Mapping data to graphics and amounts

Week 7

AEM 2850 / 5850 : R for Business Analytics

Cornell Dyson

Spring 2023

Acknowledgements: Andrew Heiss, Claus Wilke

Announcements

Today marks the start of the data visualization component of this course

See aem2850.toddgerarden.com/schedule for minor schedule updates

I will discuss prelim 1 at the end of class today

Questions before we get started?

Plan for this week

Tuesday

- Course progress
- Prologue
- Data, aesthetics, & the grammar of graphics
- Grammatical layers
- Example (part 1)
- Prelim 1
- Reference: Additional layers

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 1-5: Programming Foundations

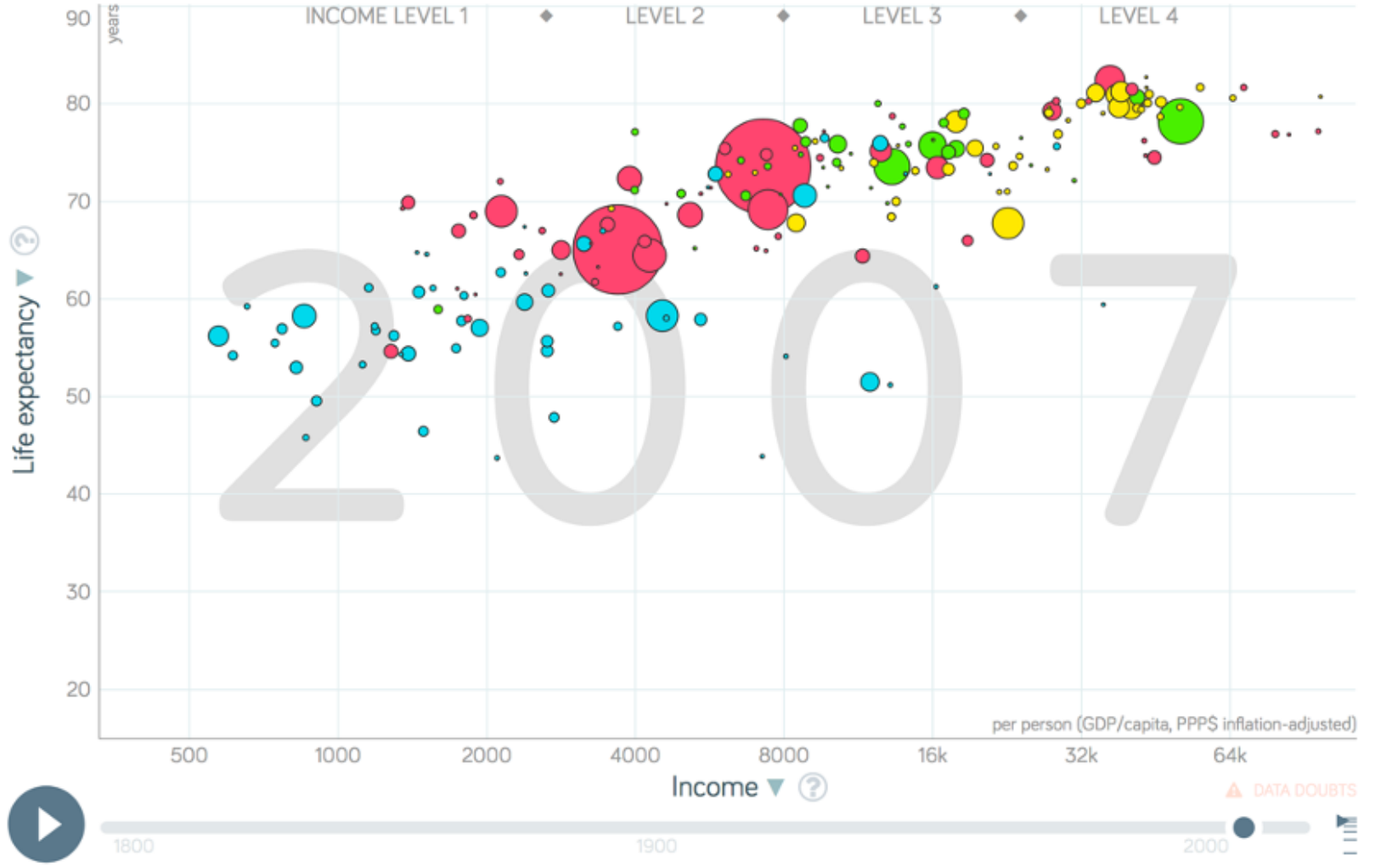
Weeks 7-10: Data Visualization Foundations

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

See aem2850.toddgerarden.com/schedule for details

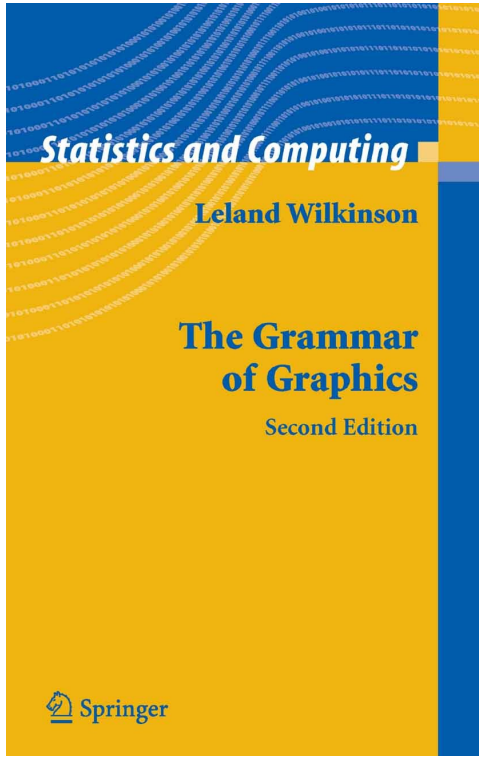
Prologue

Health and wealth



Data, aesthetics, & the grammar of graphics

Mapping data to aesthetics



Data

A column in a dataset

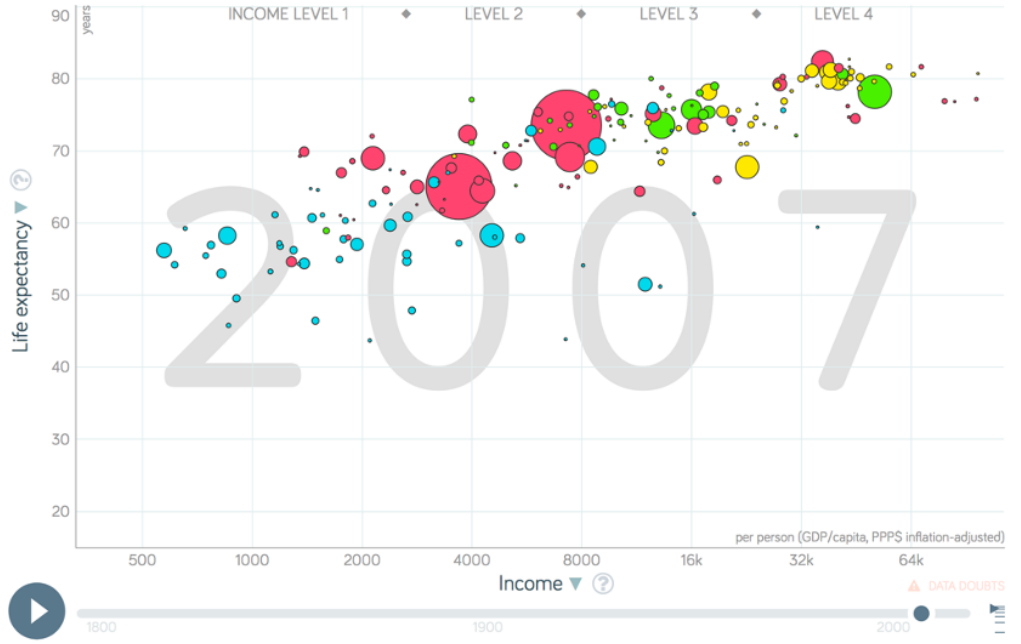
Aesthetics

Visual properties of a graph

Position, shape, color, etc.

Mapping data to aesthetics

Data	Aesthetic	Geometry
Wealth	Position (x)	Point
Health	Position (y)	Point
Continent	Color	Point
Population	Size	Point



Mapping data to aesthetics using ggplot

Data	Aesthetic	Geometry
Wealth	Position (x)	Point
Health	Position (y)	Point
Continent	Color	Point
Population	Size	Point

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

Mapping data to aesthetics using ggplot

Data	Aesthetic	Geometry
Wealth	Position (x)	Point
Health	Position (y)	Point
Continent	Color	Point
Population	Size	Point

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

Barebones `ggplot2::ggplot()` template

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

We need to specify data, aesthetic mapping, and geometry:

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

Or, in the context of a data wrangling pipeline:

```
DATA |>
  ... |> # intermediate data wrangling (optional)
  ggplot(aes(AESTHETIC MAPPINGS)) +
  GEOM_FUNCTION()
```

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(aes(x = _________,
             y = _______,
             color = _________,
             size = ___)) +
  geom______() +
  scale_x_log10() + theme_gray(base_size = 20) # ignore this line for now
```

Let's fill in the blanks... how do we start?

Mapping from `gapminder` to aesthetics

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

```
gapminder |>
  ggplot(aes(x = _________,
             y = _______,
             color = _________,
             size = ___)) +
  geom______() +
  scale_x_log10() + theme_gray(base_size = 20) # ignore this line for now
```

Let's fill in the blanks... what should we map to x? y?

Mapping from `gapminder` to aesthetics

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

```
gapminder |>
  ggplot(aes(x = gdpPercap,
             y = lifeExp,
             color = _________,
             size = ___)) +
  geom______() +
  scale_x_log10() + theme_gray(base_size = 20) # ignore this line for now
```

Let's fill in the blanks... what should we map to color? size?

Mapping from `gapminder` to aesthetics

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

```
gapminder |>
  ggplot(aes(x = gdpPercap,
             y = lifeExp,
             color = continent,
             size = pop)) +
  geom______() +
  scale_x_log10() + theme_gray(base_size = 20) # ignore this line for now
```

Let's fill in the blanks... what geometry should we use?

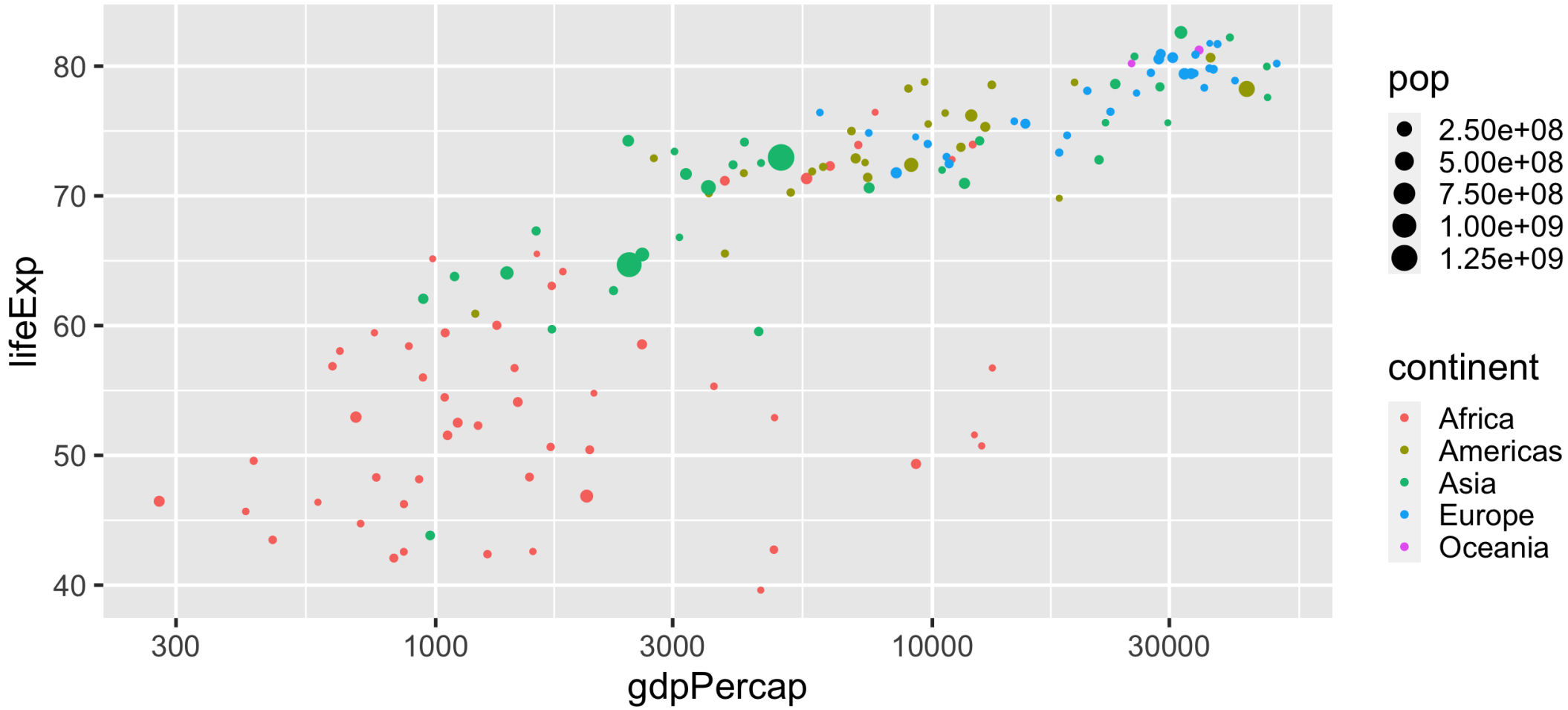
Mapping from `gapminder` to aesthetics

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

