

# Mapping data to graphics and amounts

**Week 7**

AEM 2850 / 5850 : R for Business Analytics  
Cornell Dyson  
Spring 2024

Acknowledgements: [Andrew Heiss](#), [Claus Wilke](#)

# Announcements

This week marks the start of the data visualization component of this course

Next lab is due Monday

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

Questions before we get started?

# Plan for this week

## Tuesday

- Course progress
- Prologue
- Data, aesthetics,  
& the grammar of graphics
- Grammatical layers
- example-07-1
- Prelim 1
- Reference: Additional layers

## Thursday

- Prologue
- Amounts
- Plotting amounts using ggplot
- example-07-2

# 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**

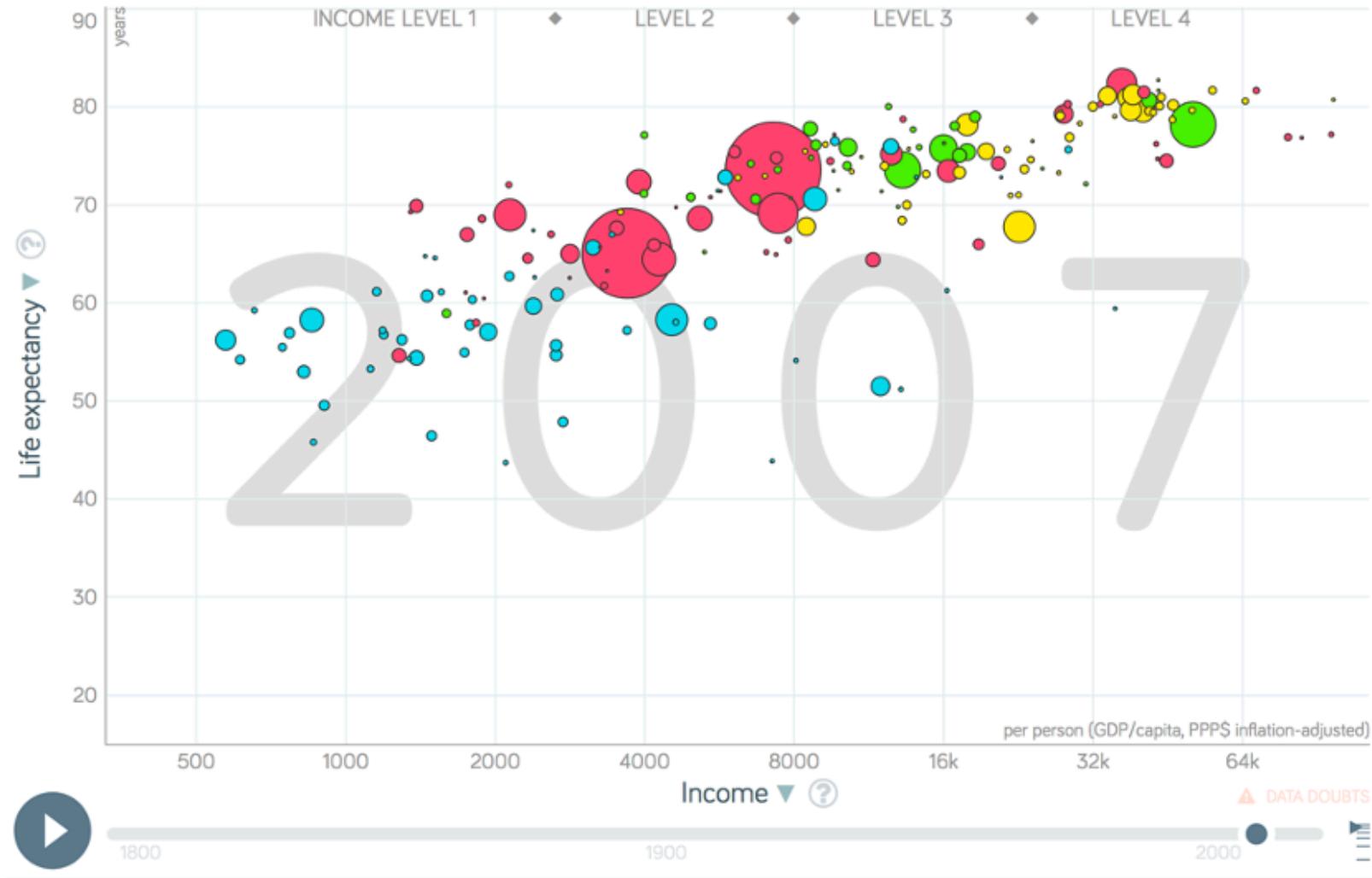
**Weeks 7-10: Data Visualization Foundations**

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

See [aem2850.toddgerarden.com/schedule](http://aem2850.toddgerarden.com/schedule) for details

