabinet** I **am** com**plicit in**
Watching him grabbing at power **and kissing it**
If Washington isn't gon' listen
To **disciplined dissidents this is** the **difference**
This kid **is** out

Kendrick Lamar

Trapped inside your de**sire** to **fire** bu**llets that stray**
Track attire just tell **you** I'm **tired** and ran **away**
I should **ask** a **choir "What do you require**
to sing a song **that acquire me** to have **faith?"**

"good kid" on "good kid, m.A.A.d city"

Animation, time, and sound