```
gapminder |>
  ggplot(aes(x = gdpPercap,
             y = lifeExp,
             color = continent,
             size = pop)) +
  geom_point() +
  scale_x_log10() + theme_gray(base_size = 20) # ignore this line for now
```

All done! Let's see what we get...

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 to foundational `ggplot()` with `+`

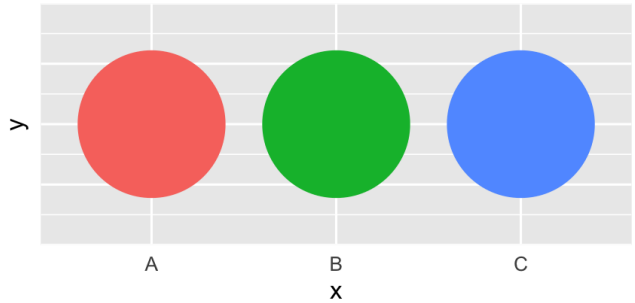
Why `+` and not `|>`?

> ggplot2 was written before the pipe was discovered

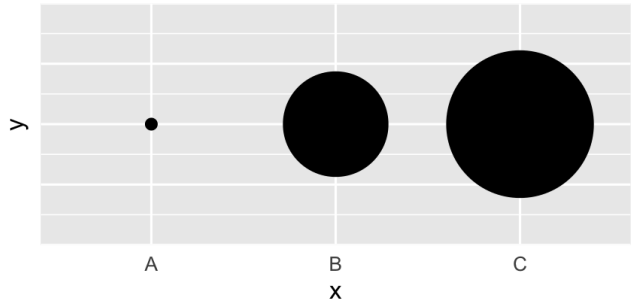
Treat the `+` the same as `|>`

Possible aesthetics

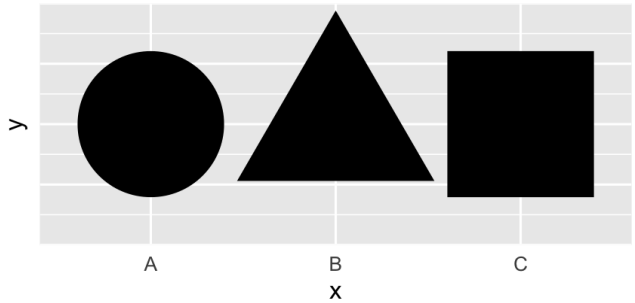
`color` (discrete)



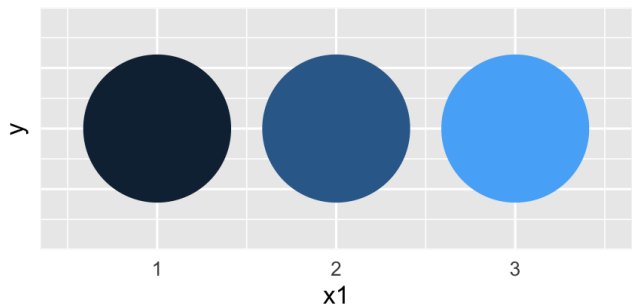
`size`



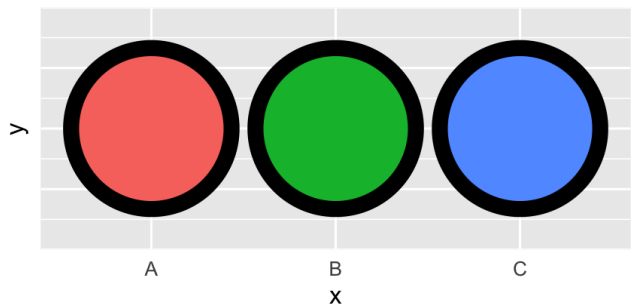
`shape`



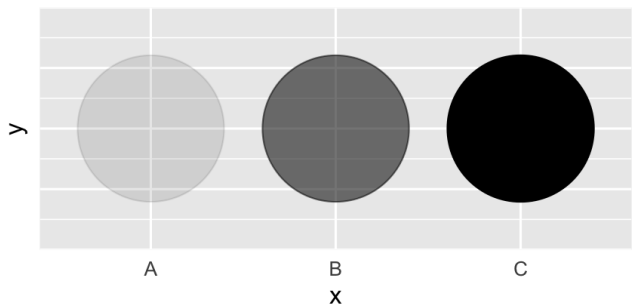
`color` (continuous)



`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

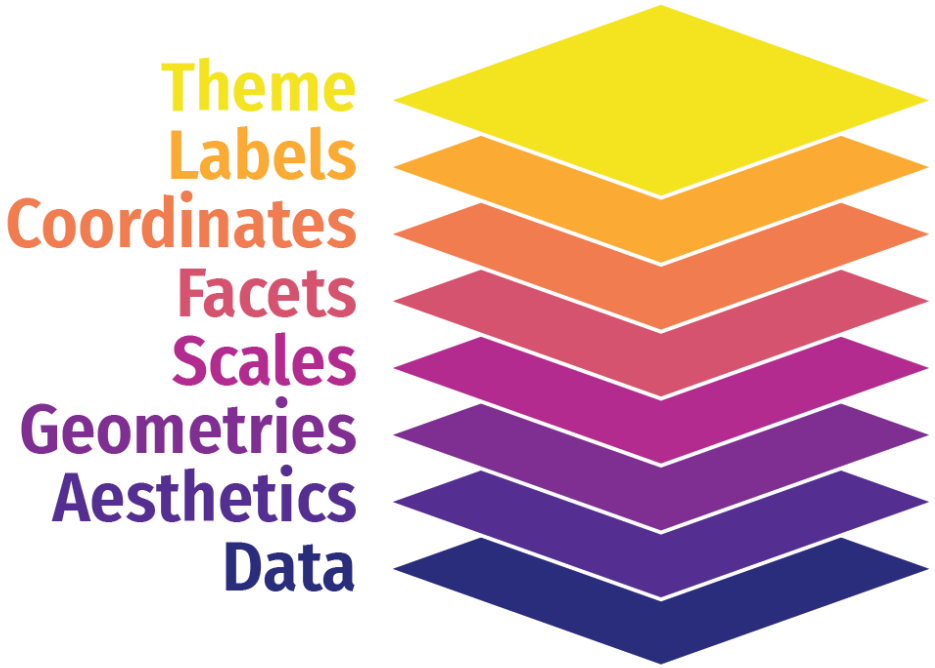
See the **ggplot2** documentation for examples of all the different geom layers

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

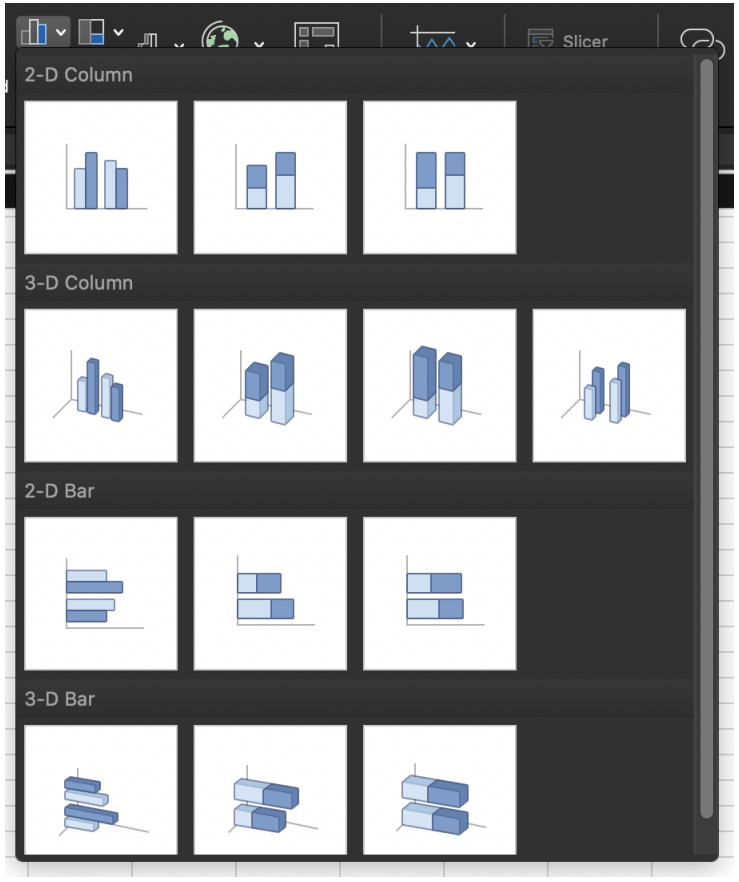
We will primarily learn by doing, though this slide deck contains a preview of additional layers in case you need some bedtime reading



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**

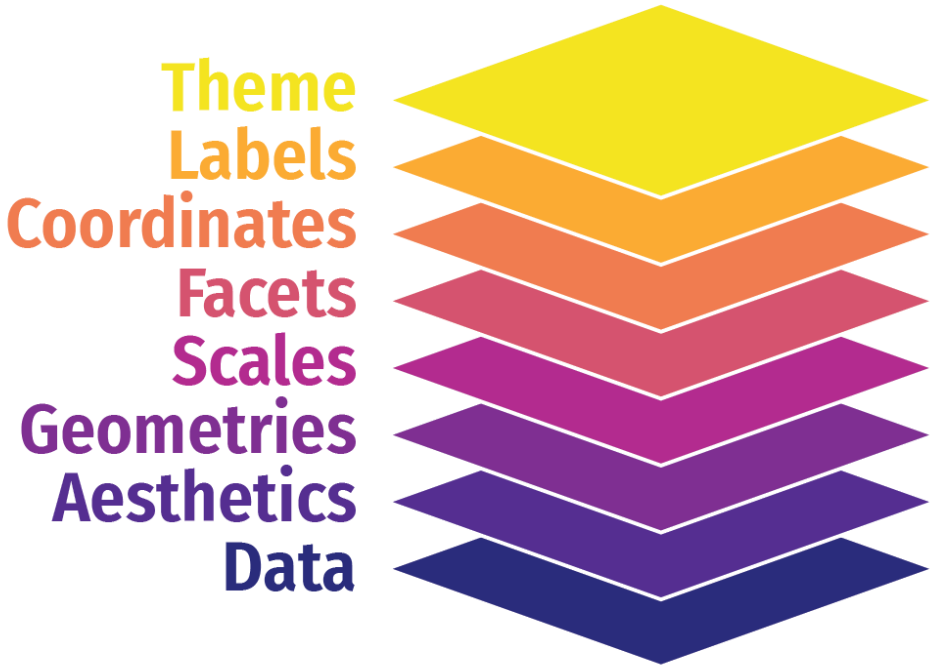


A true grammar

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()`)

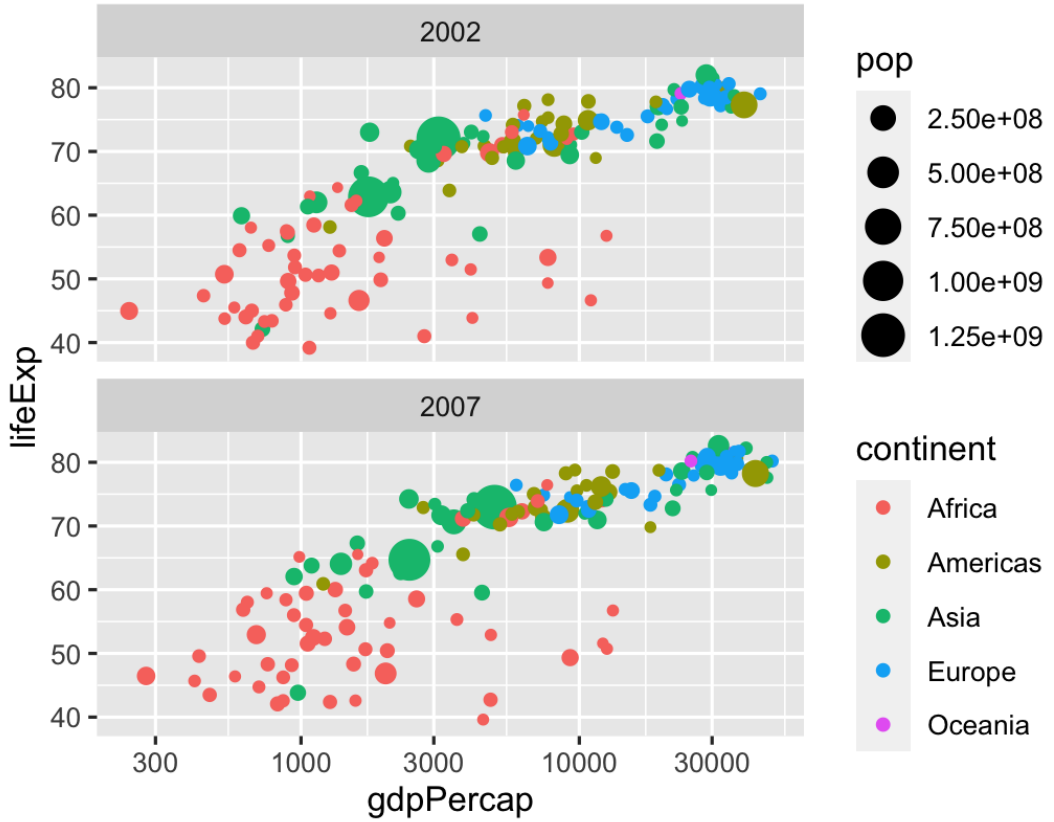
This grammar is portable to other software like Tableau, which is built on the grammar of graphics



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 x-axis with a log, and facet by year

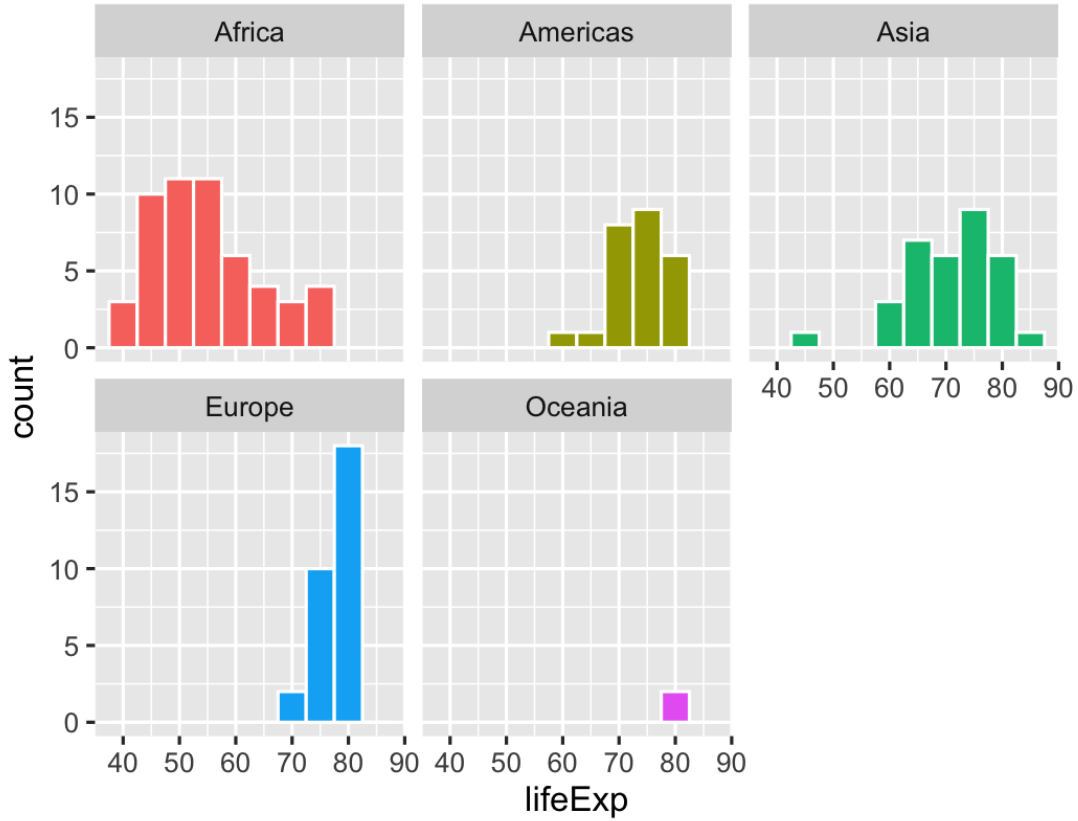
```
ggplot(data = filter(gapminder, year %in% c(2002, 2
       mapping = aes(x = gdpPercap,
                     y = lifeExp,
                     color = continent,
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  facet_wrap(vars(year), ncol = 1)
```



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

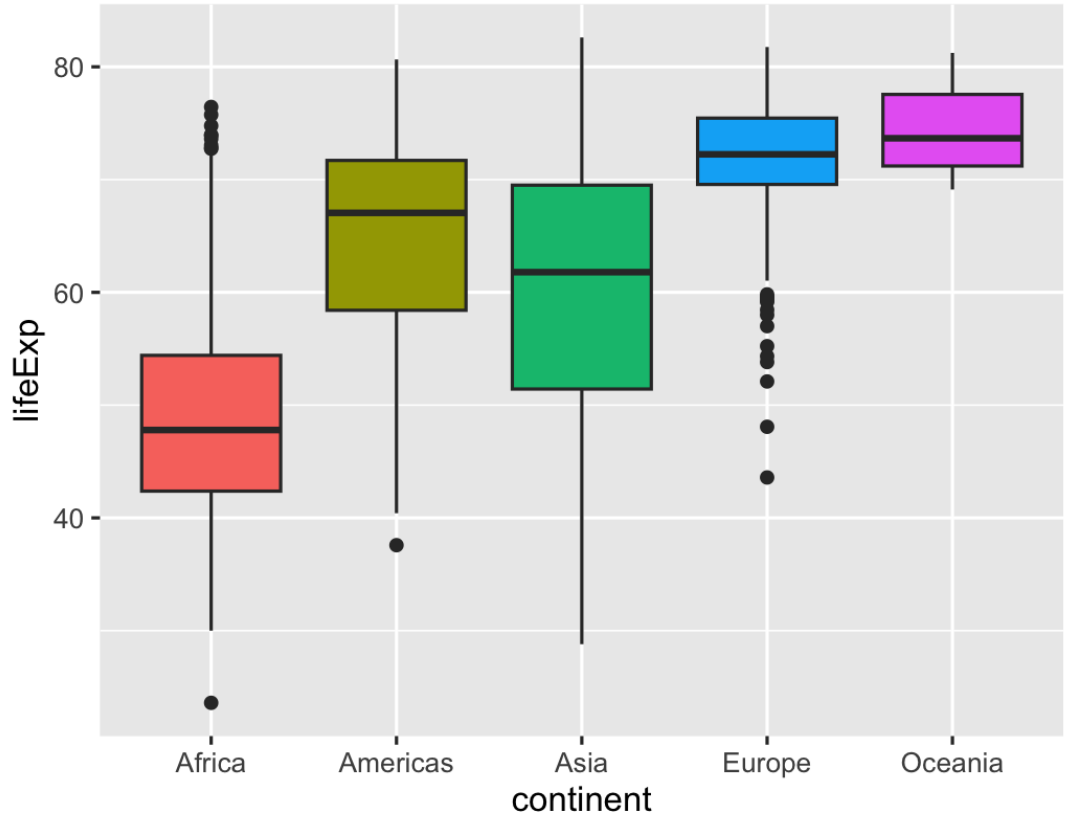