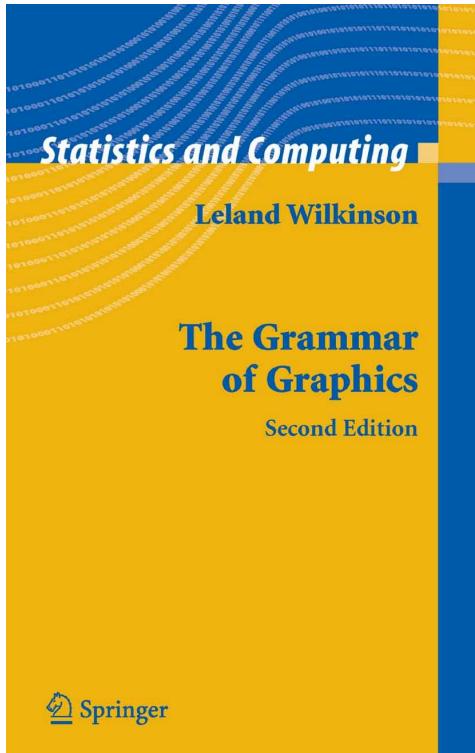
# Prologue

# Health and wealth



Data, aesthetics,  
& the grammar of graphics

# Mapping data to aesthetics



**Data**

A column in a dataset

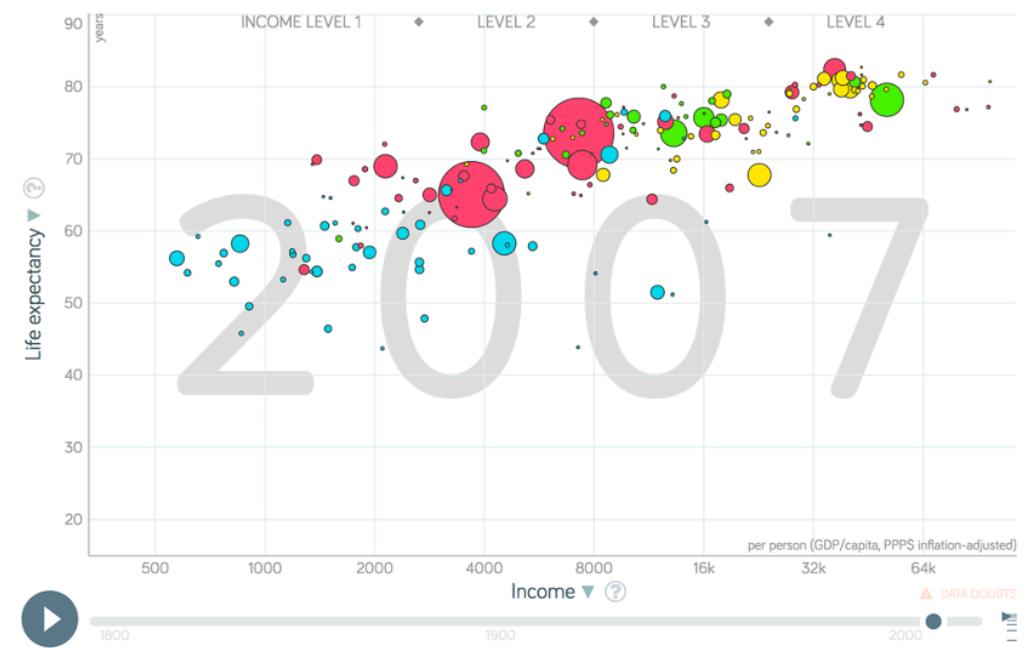
**Aesthetics**

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_classic(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_classic(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_classic(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_classic(base_size = 20) # ignore this line for now
```

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

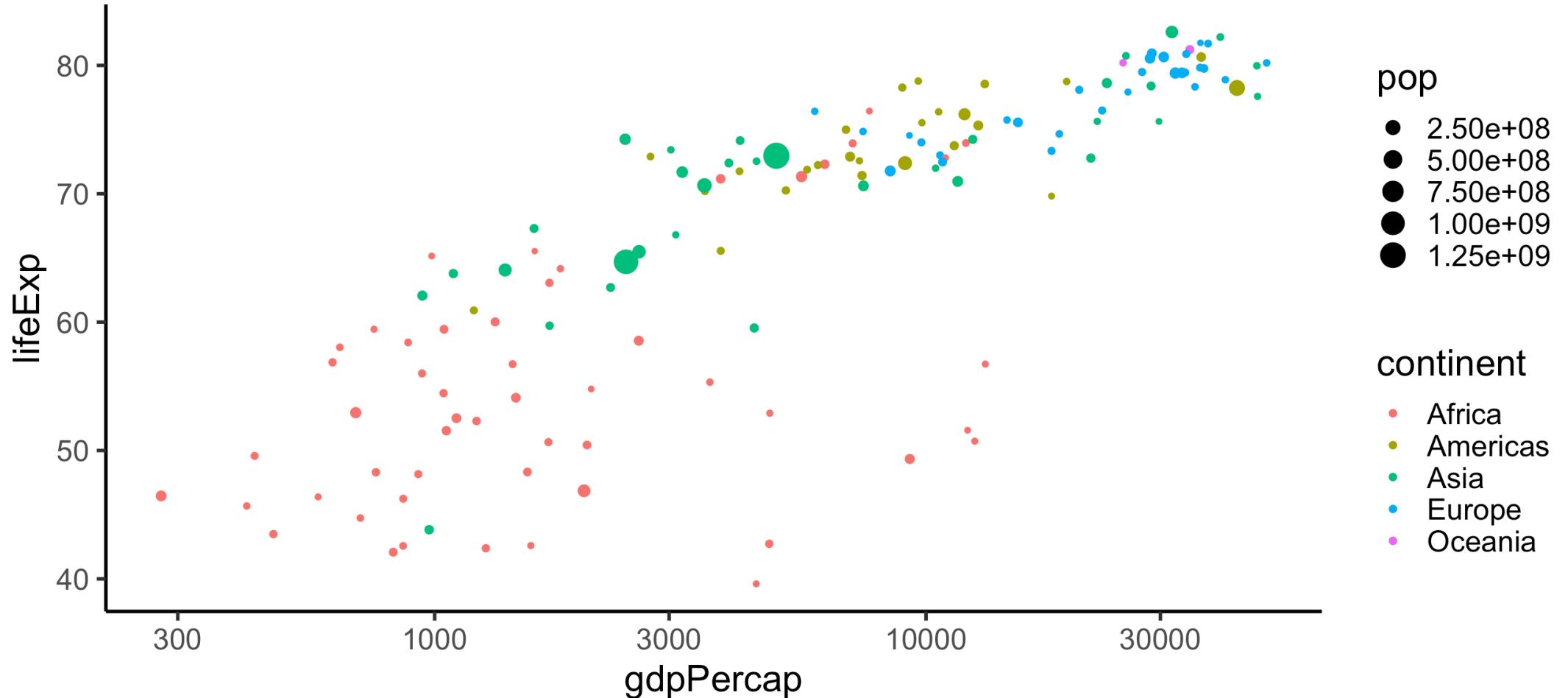
# Mapping from gapminder to aesthetics

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_classic(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

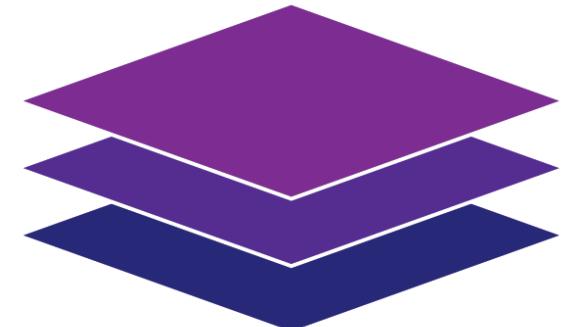
So far we know about data, aesthetics, and geometries

Think of these components as **layers**

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

Why **+** and not **|>**?

Geometries  
Aesthetics  
Data

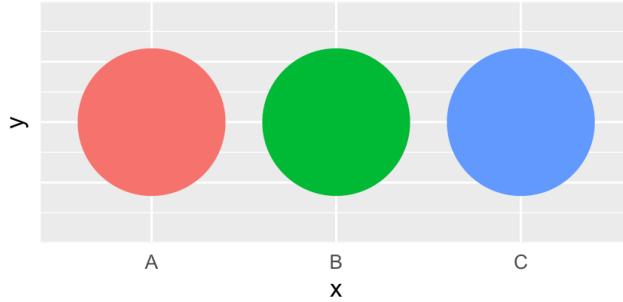


ggplot2 was written before the pipe was discovered

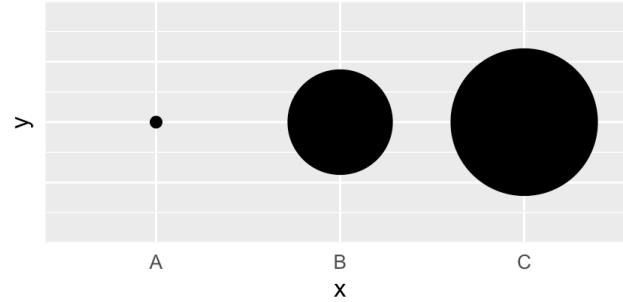