```
ggplot(data = gapminder_2007,
       mapping = aes(x = lifeExp,
                     fill = continent)) +
  geom_histogram(binwidth = 5,
                 color = "white") +
  guides(fill = "none") +  # Turn off legend
  facet_wrap(vars(continent))
```



Describing graphs with the grammar

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

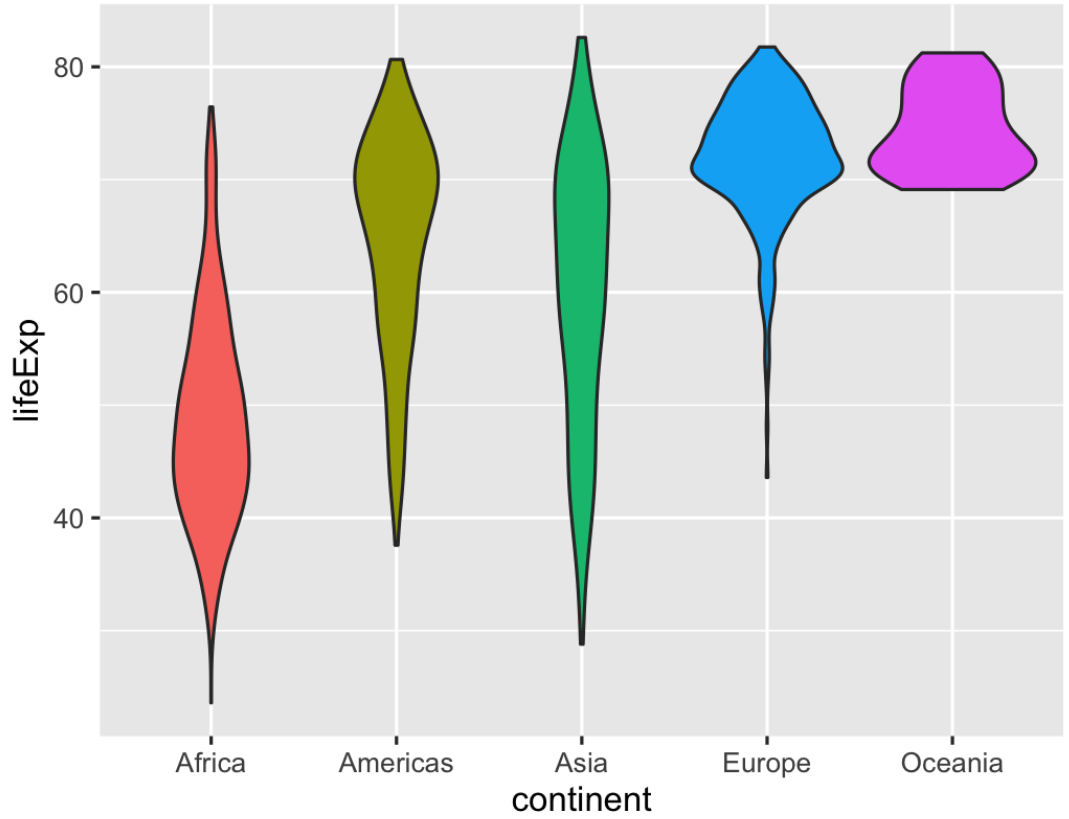
```
ggplot(data = gapminder,
       mapping = aes(x = continent,
                     y = lifeExp,
                     fill = continent)) +
  geom_boxplot() +
  guides(fill = "none")  # Turn off legend
```



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

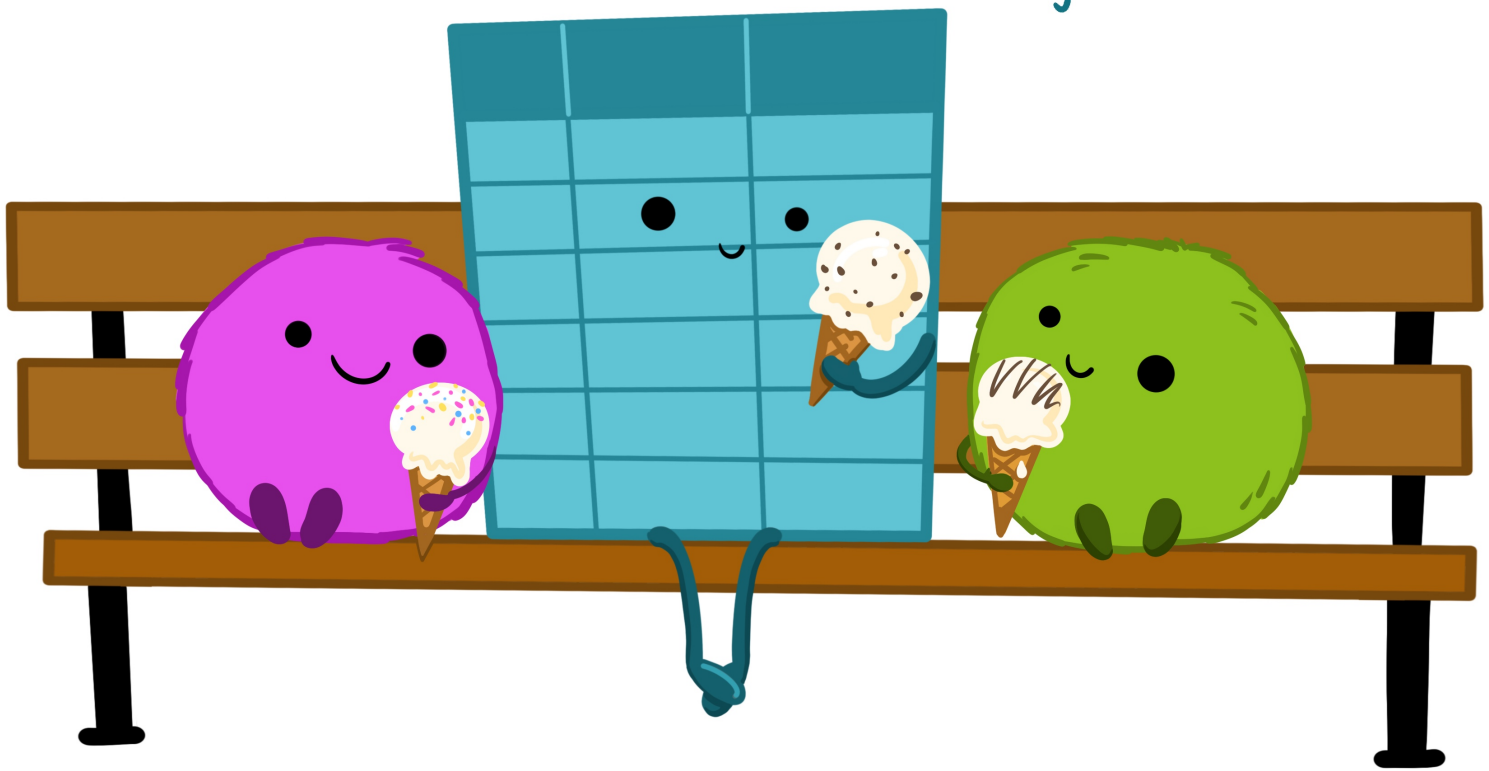
`ggplot()` is incredibly flexible

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

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

We never had to modify `gapminder` to make all those different plots!

make friends with tidy data.



example-07: mapping-data-to-graphics-practice.R

Putting it all together

In the example, we'll build a plot sequentially to see how each grammatical layer changes its appearance

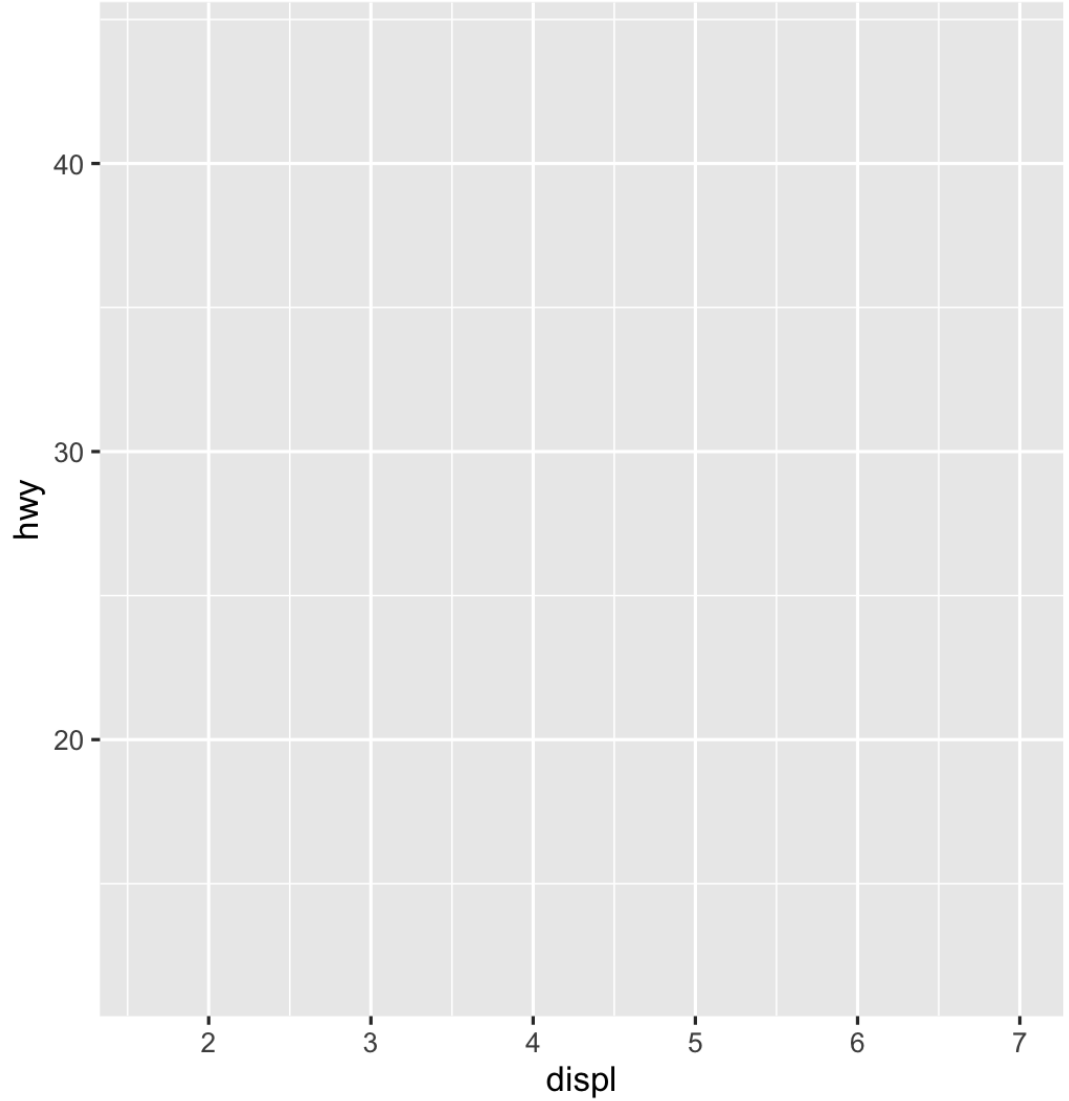
The rest of this section presents that code and output in slide form

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
...	...	...	...	...	...	...

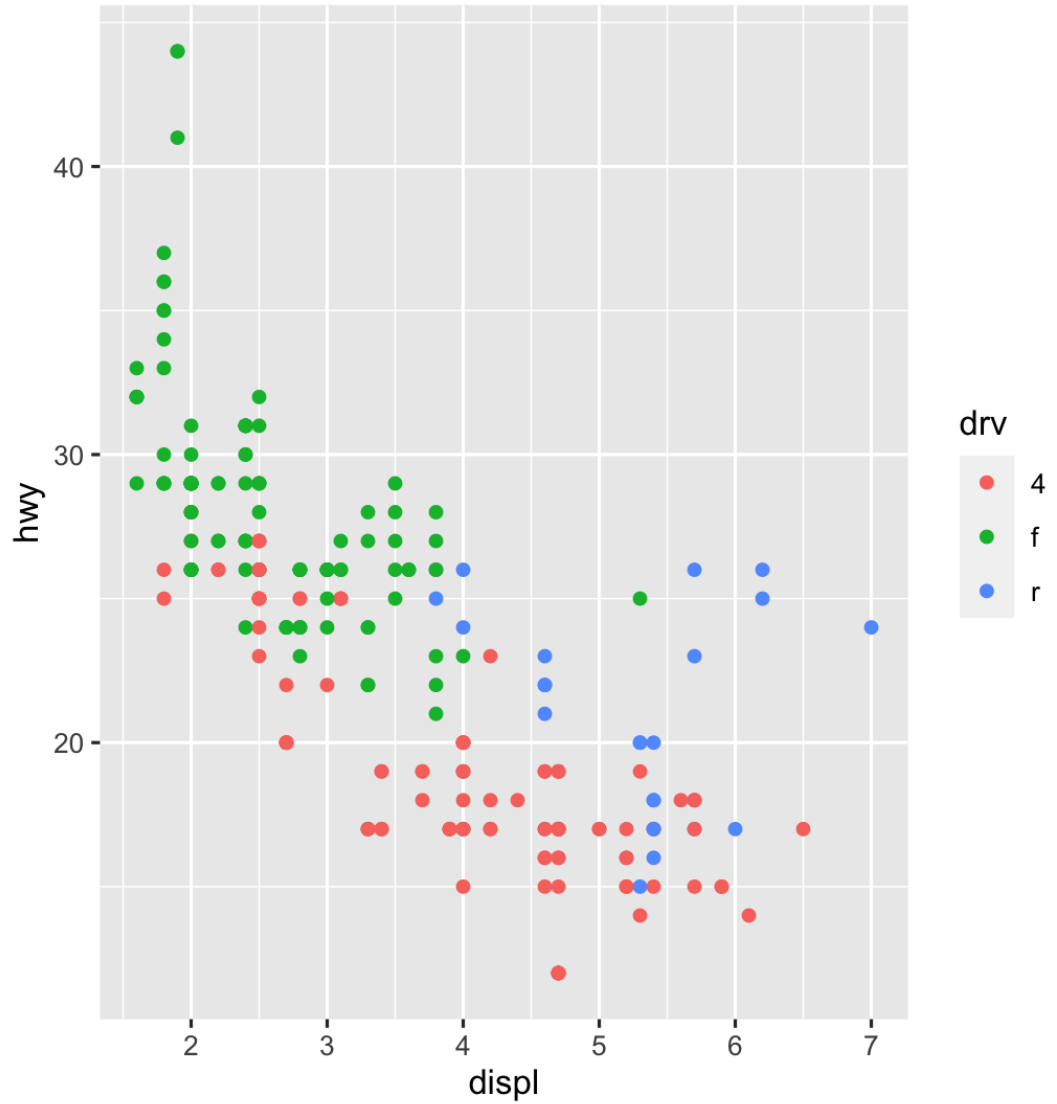
Start with data and aesthetics

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



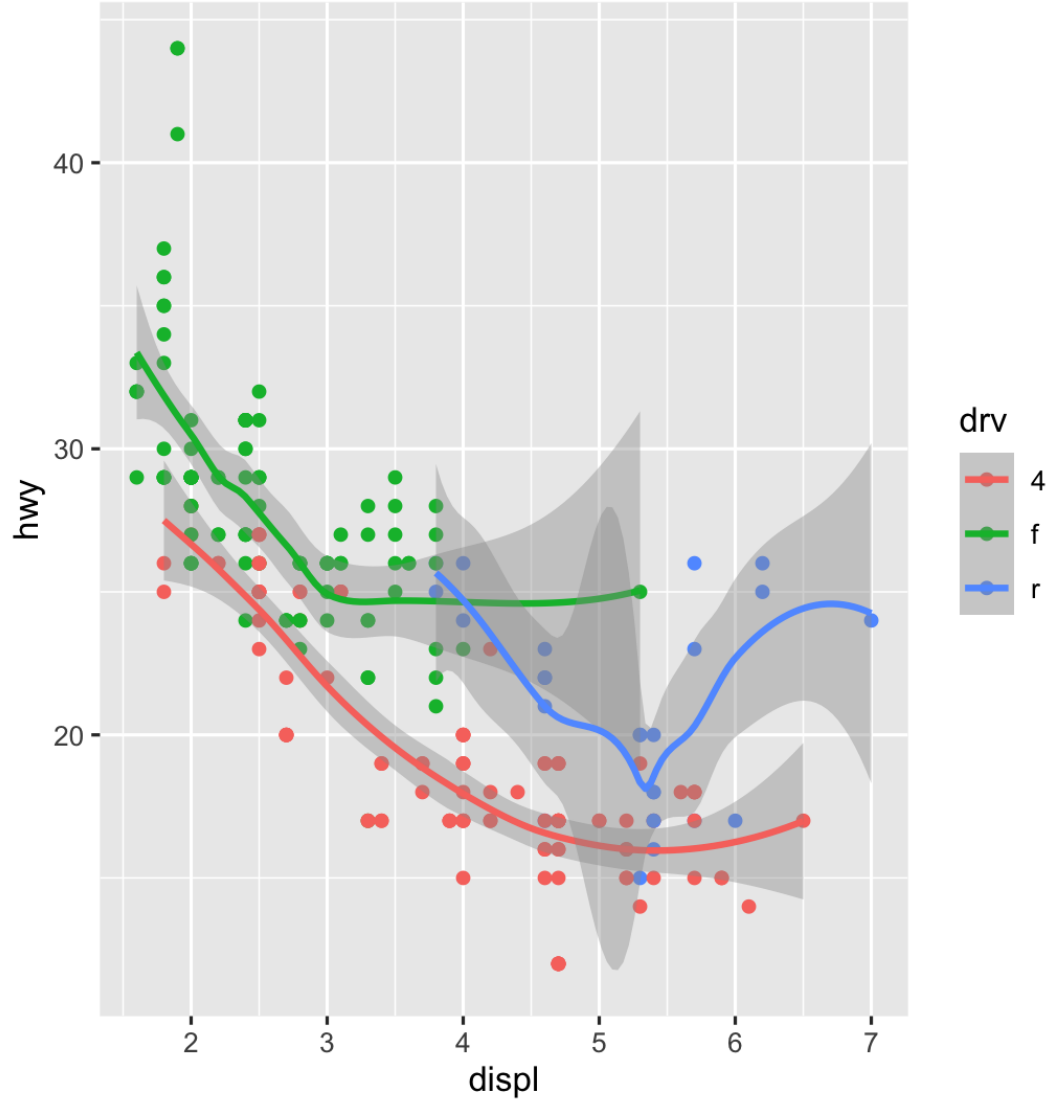
Add a point geom

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



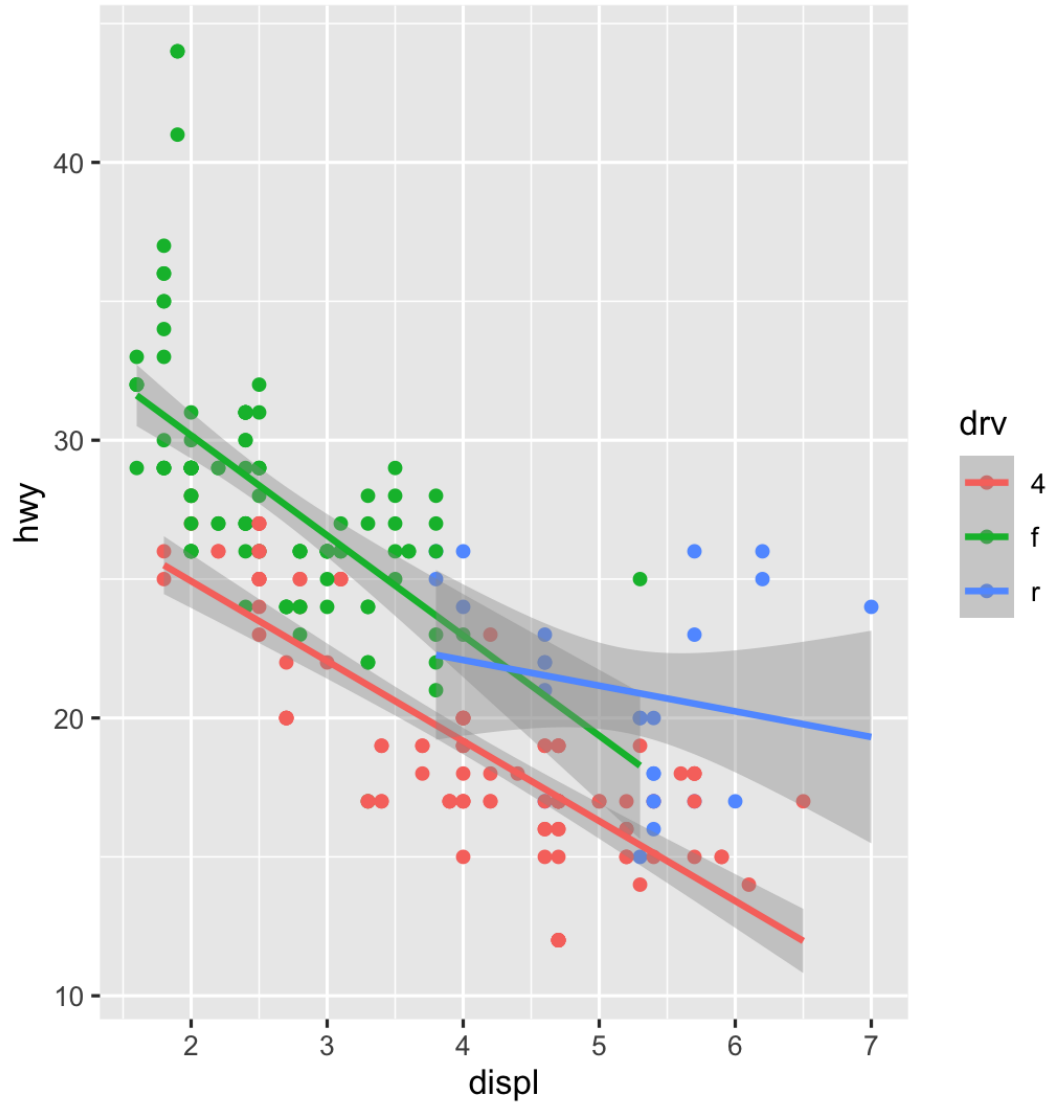
Add a smooth geom

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



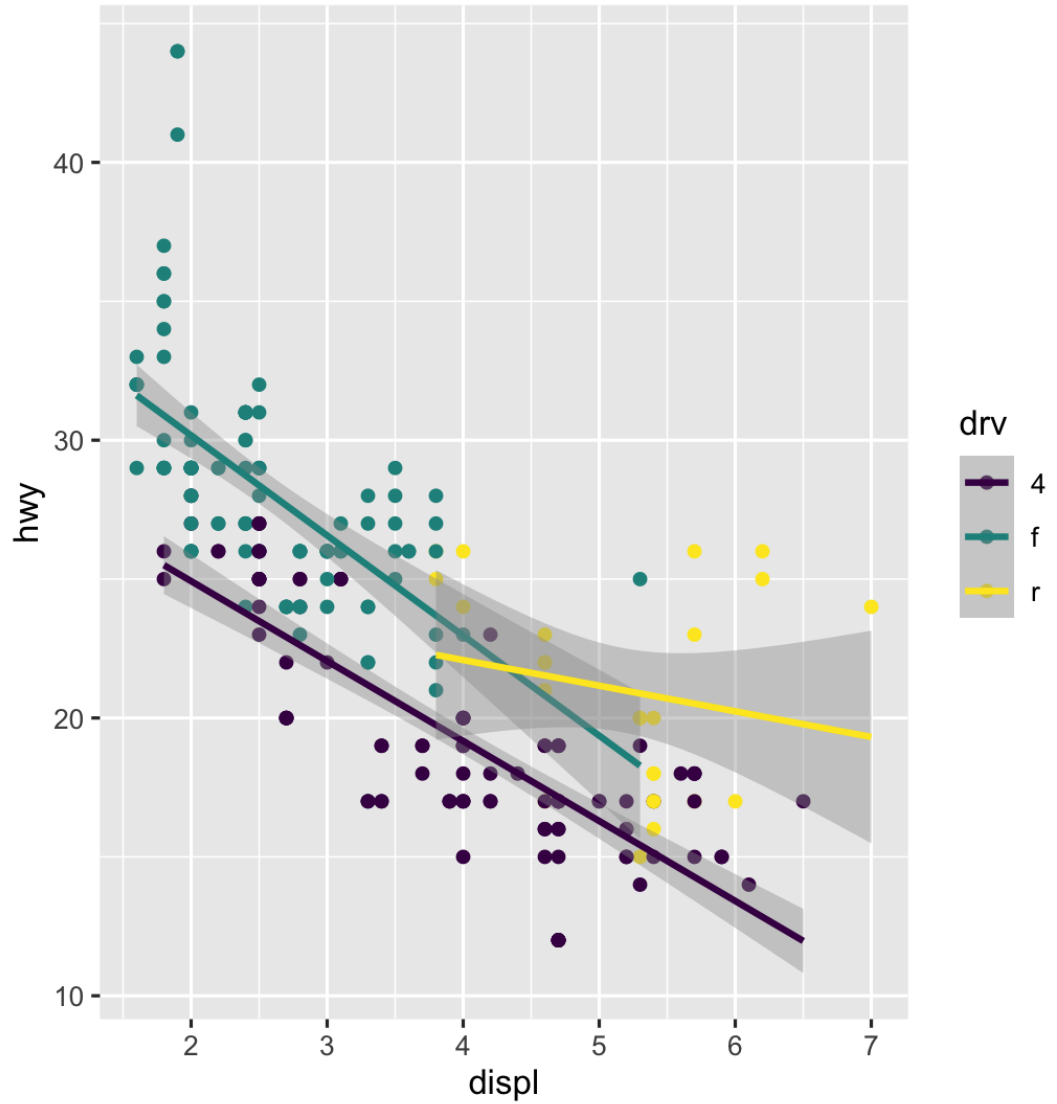
Make it straight

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



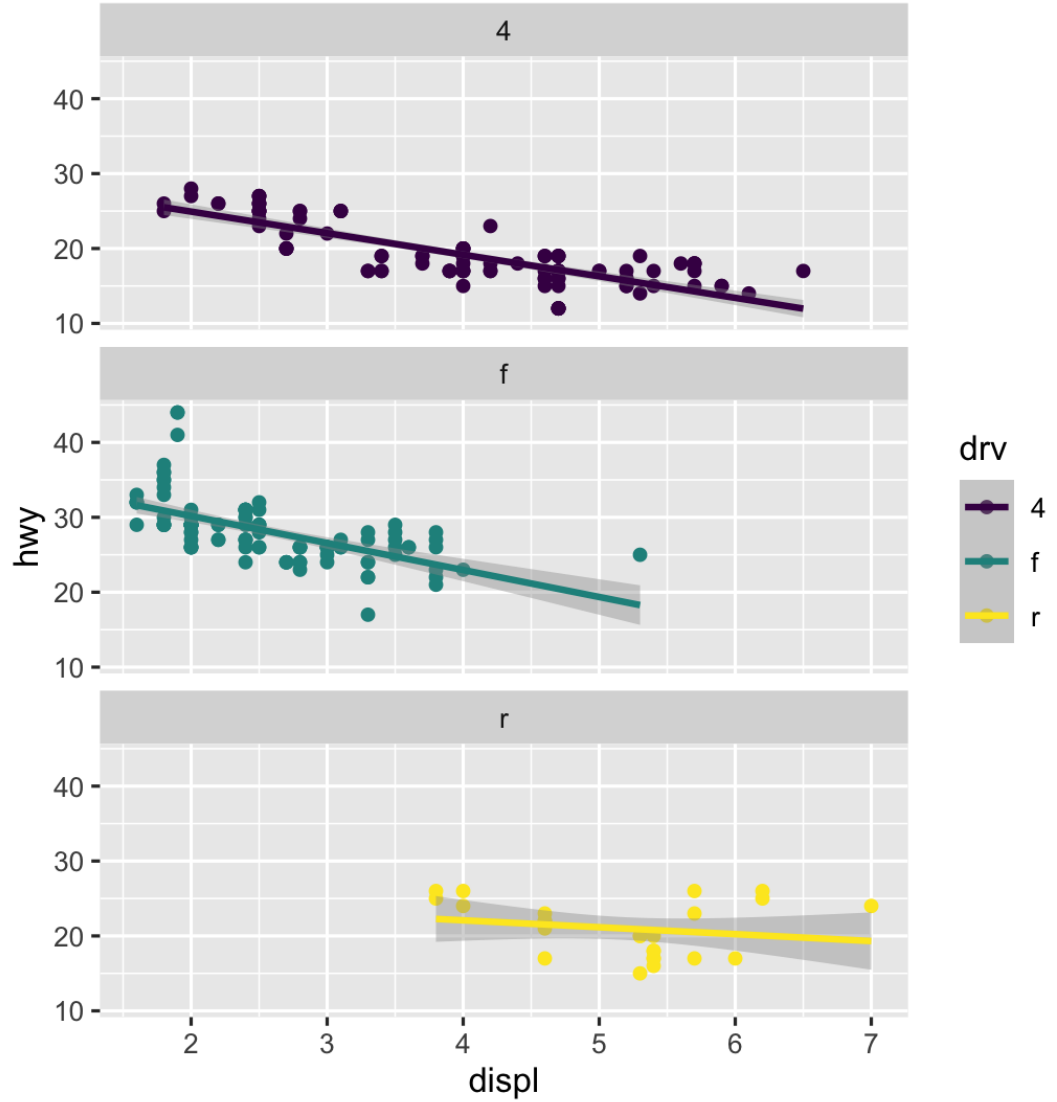
Use a viridis color scale

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



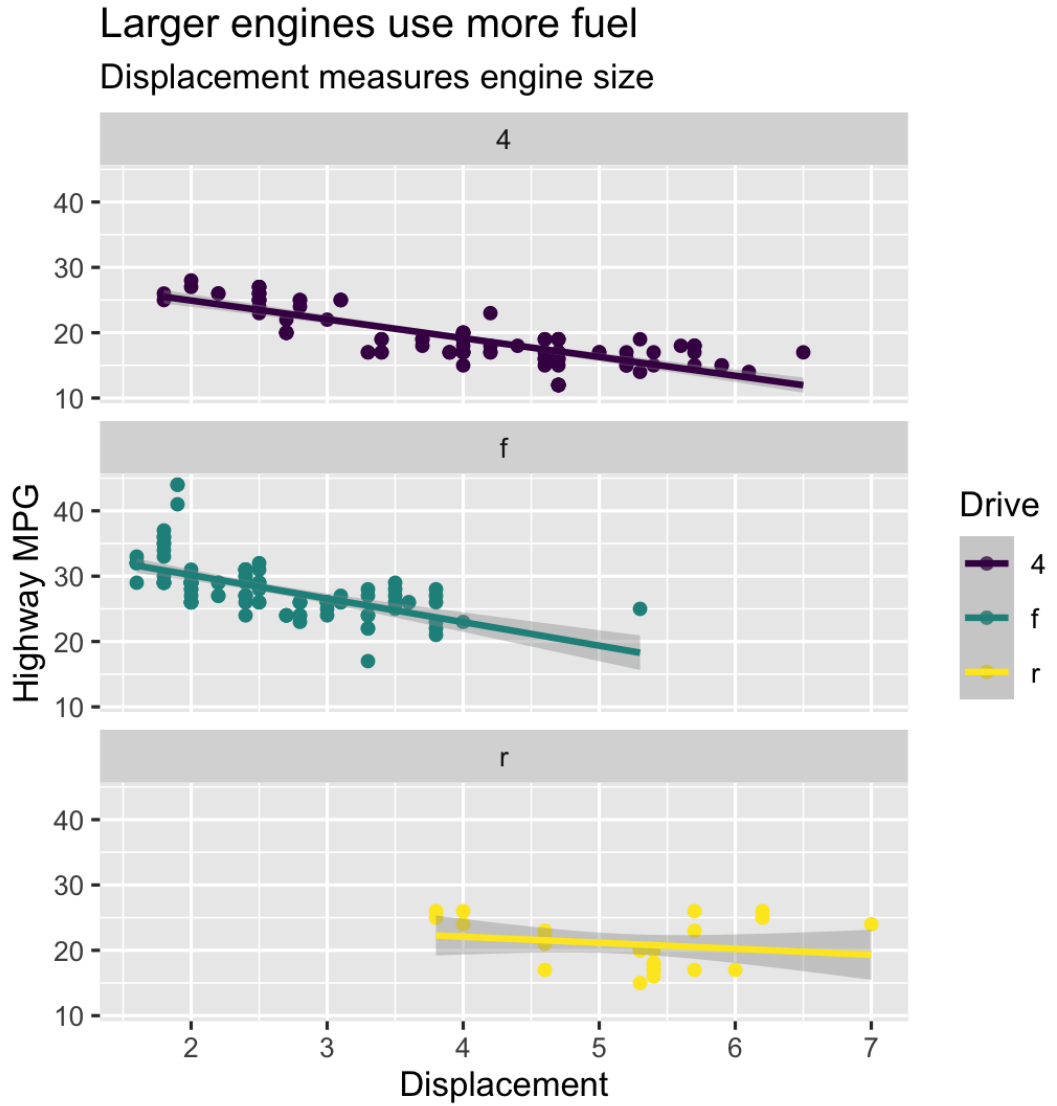
Facet by drive train type

```
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)
```



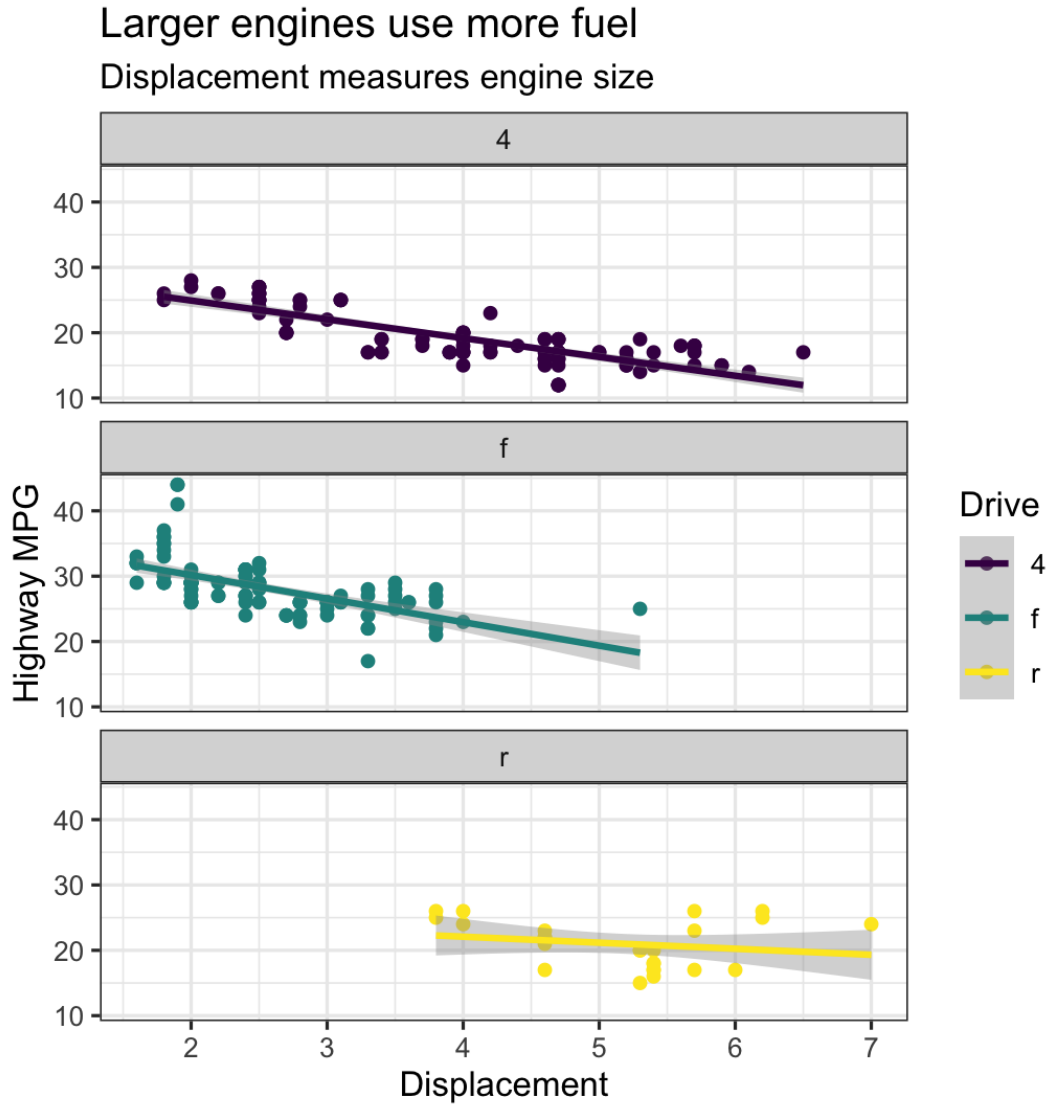
Add labels