Treat the **+** the same as **|>**

# Possible aesthetics

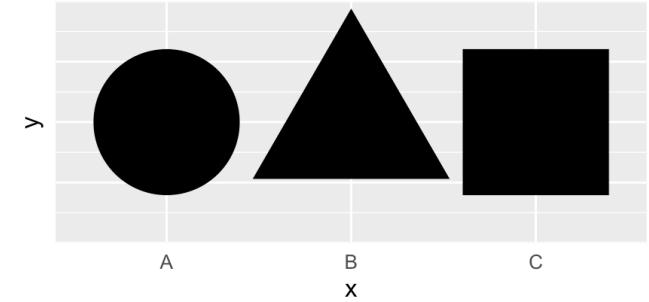
color (discrete)



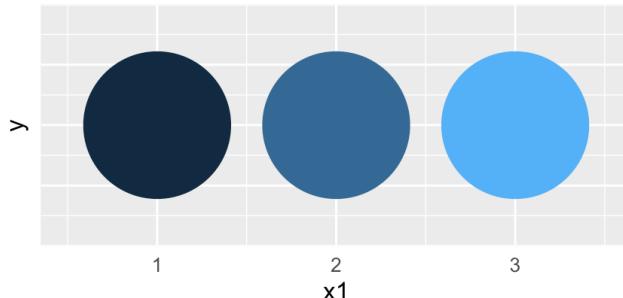
size



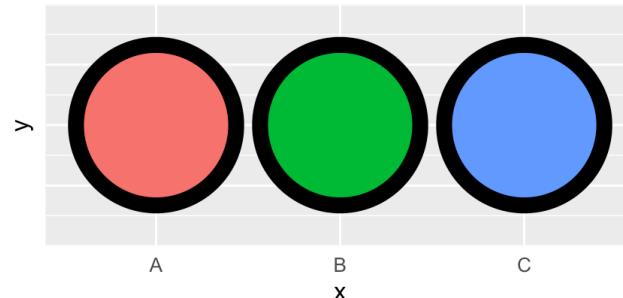
shape



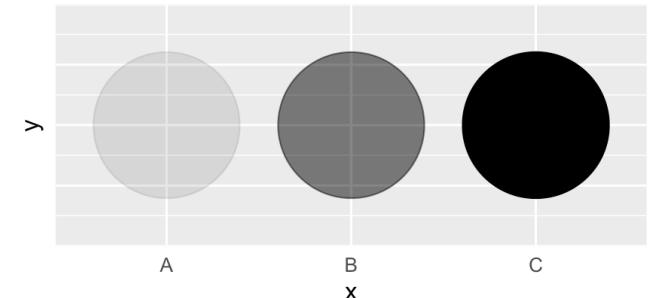
color (continuous)



fill



alpha (opacity)



# Possible geometries

---

## Example geometry What it makes



`geom_col()`

Bar charts

*text*

`geom_text()`

Text



`geom_point()`

Points



`geom_boxplot()`

Boxplots



`geom_sf()`

Maps

# Possible geometries

There are dozens of possible geometries

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

See the [ggplot2 documentation](#) for examples of all the different geometry layers

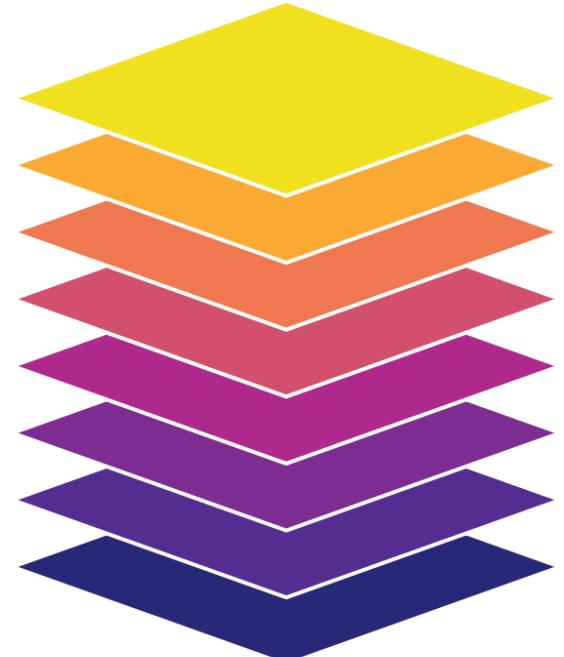
# Additional layers

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

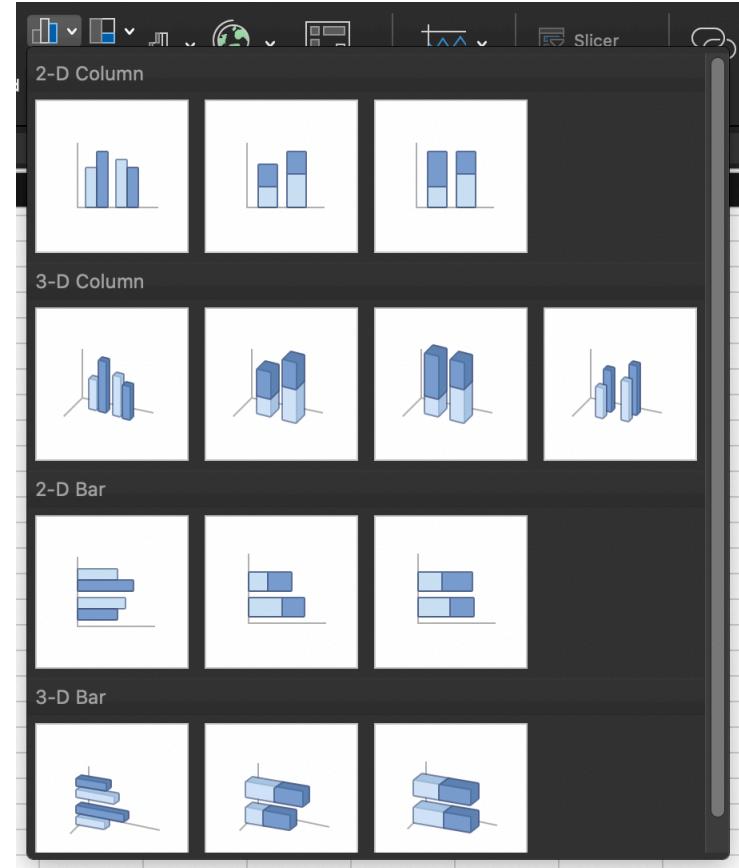
Theme  
Labels  
Coordinates  
Facets  
Scales  
Geometries  
Aesthetics  
Data



# 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**



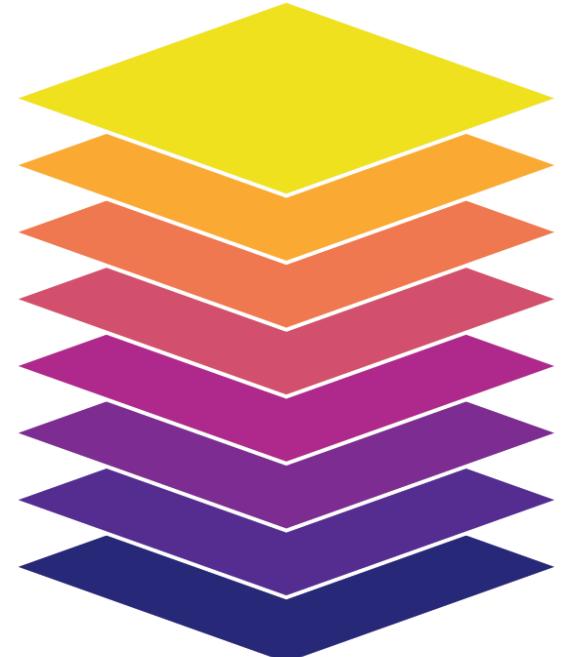
# A true grammar

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