```
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 use more fuel",
       subtitle = "Displacement measures engine siz
```



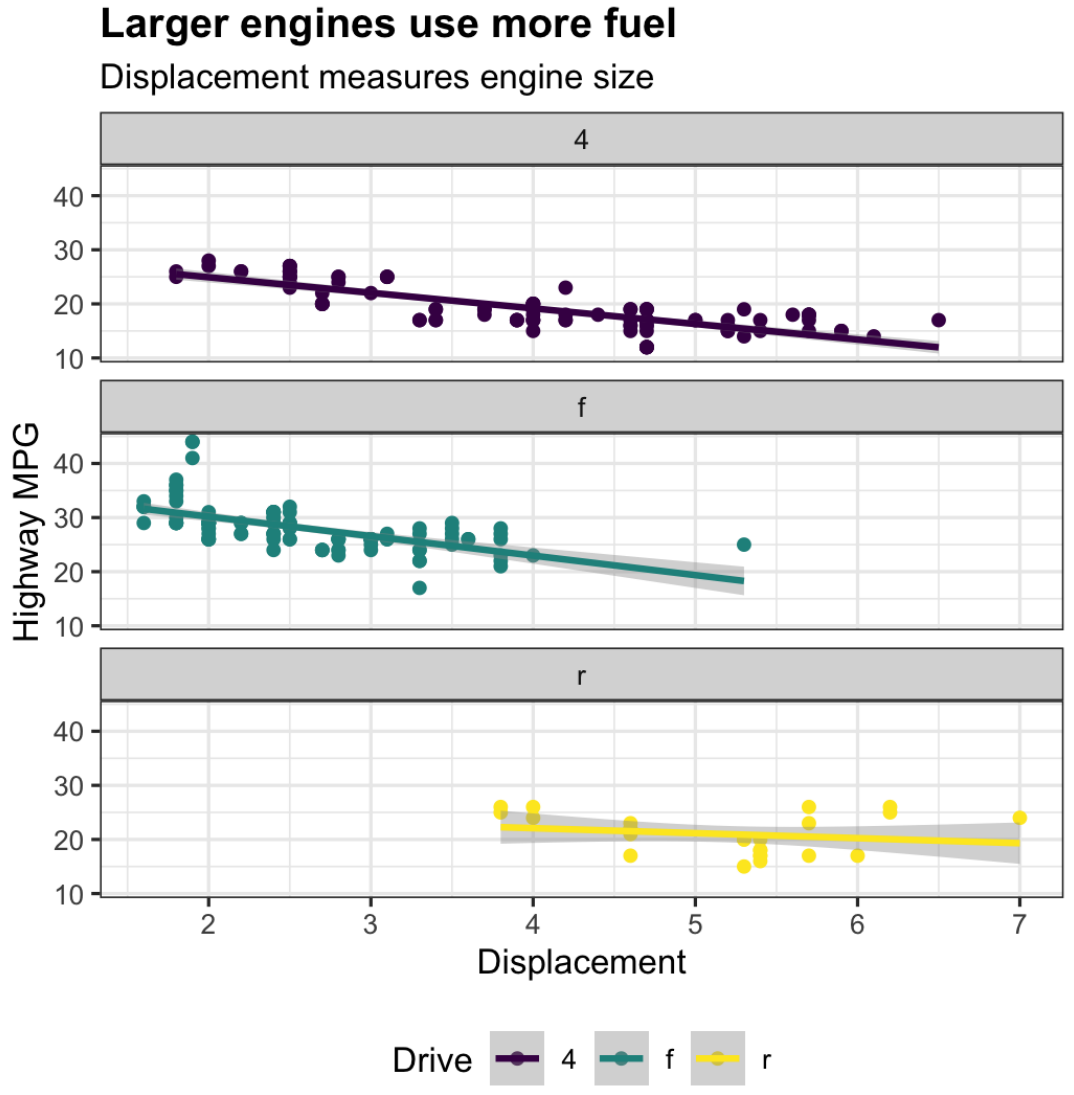
Add a theme

```
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 use more fuel",
       subtitle = "Displacement measures engine siz
  theme_bw()
```



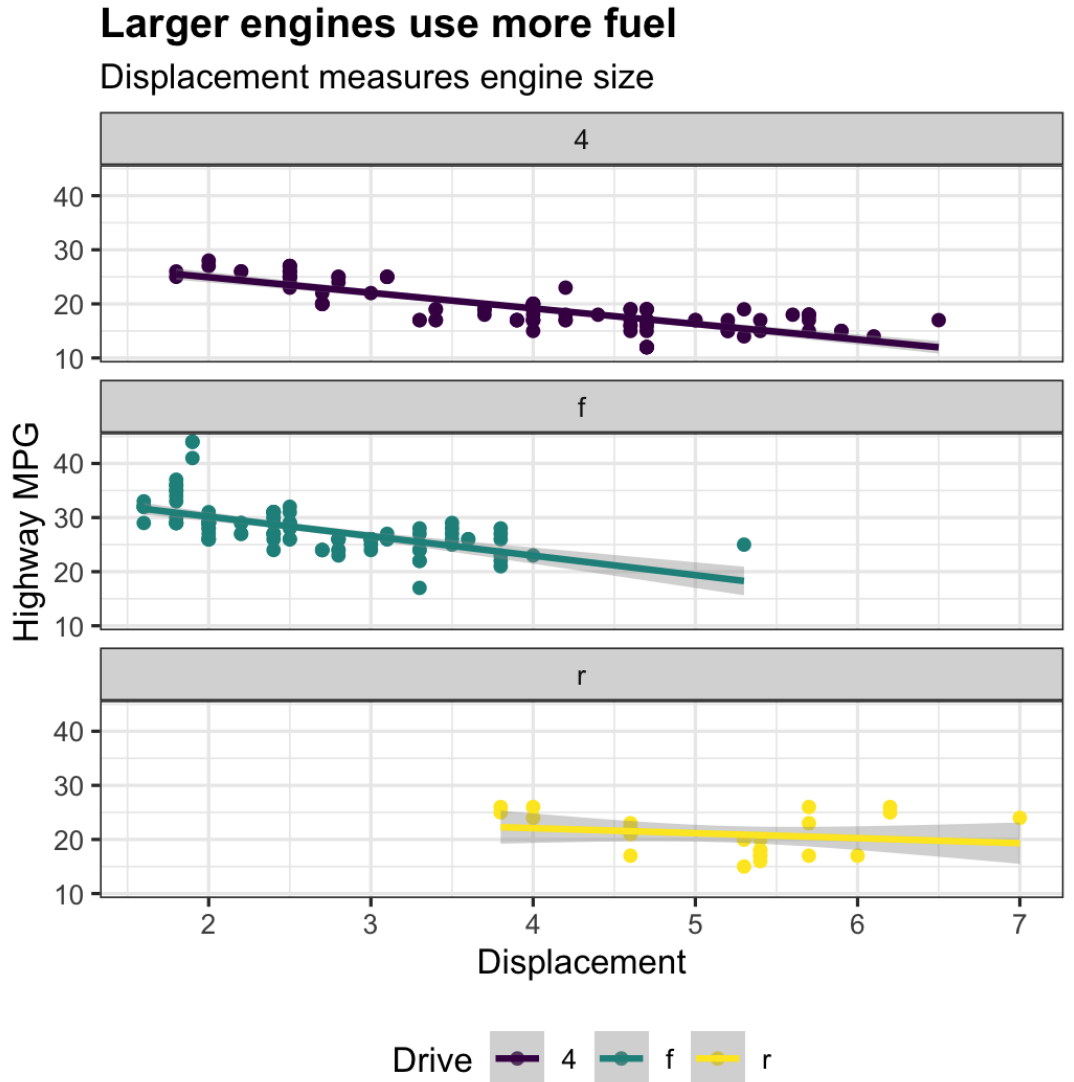
Modify the theme