- Geometries can be interchangeable (e.g., `geom_violin()` and `geom_boxplot()`)

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

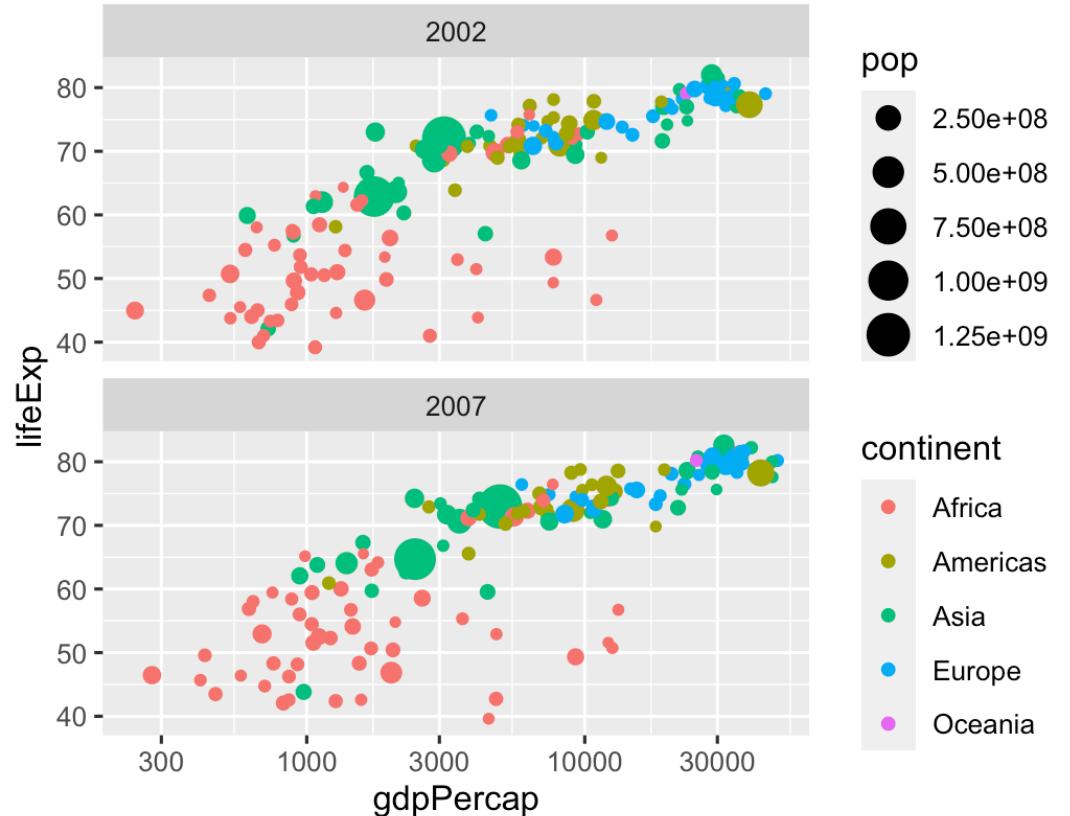
Theme  
Labels  
Coordinates  
Facets  
Scales  
Geometries  
Aesthetics  
Data



# 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

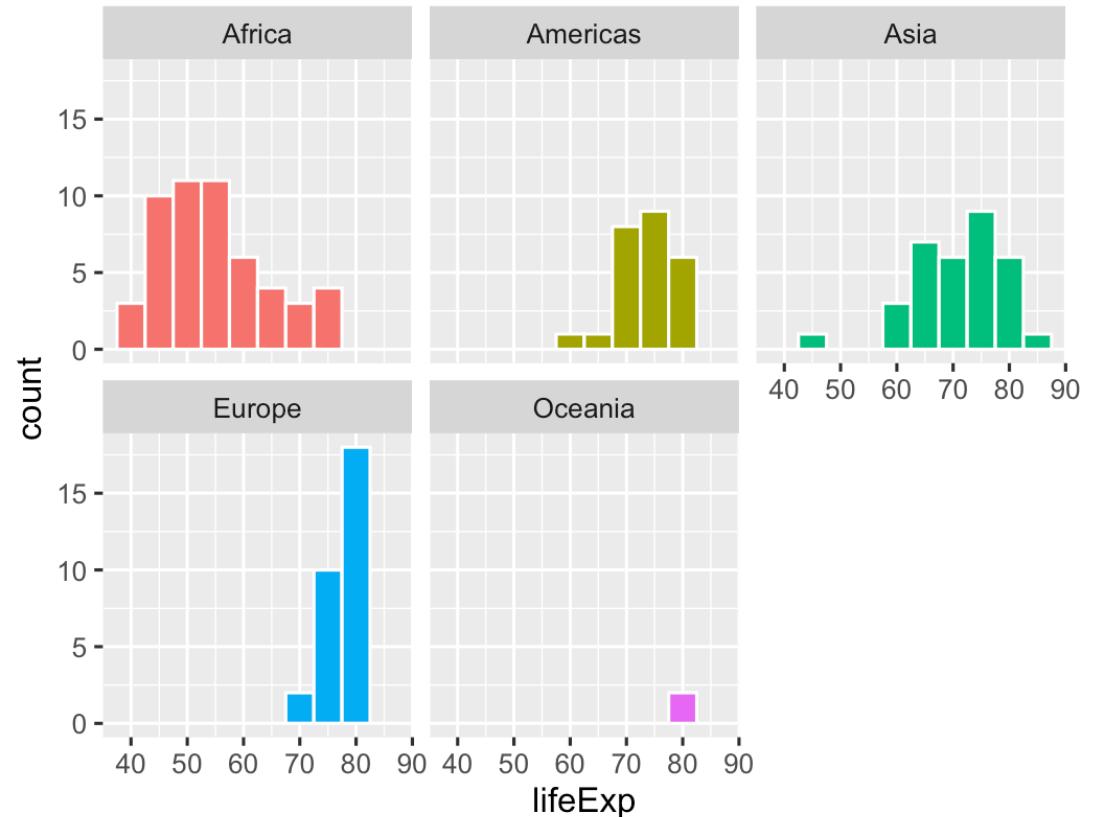
```
gapminder |>  
  filter(year %in% c(2002, 2007)) |>  
  ggplot(aes(x = gdpPerCap,  
             y = lifeExp,  
             color = continent,  
             size = pop)) +  
  geom_point() +  
  scale_x_log10() +  
  facet_wrap(vars(year), ncol = 1)
```



# 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

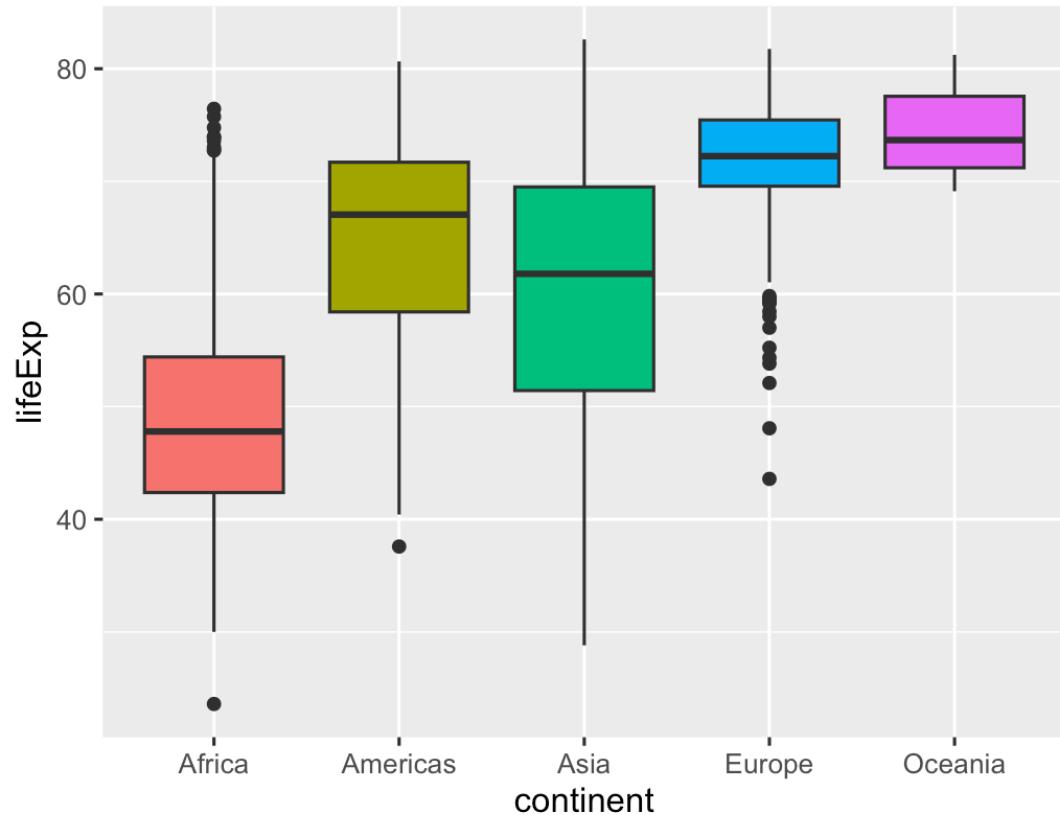
```
ggplot(data = gapminder_2007,  
       mapping = aes(x = lifeExp,  
                      fill = continent)) +  
  geom_histogram(binwidth = 5,  
                 color = "white") +  
  guides(fill = "none") + # Turn off legend  
  facet_wrap(vars(continent))
```



# Describing graphs with the grammar

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

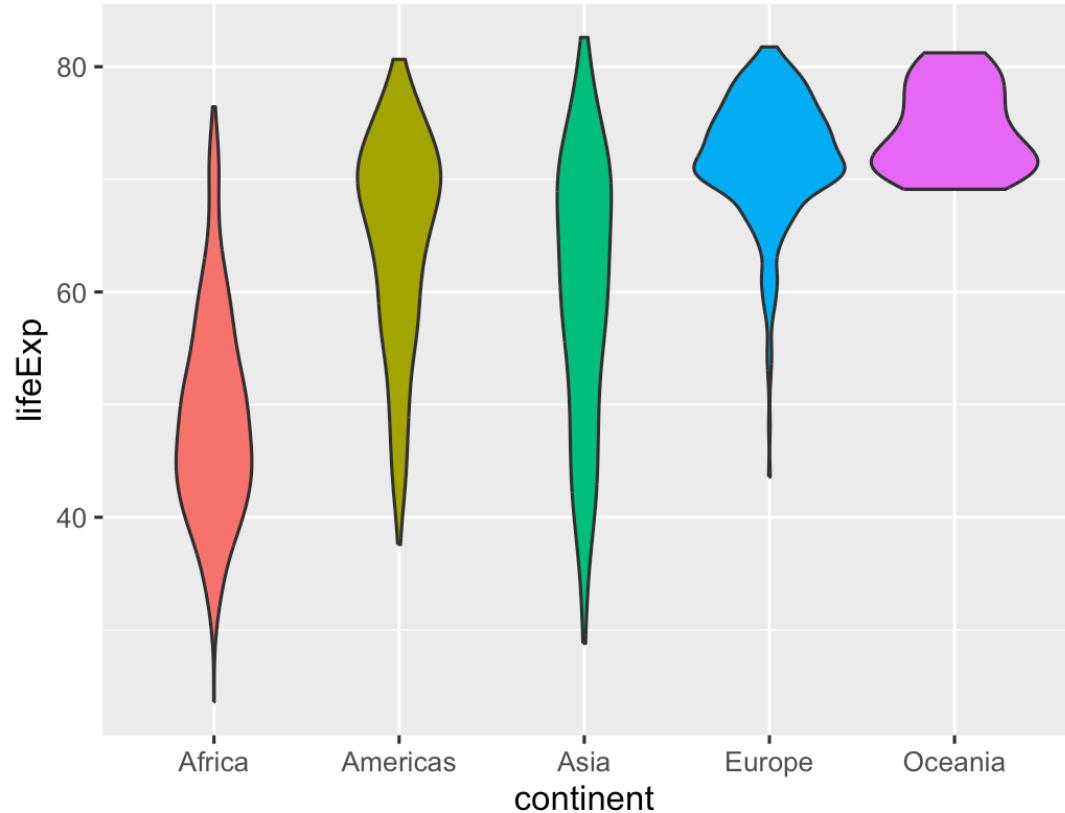
```
ggplot(data = gapminder,  
       mapping = aes(x = continent,  
                      y = lifeExp,  
                      fill = continent)) +  
  geom_boxplot() +  
  guides(fill = "none") # Turn off legend
```



# 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

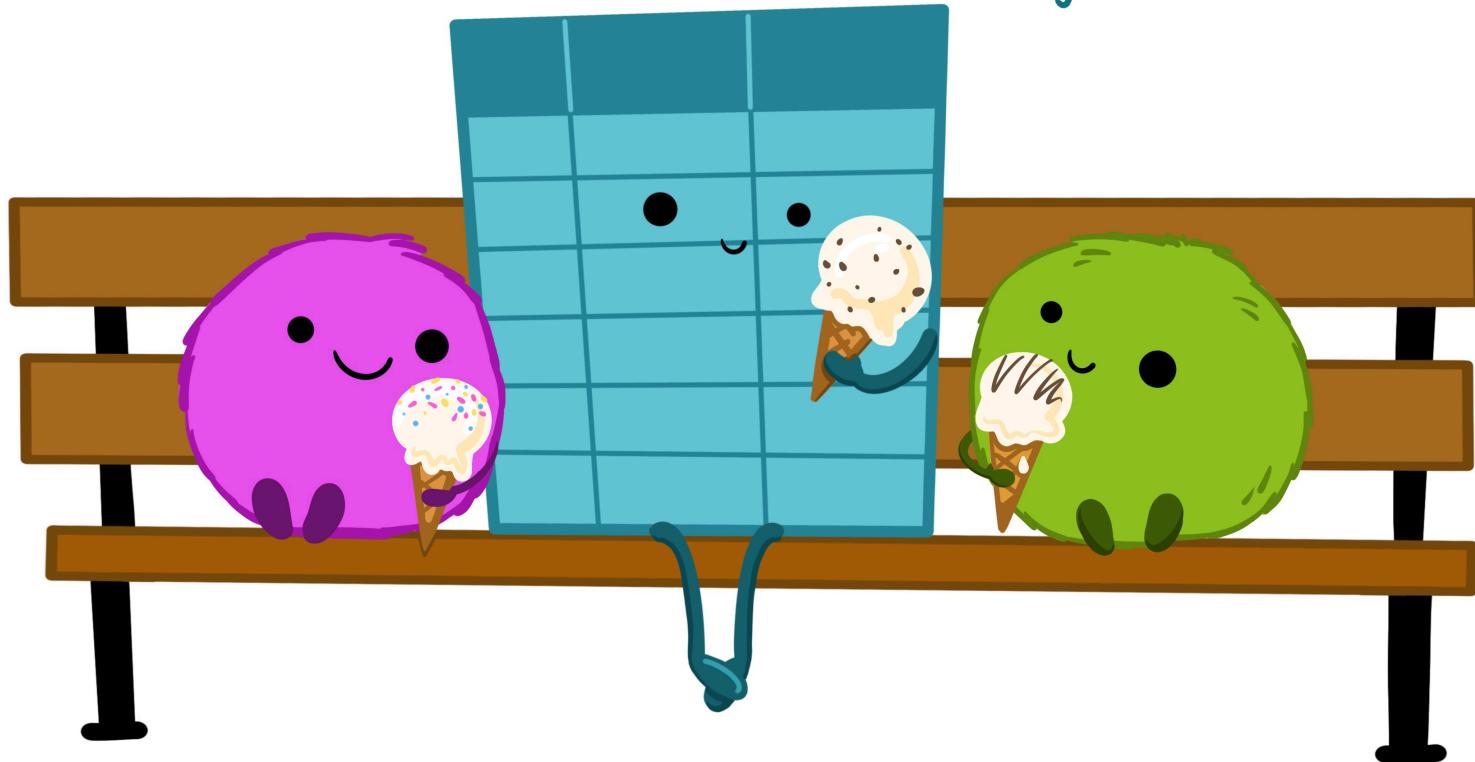
`ggplot()` is incredibly flexible

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-07-1: 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

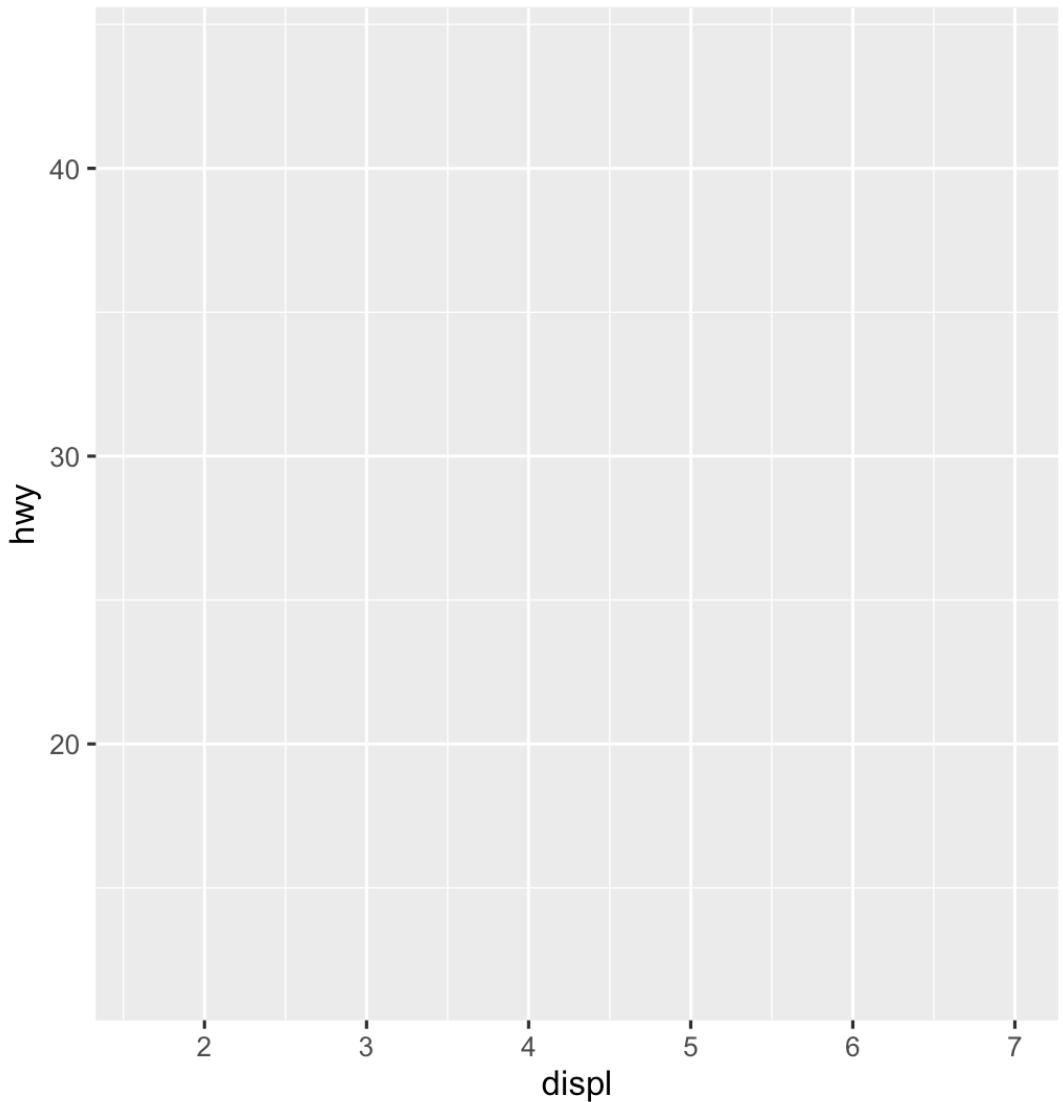
For illustrative purposes, let's use data on cars from the data frame

`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
...	...	...	...	...	...	...

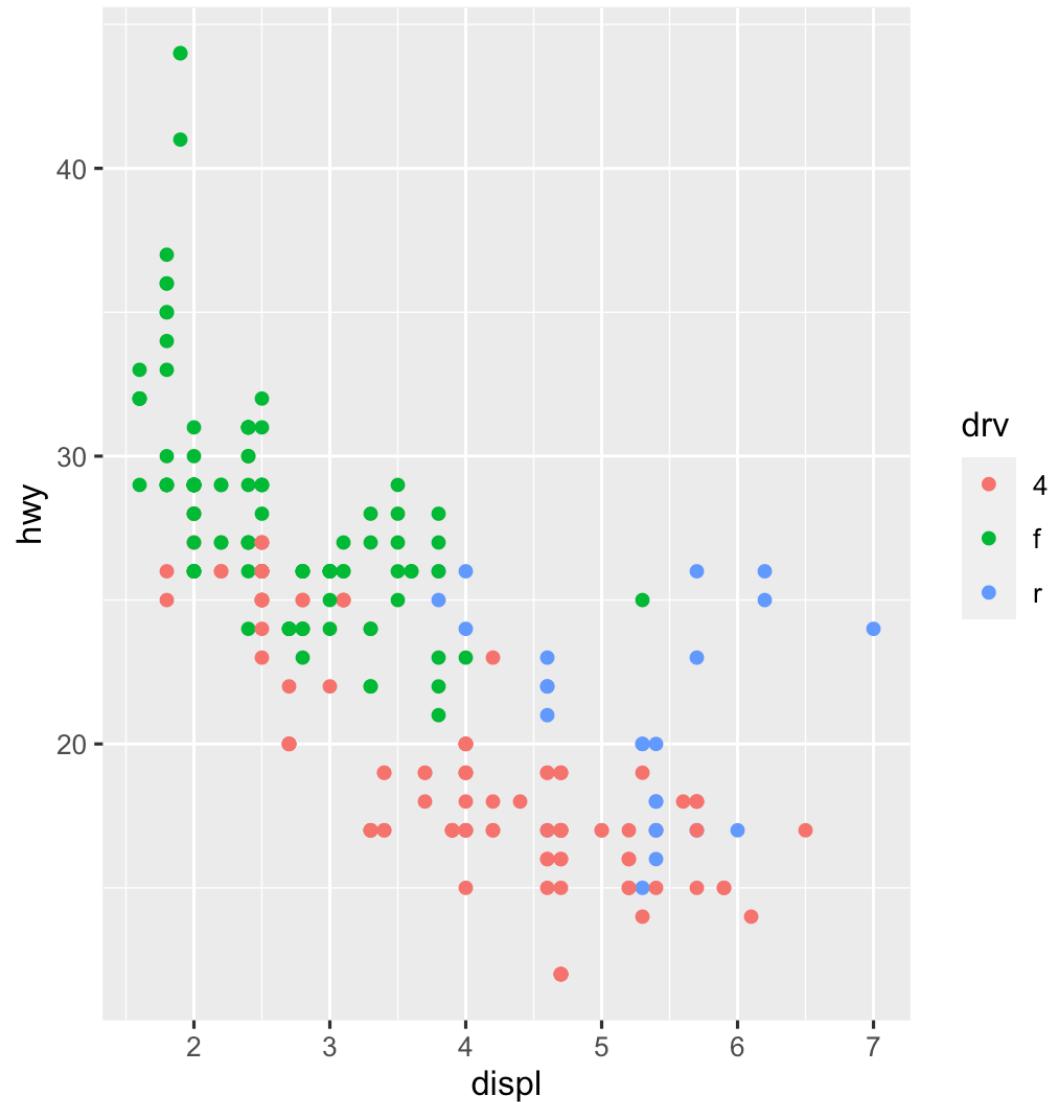
## Start with data and aesthetics

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



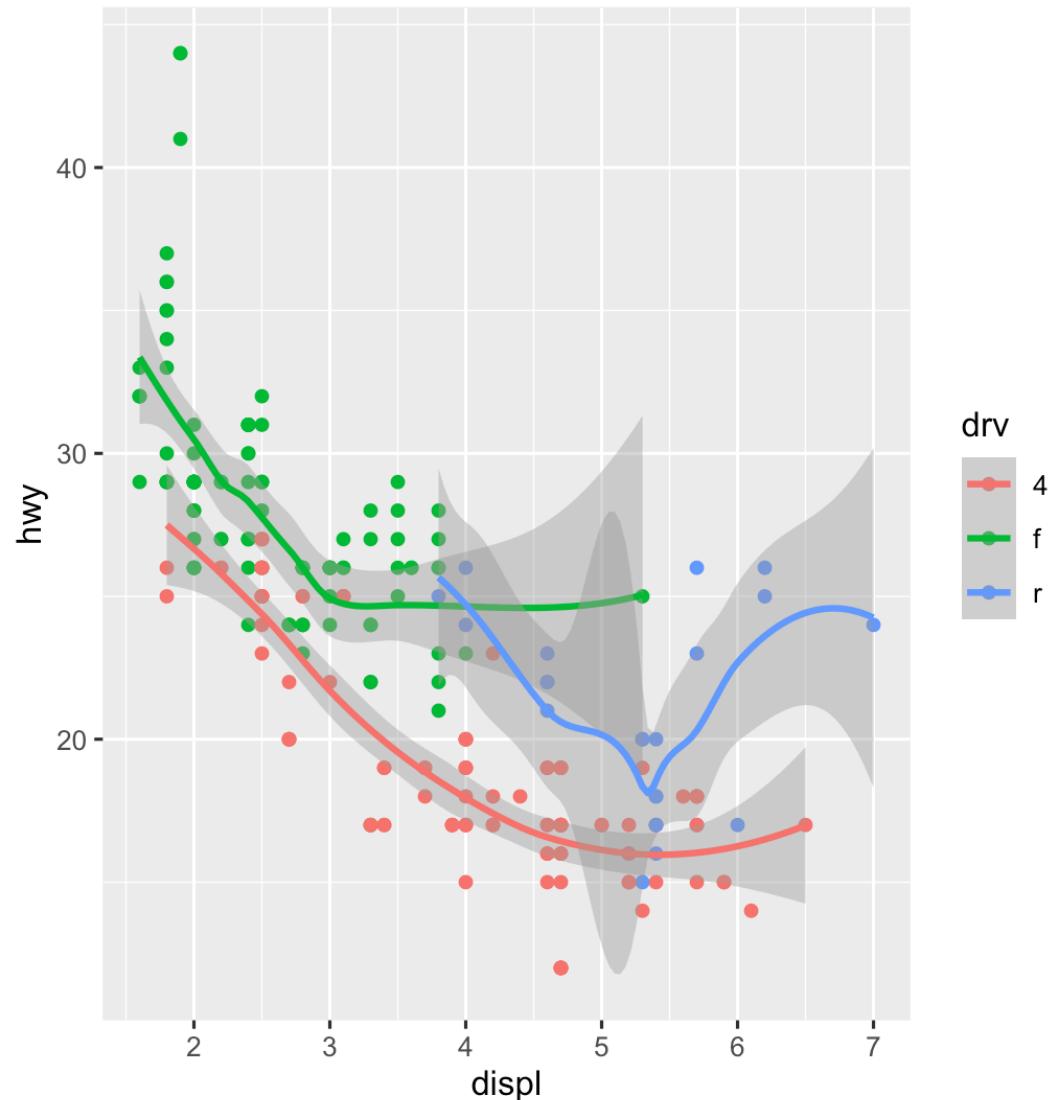
# Add a point geom

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



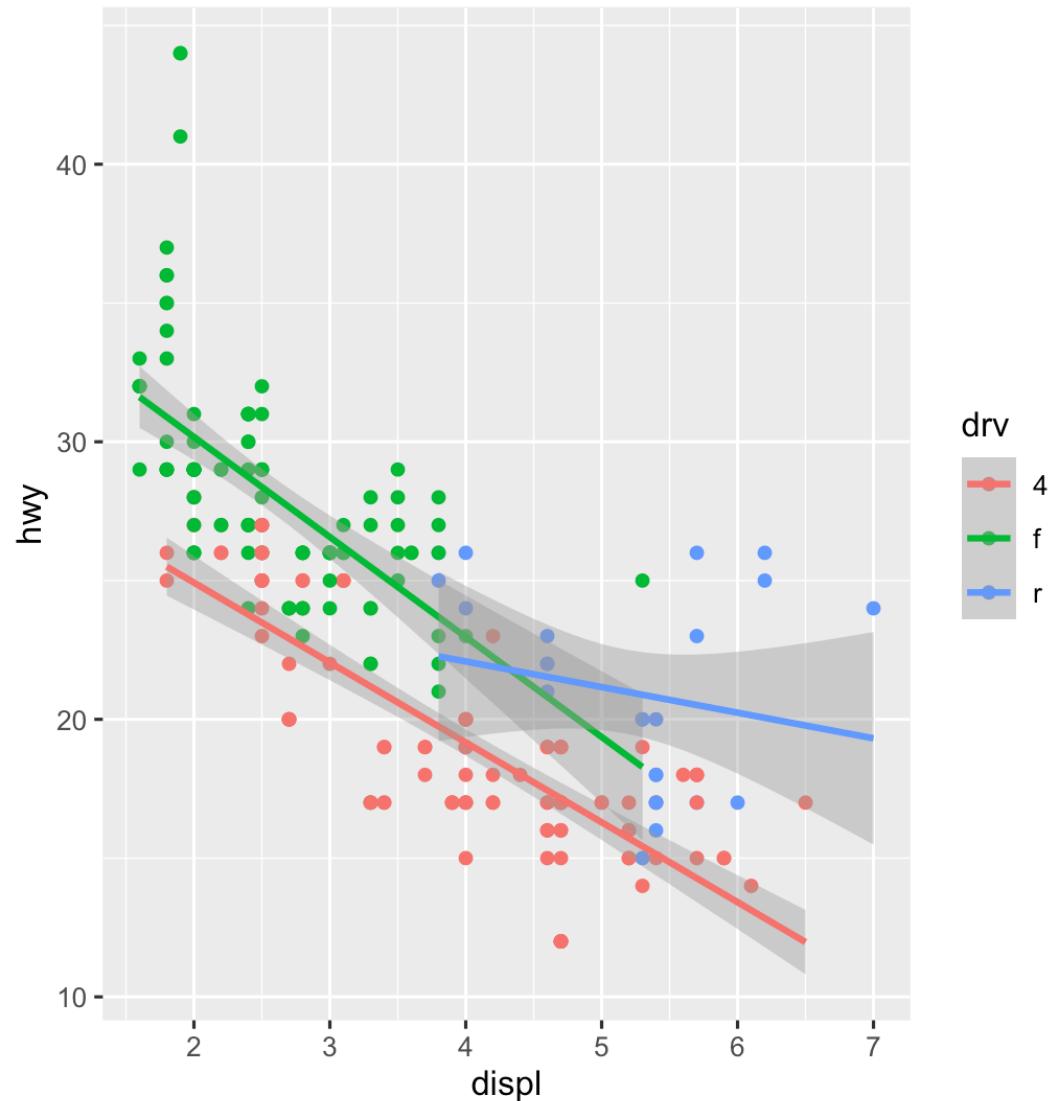
# Add a smooth geom

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



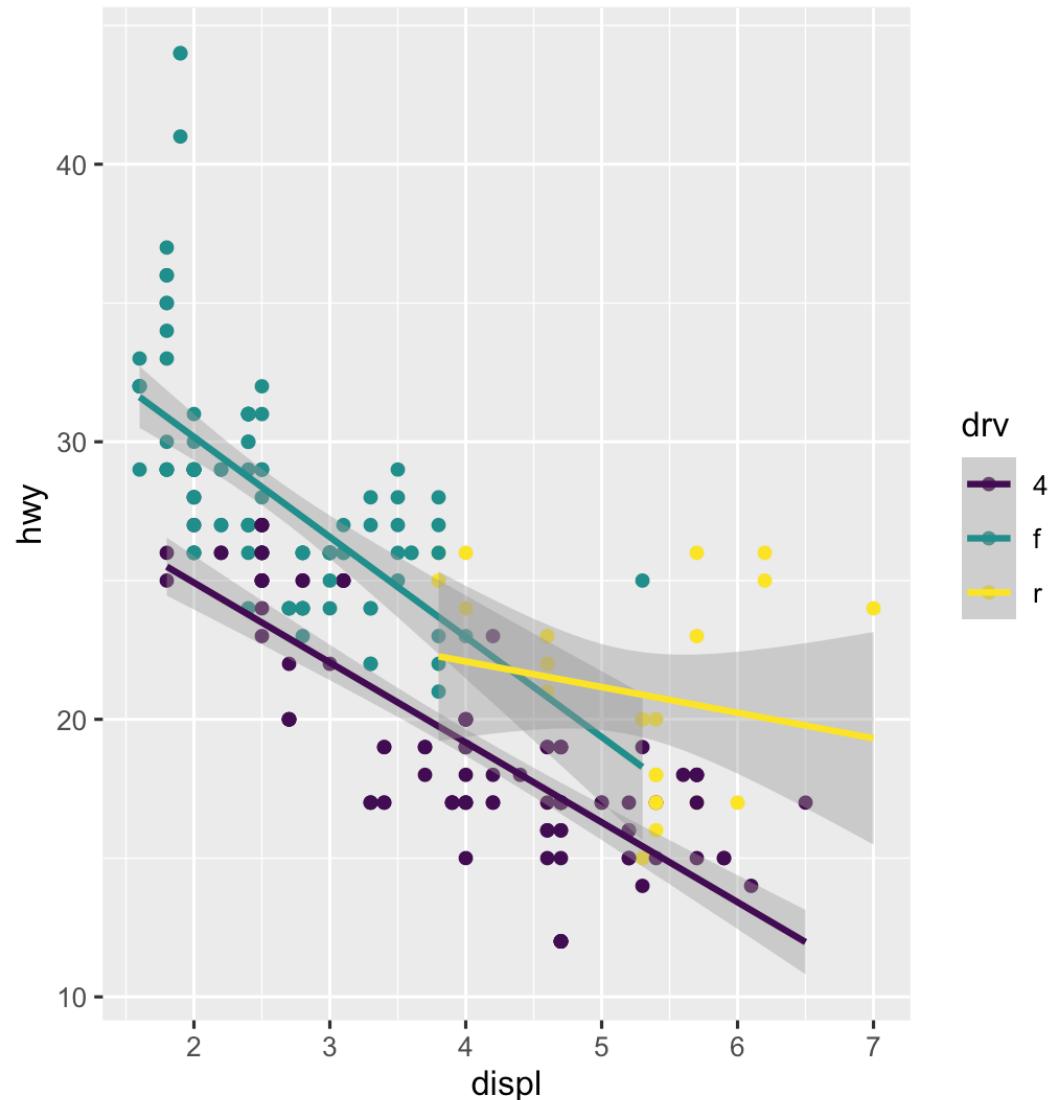
# Make it straight

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



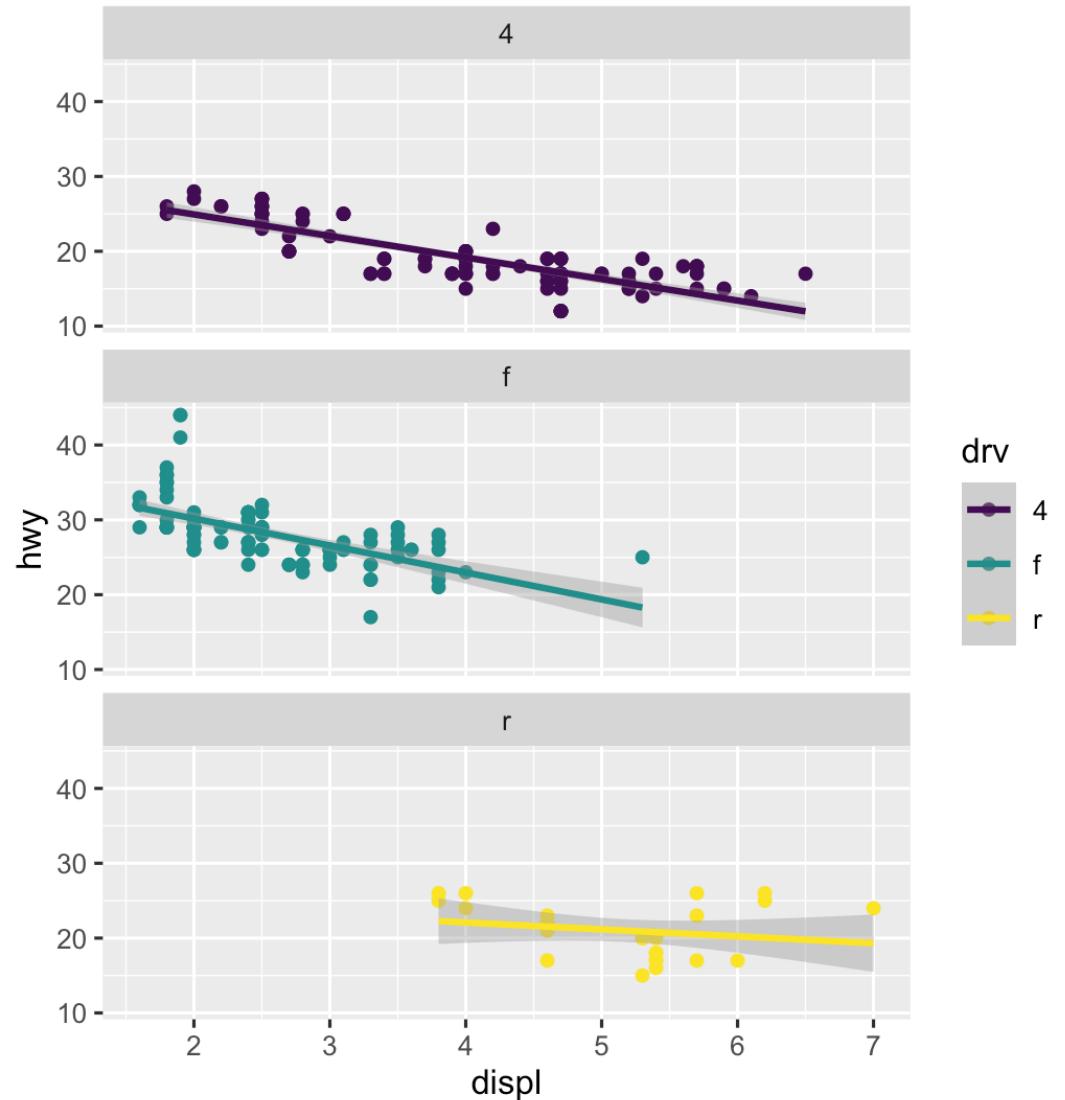
## Use a viridis color scale