```
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 use more fuel",
       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 use more fuel",
       subtitle = "Displacement measures engine siz
  theme_bw() +
  theme(legend.position = "bottom",
        plot.title = element_text(face = "bold"))
```

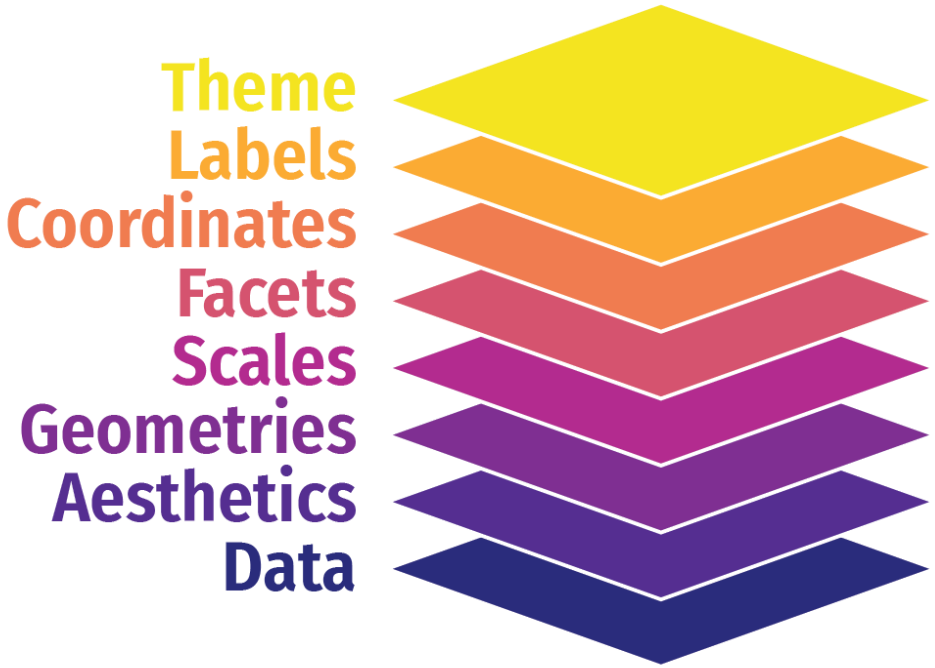


Additional layers

Additional layers

The next several slides contains a preview of additional layers

These are intended as a reference



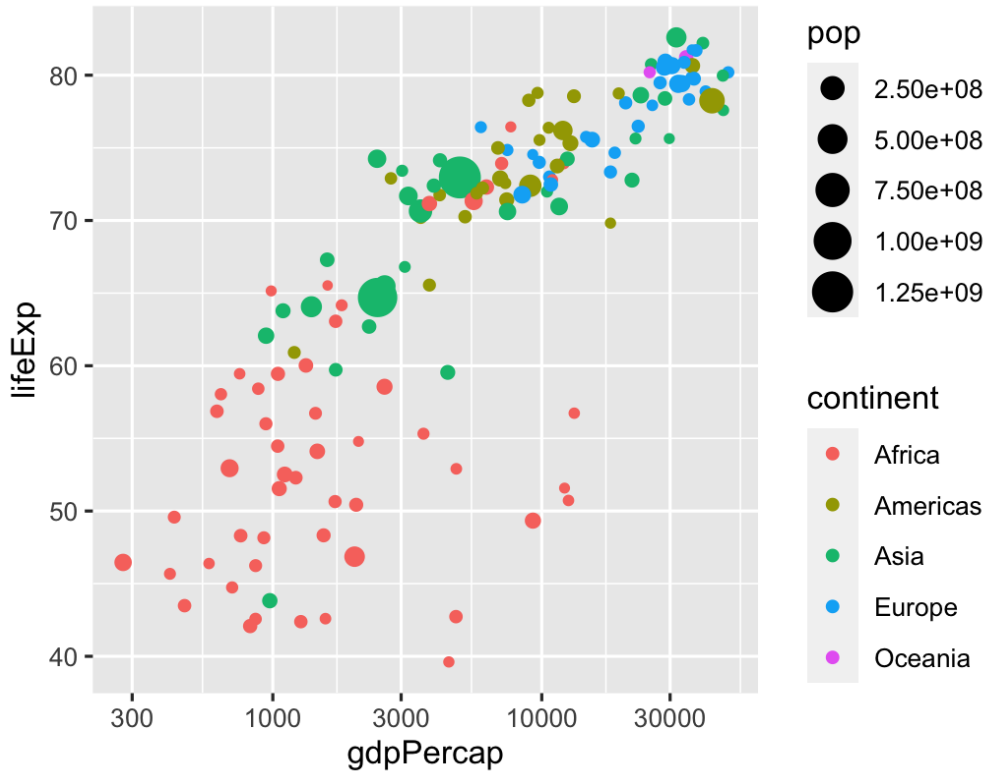
Scales

Scales change the properties of the variable mapping

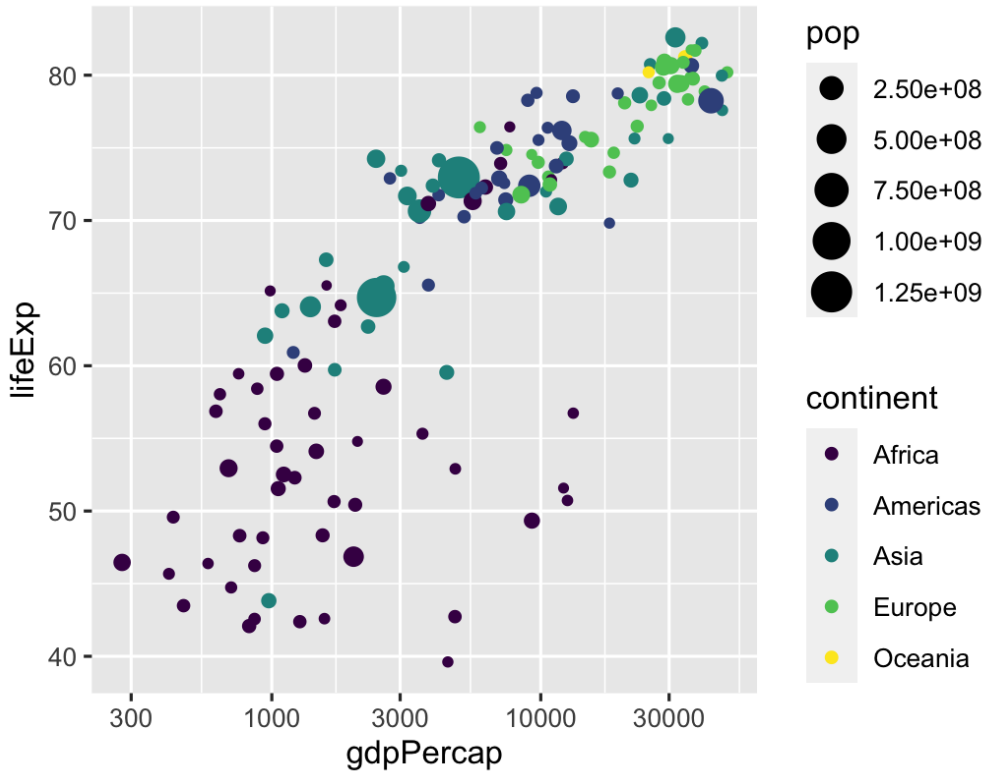
Example layer	What it does
`scale_x_continuous()`	Make the x-axis continuous
`scale_x_continuous(breaks = 1:5)`	Manually specify axis ticks
`scale_x_log10()`	Log the x-axis
`scale_color_gradient()`	Use a gradient
`scale_fill_viridis_d()`	Fill with discrete viridis colors

Scales

`scale_x_log10()`



`scale_color_viridis_d()`



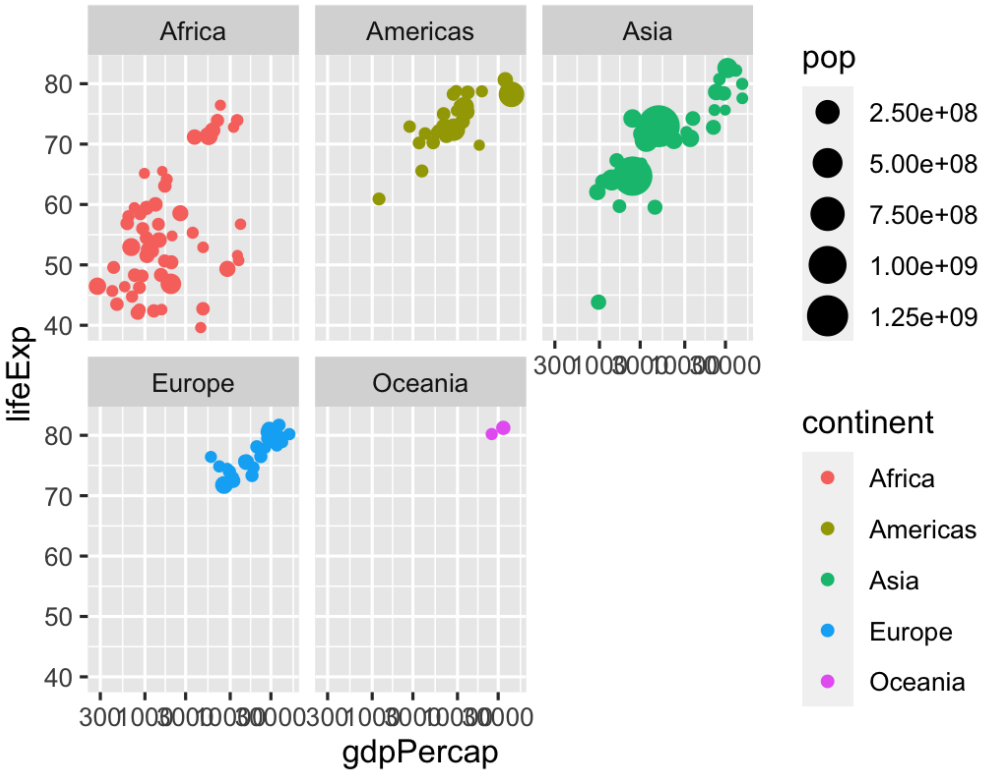
Facets

Facets show subplots for different subsets of data

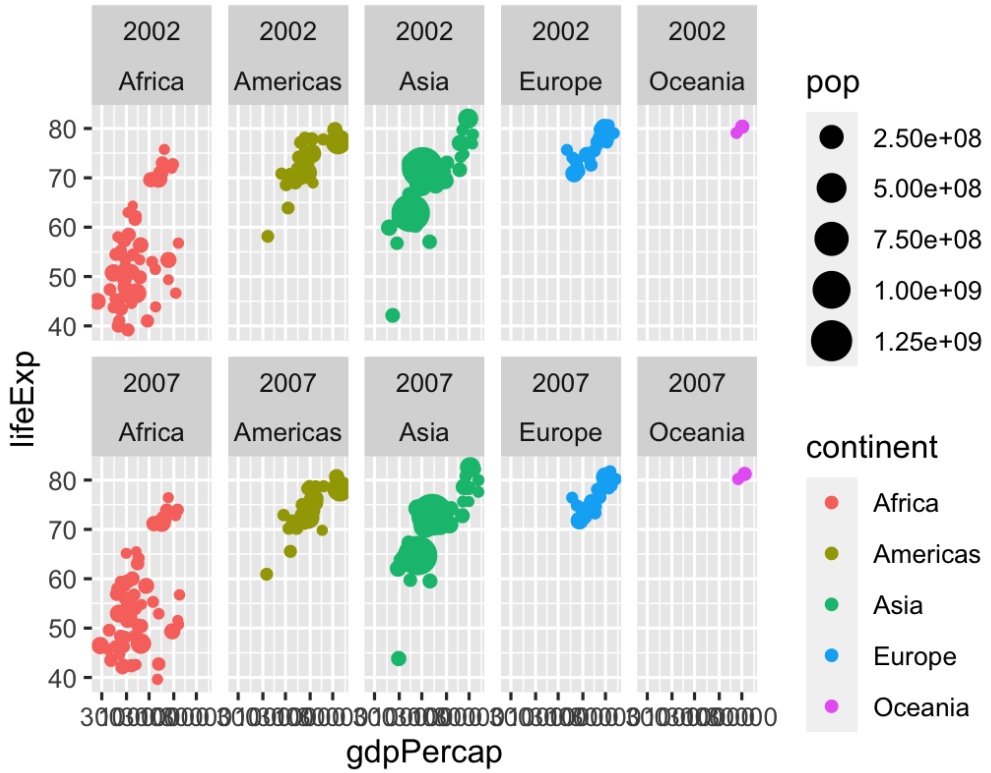
Example layer	What it does
`facet_wrap(vars(continent))`	Plot for each continent
`facet_wrap(vars(continent, year))`	Plot for each continent/year
`facet_wrap(..., ncol = 1)`	Put all facets in one column
`facet_wrap(..., nrow = 1)`	Put all facets in one row

Facets

`facet_wrap(vars(continent))`



`facet_wrap(vars(continent, year))`



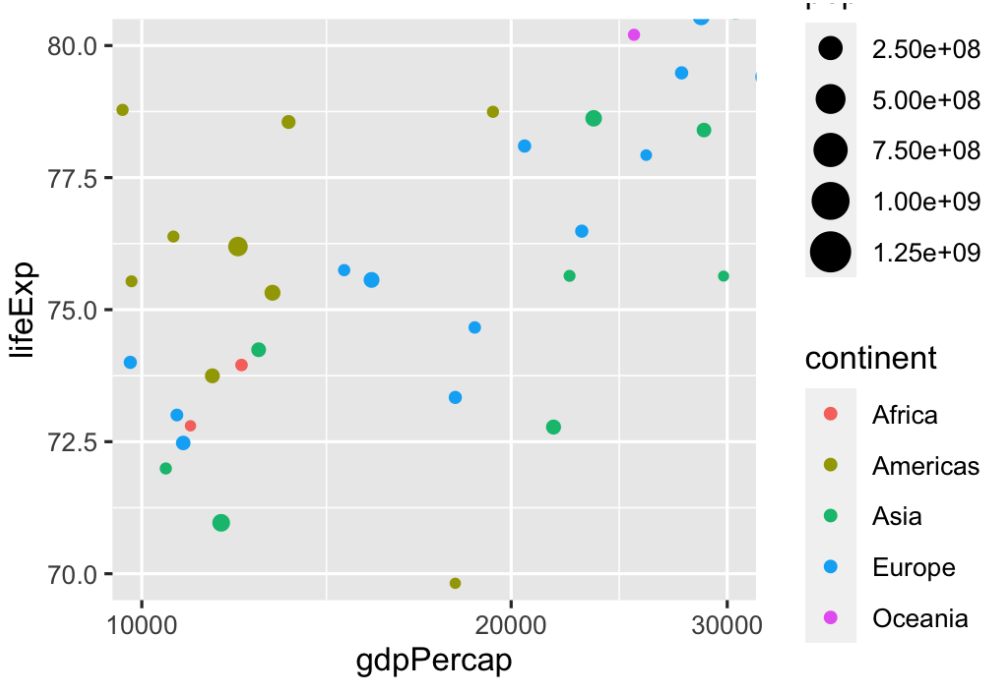
Coordinates

Change the coordinate system

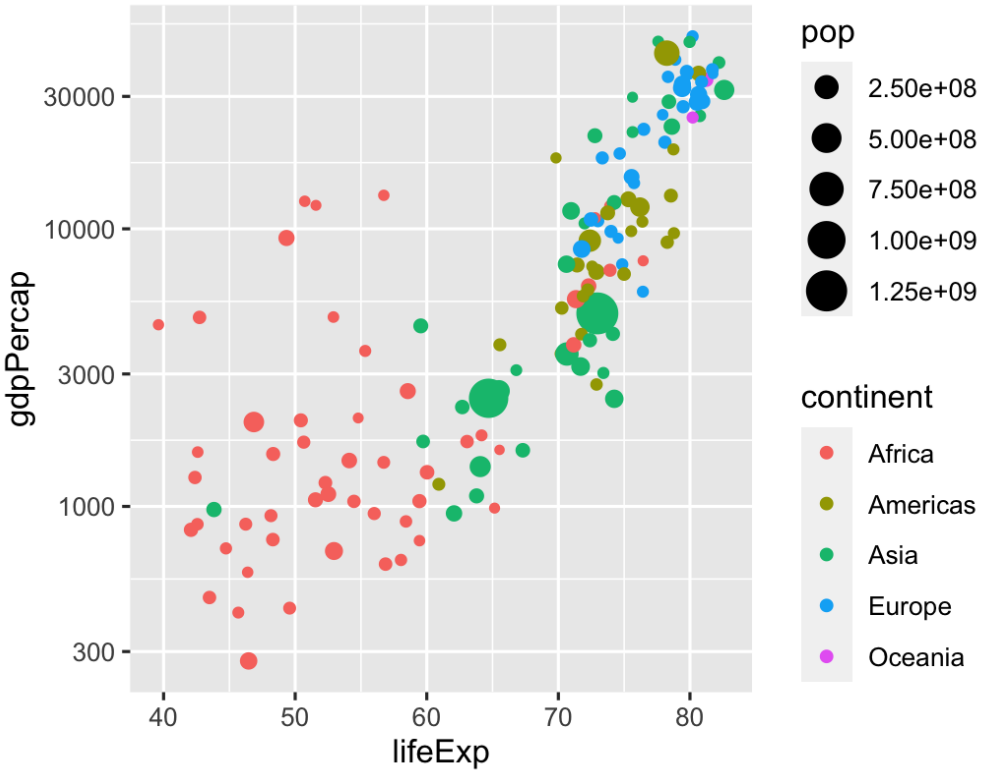
Example layer	What it does
`coord_cartesian(ylim = c(1, 10))`	Zoom in where y is 1–10
`coord_flip()`	Switch x and y
`coord_polar()`	Use polar coordinates

Coordinates

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