```
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

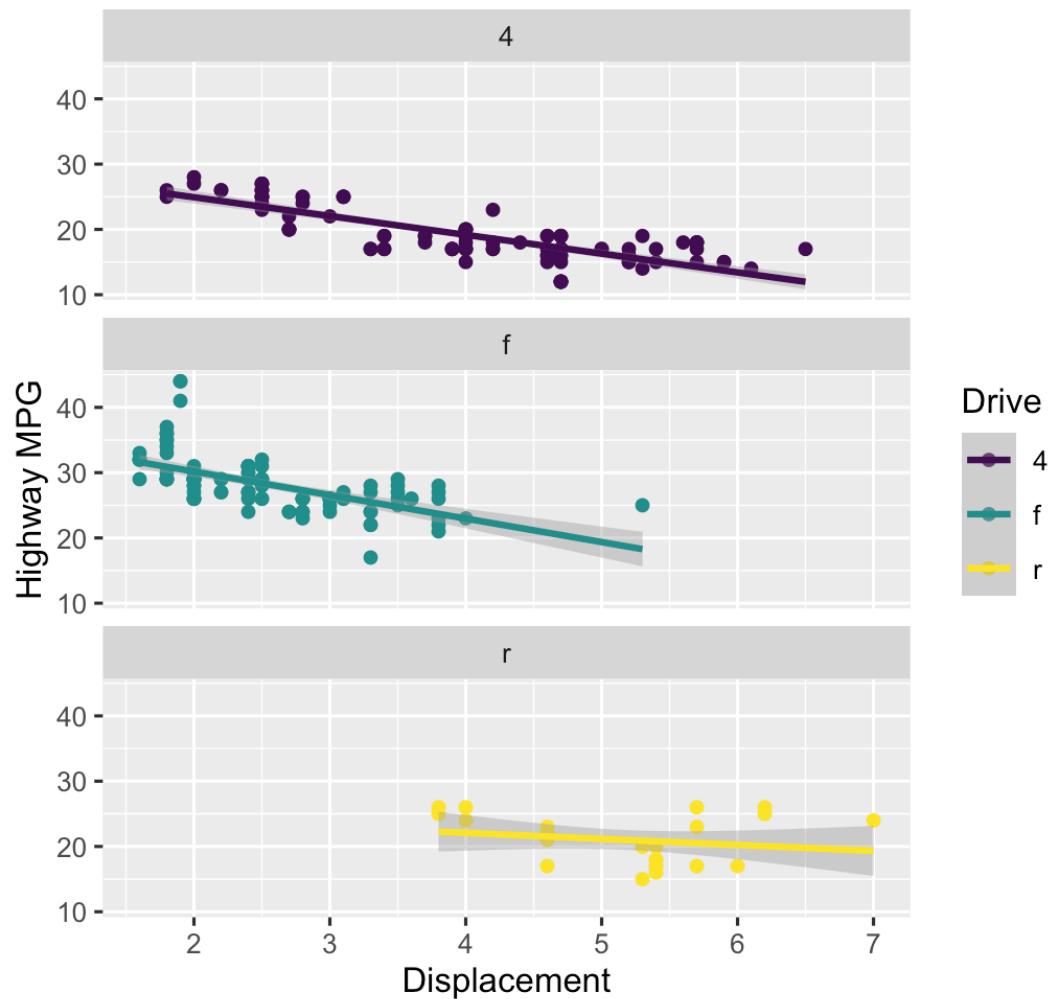
```
ggplot(data = mpg,  
       mapping = aes(x = displ,  
                      y = hwy,  
                      color = drv)) +  
  geom_point() +  
  geom_smooth(method = "lm") +  
  scale_color_viridis_d() +  
  facet_wrap(vars(drv), ncol = 1)
```



## 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 size")
```

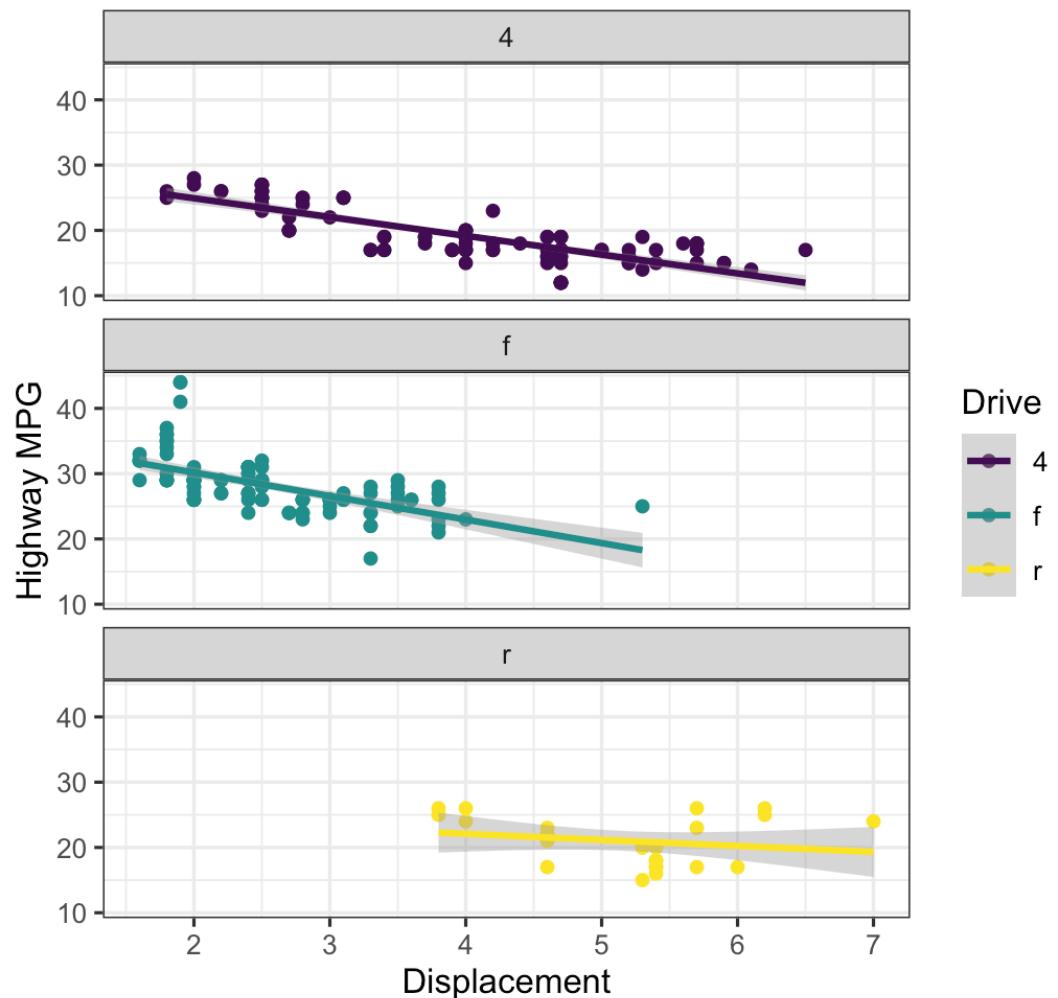
Larger engines use more fuel  
Displacement measures engine size



# 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  
theme_bw()
```

Larger engines use more fuel  
Displacement measures engine size

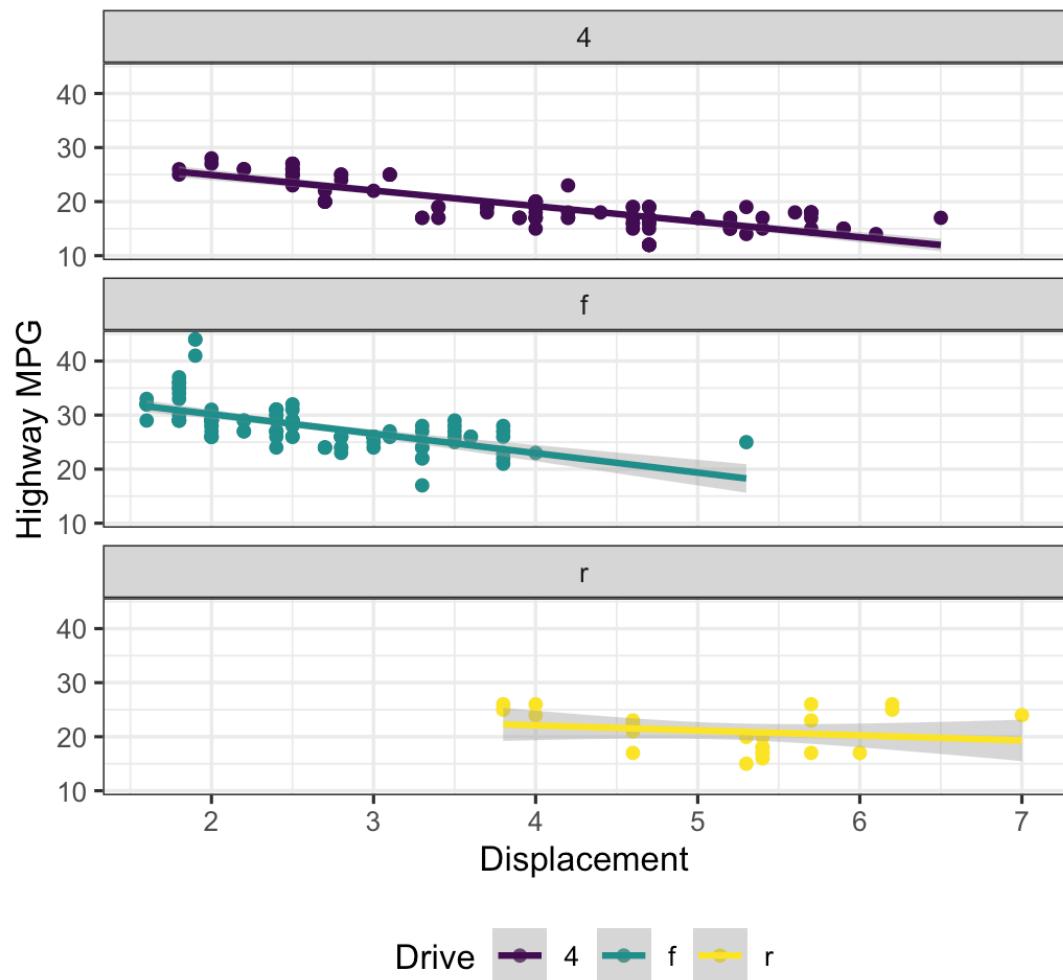


# Modify the theme

```
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  
theme_bw() +  
  theme(legend.position = "bottom",  
        plot.title = element_text(face = "bold"))
```

## Larger engines use more fuel

Displacement measures engine size

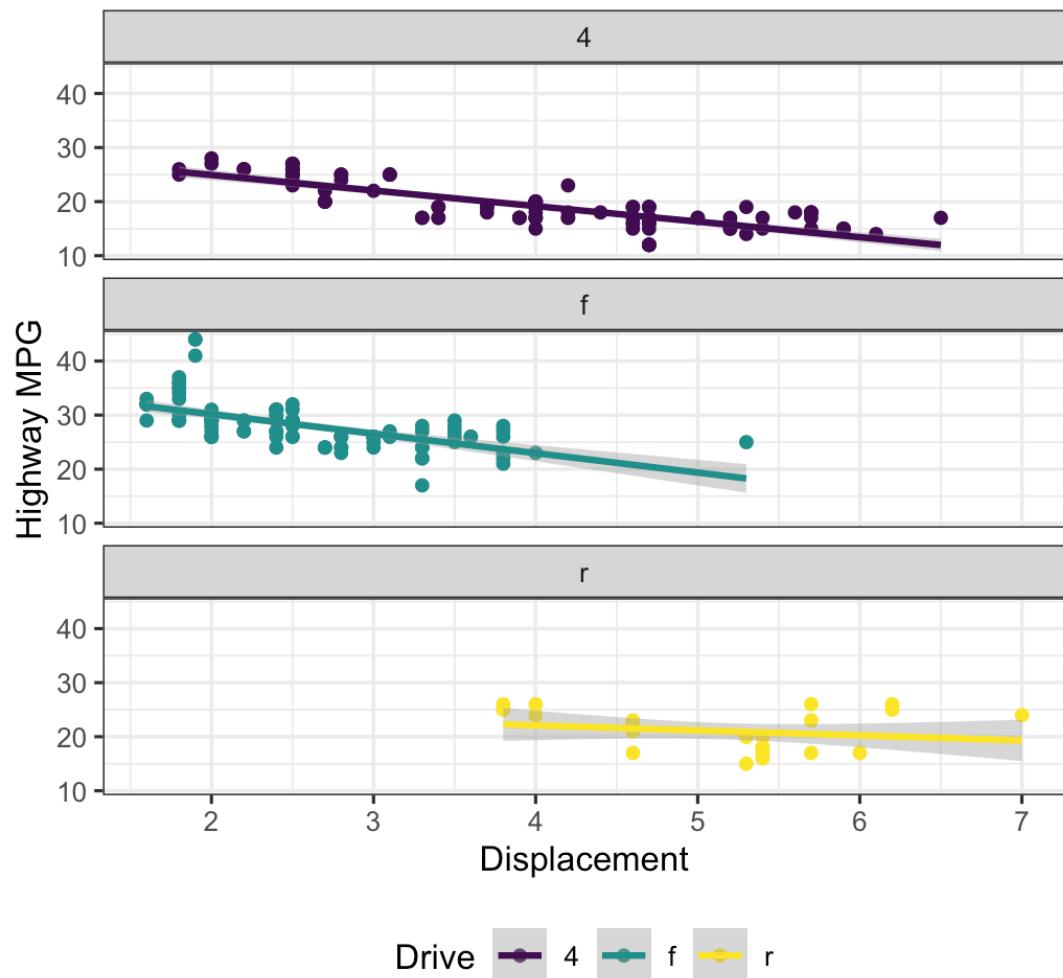


# Finished!

```
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  
theme_bw() +  
theme(legend.position = "bottom",  
     plot.title = element_text(face = "bold"))
```

## Larger engines use more fuel

Displacement measures engine size



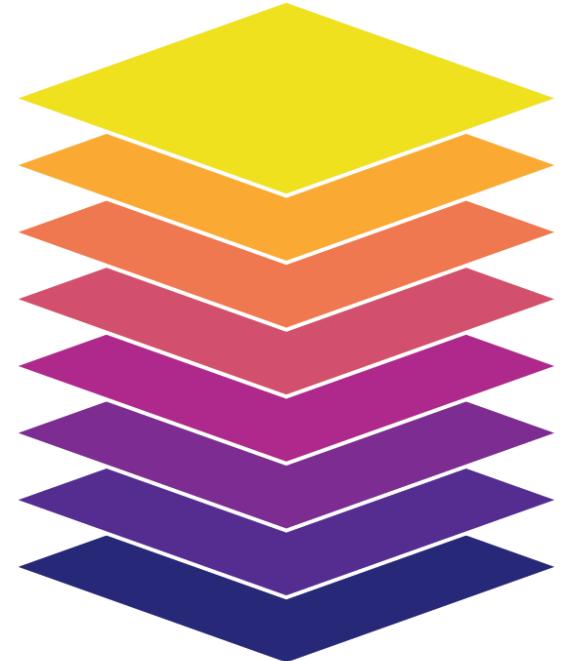
# Additional layers

# Additional layers

The next several slides contains a preview of additional layers

These are intended as a reference

Theme  
Labels  
Coordinates  
Facets  
Scales  
Geometries  
Aesthetics  
Data



# Scales

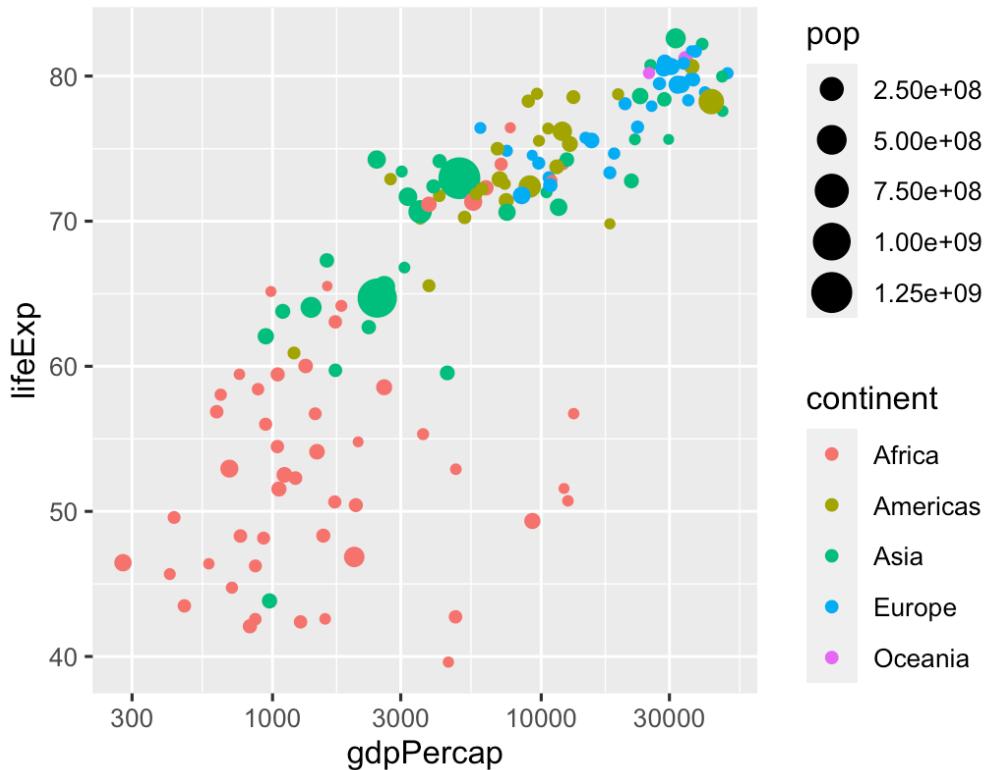
Scales change the properties of the variable mapping

---

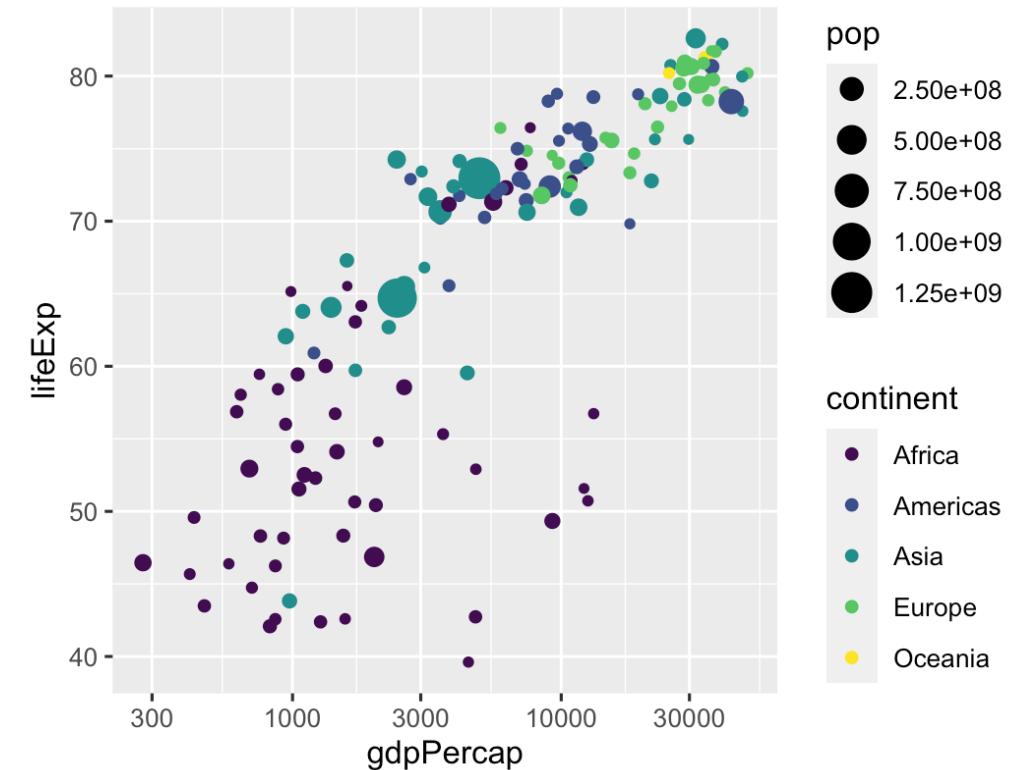
Example layer	What it does
<code>scale_x_continuous()</code>	Make the x-axis continuous
<code>scale_x_continuous(breaks = 1:5)</code>	Manually specify axis ticks
<code>scale_x_log10()</code>	Log the x-axis
<code>scale_color_gradient()</code>	Use a gradient
<code>scale_fill_viridis_d()</code>	Fill with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

# Scales

scale\_x\_log10()



scale\_color\_viridis\_d()



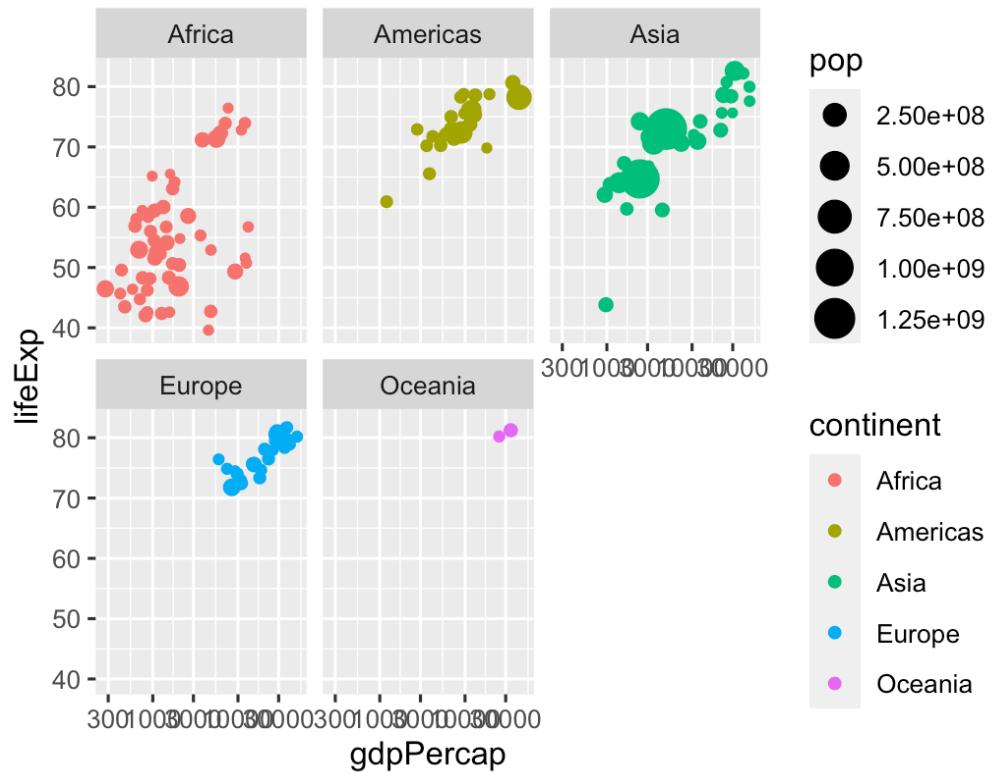
# Facets

Facets show subplots for different subsets of data

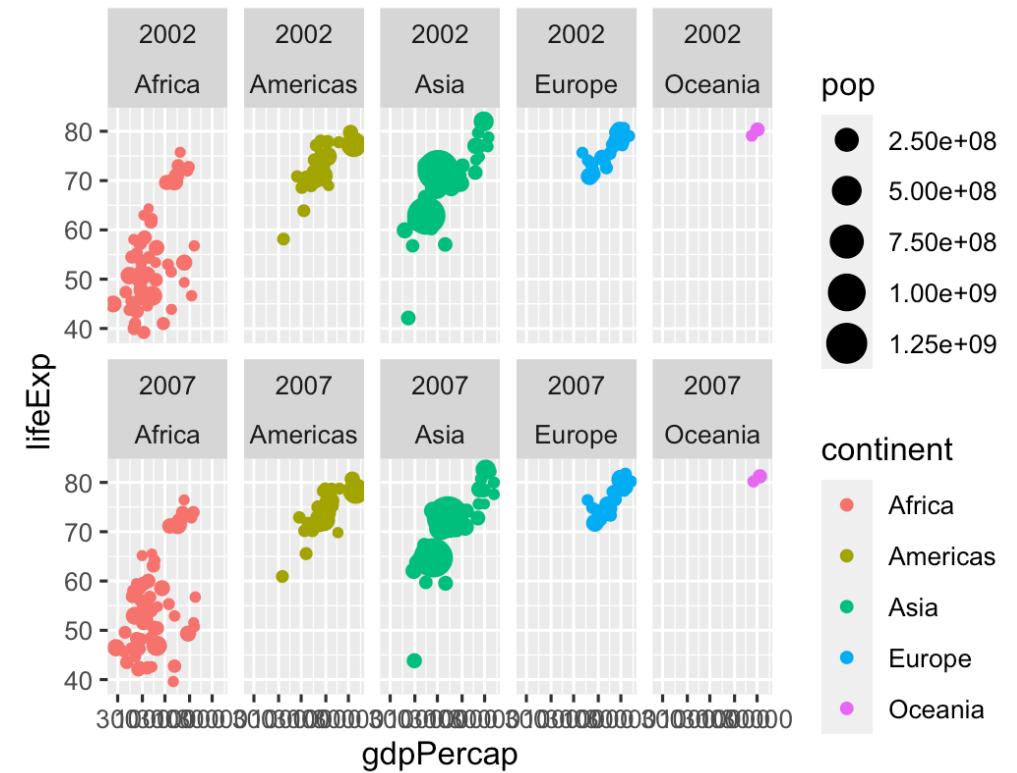
Example layer	What it does
<code>facet_wrap(vars(continent))</code>	Plot for each continent
<code>facet_wrap(vars(continent, year))</code>	Plot for each continent/year
<code>facet_wrap(..., ncol = 1)</code>	Put all facets in one column
<code>facet_wrap(..., nrow = 1)</code>	Put all facets in one row