```
coord_flip()
```



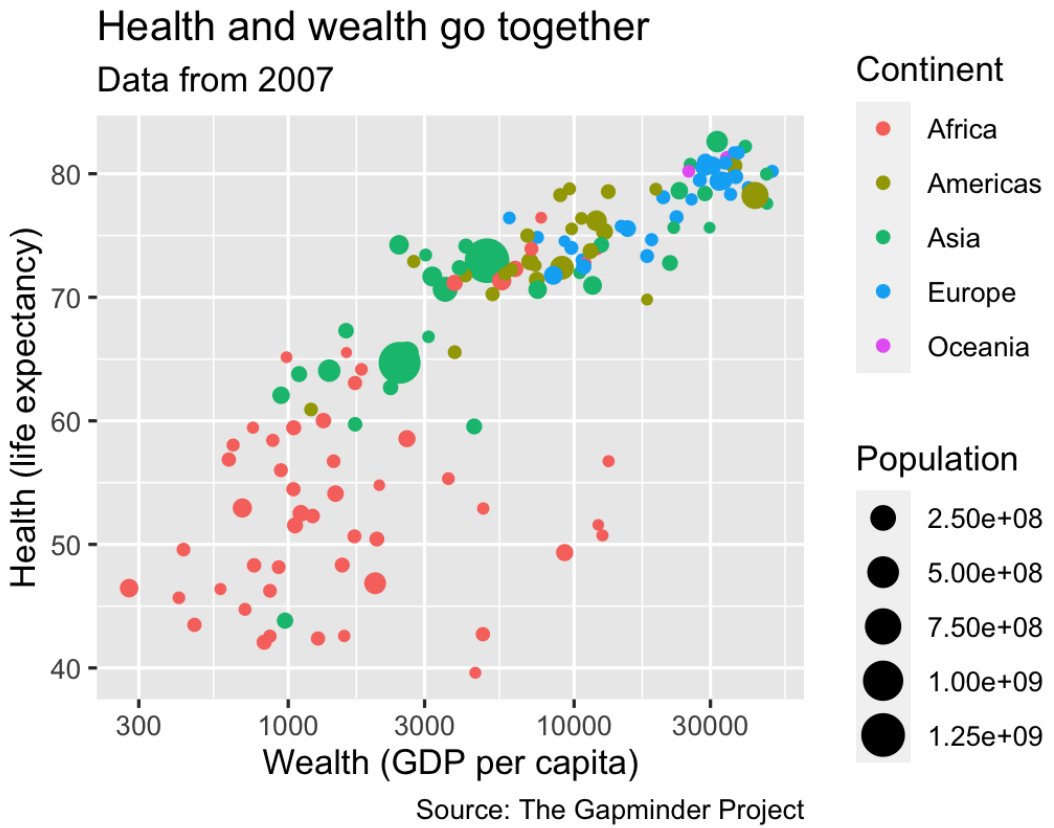
Labels

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

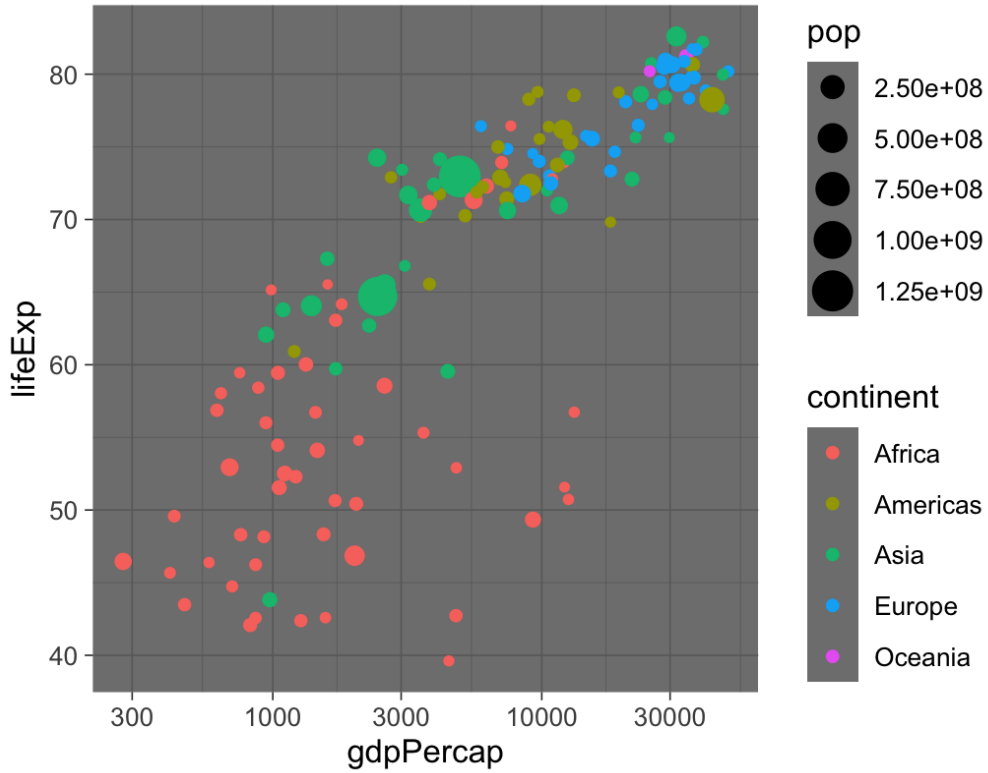
`theme()` can be used to change the appearance of anything in a plot

Lots of themes built in and available from other packages

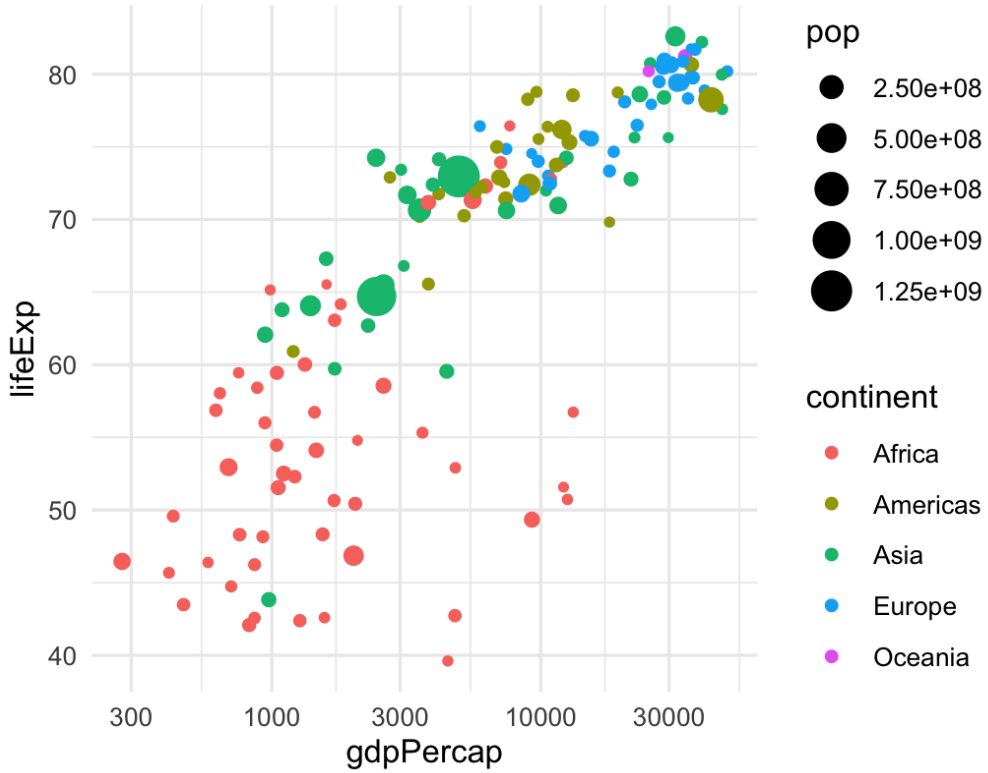
Example layer	What it does
`theme_grey()`	Default grey background
`theme_bw()`	Black and white
`theme_dark()`	Dark
`theme_minimal()`	Minimal

Theme

`theme_dark()`

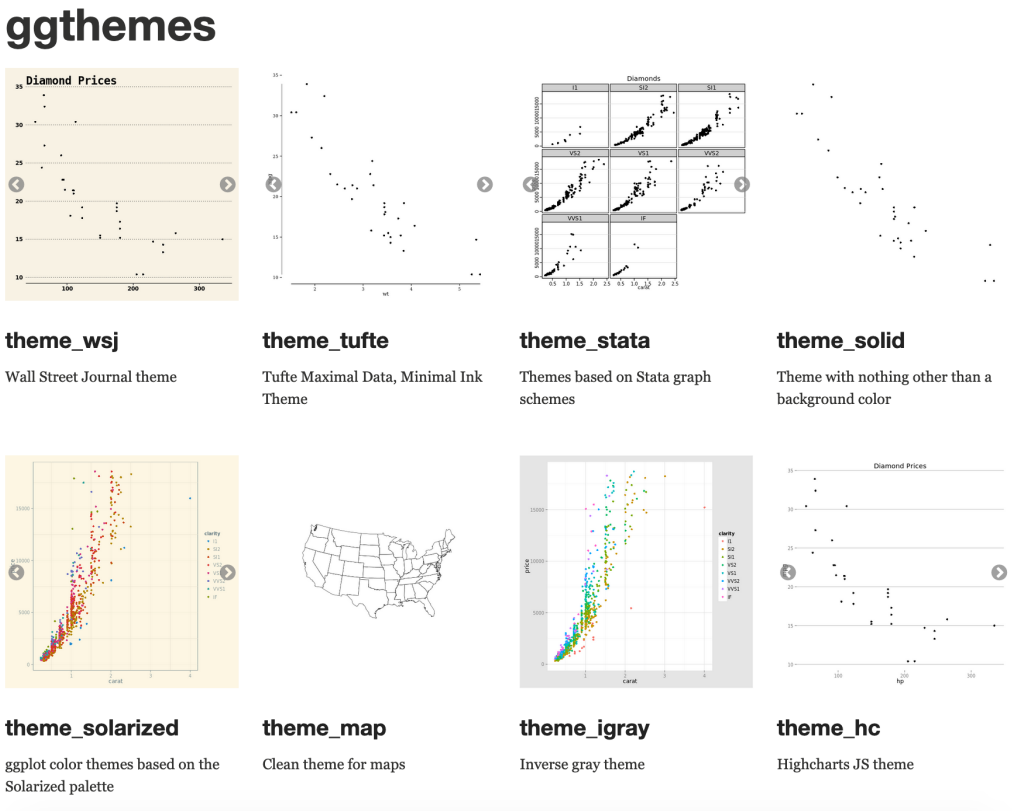


`theme_minimal()`



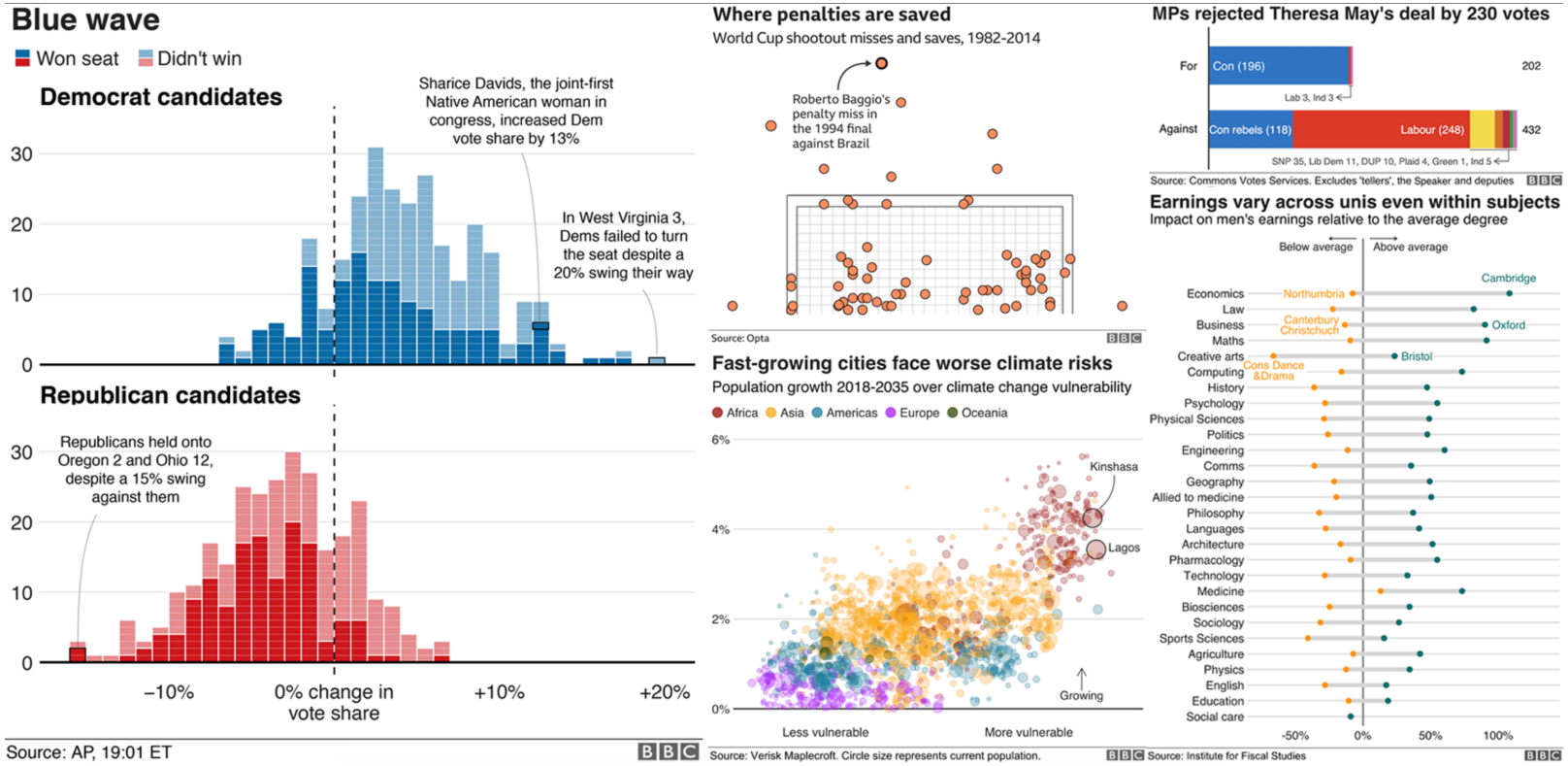
Theme

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



Theme

Organizations often make their own custom themes, like the BBC

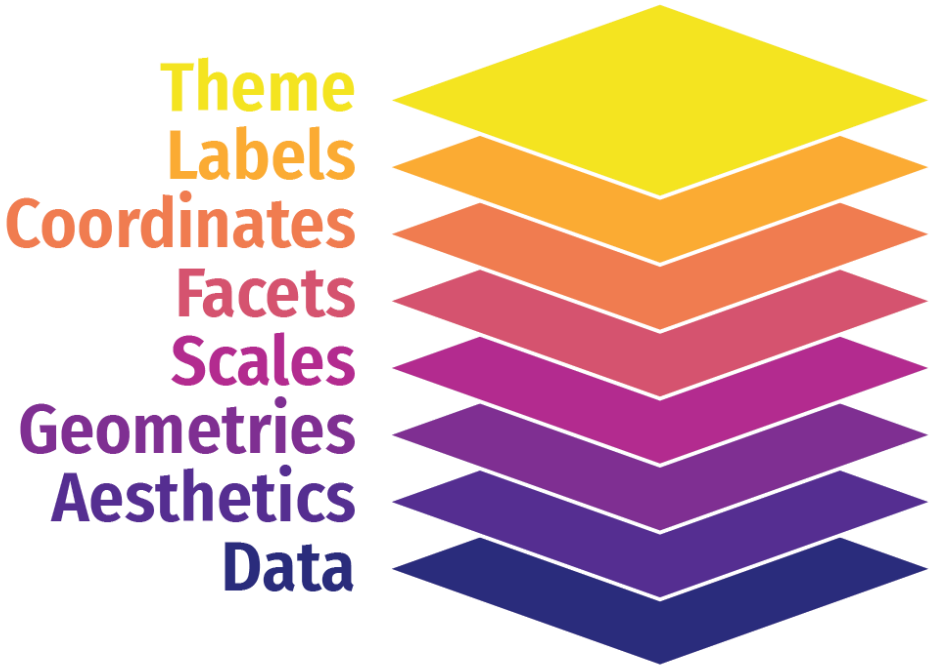


Theme

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