# Facets

facet\_wrap(vars(continent))



facet\_wrap(vars(continent, year))



# Coordinates

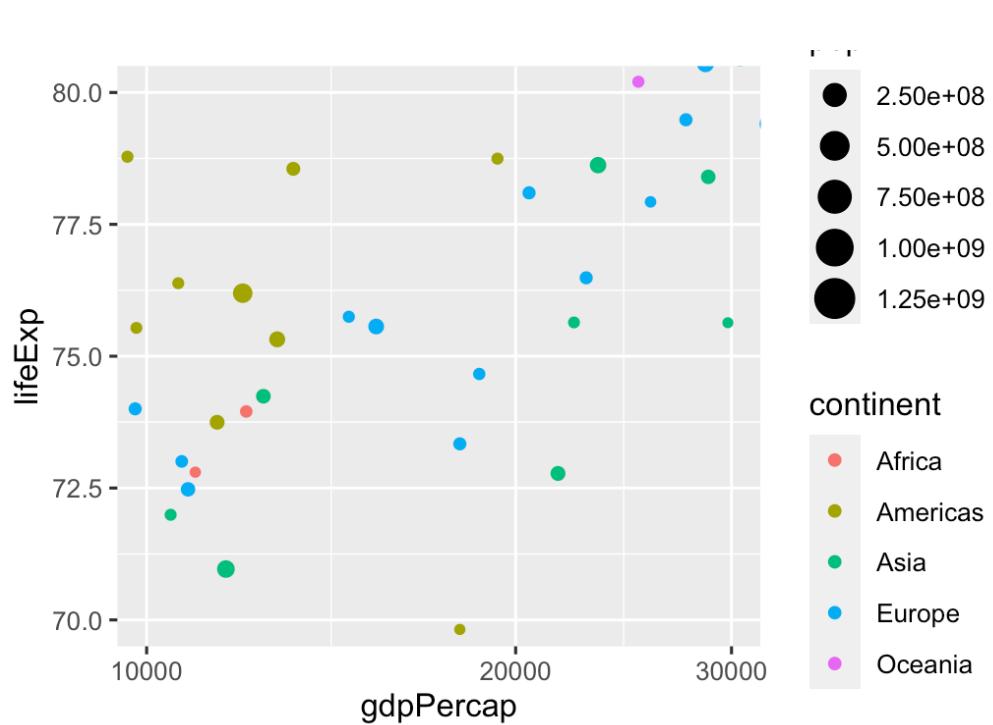
Change the coordinate system

---

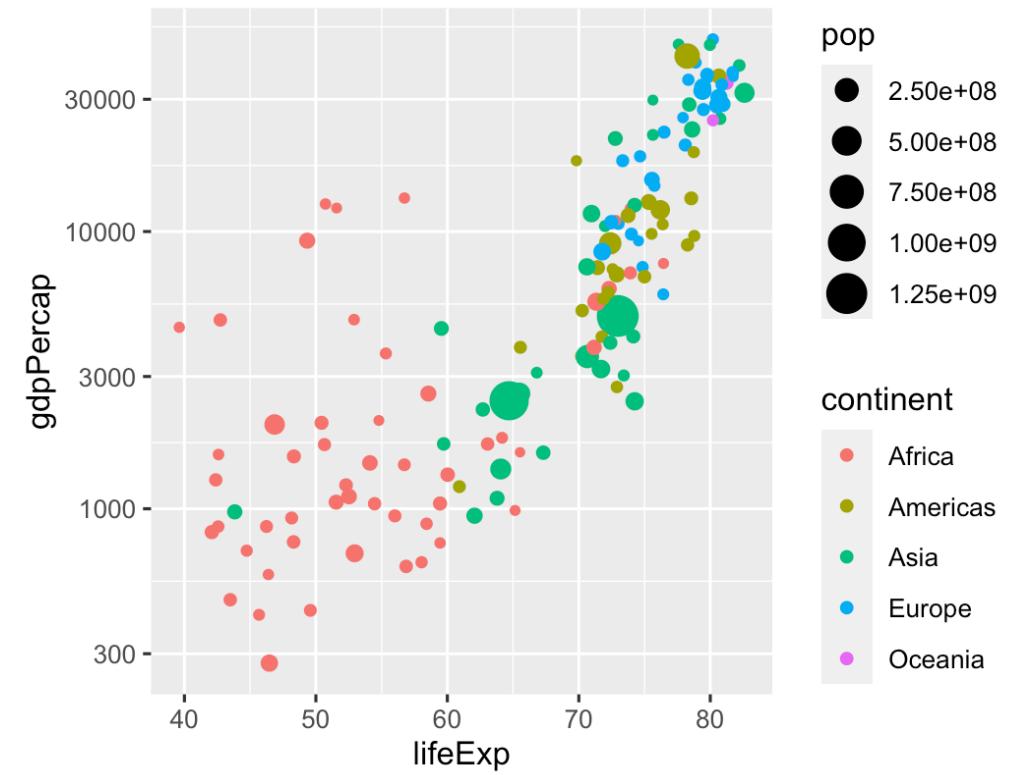
Example layer	What it does
<code>coord_cartesian(ylim = c(1, 10))</code>	Zoom in where y is 1–10
<code>coord_flip()</code>	Switch x and y
<code>coord_polar()</code>	Use polar coordinates

# Coordinates

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



```
coord_flip()
```



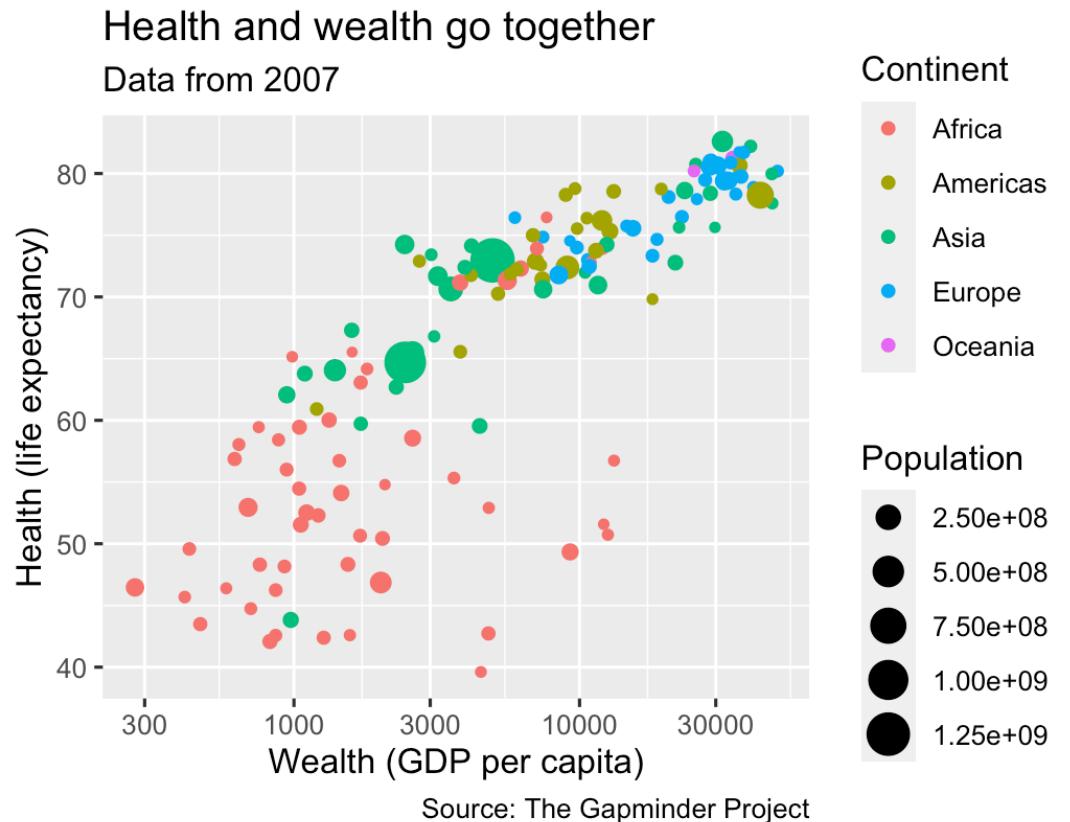
# Labels

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

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

# Labels

```
gapminder_2007 |>  
  ggplot(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

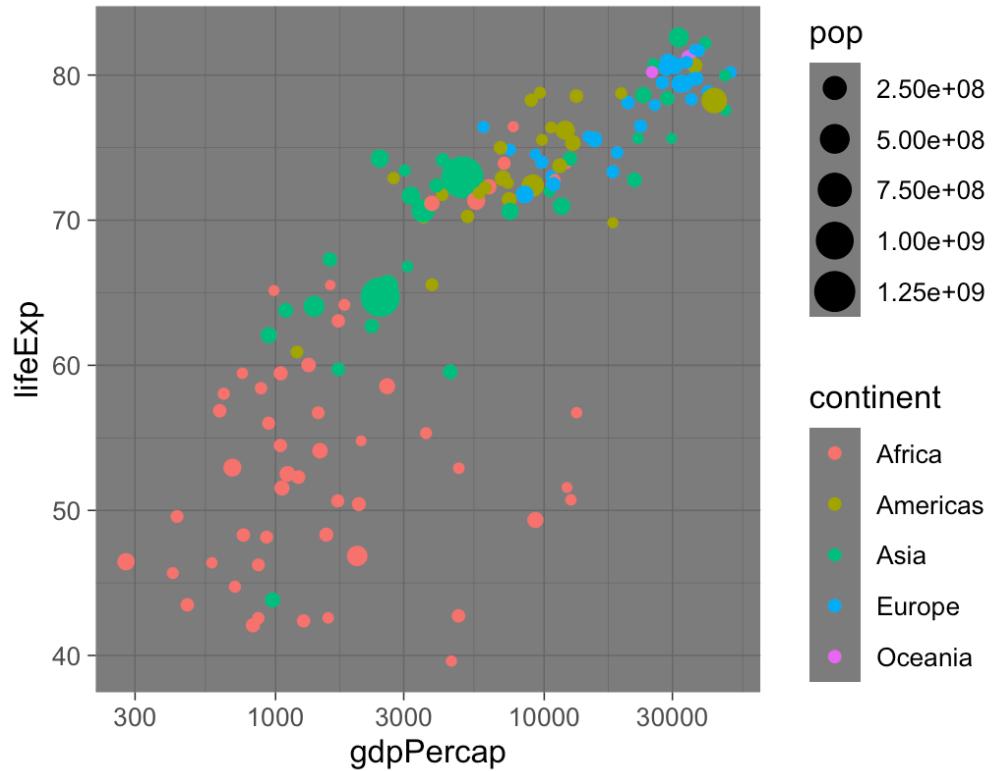
---

Example layer	What it does
<code>theme_grey()</code>	Default grey background
<code>theme_bw()</code>	Black and white
<code>theme_dark()</code>	Dark
<code>theme_minimal()</code>	Minimal

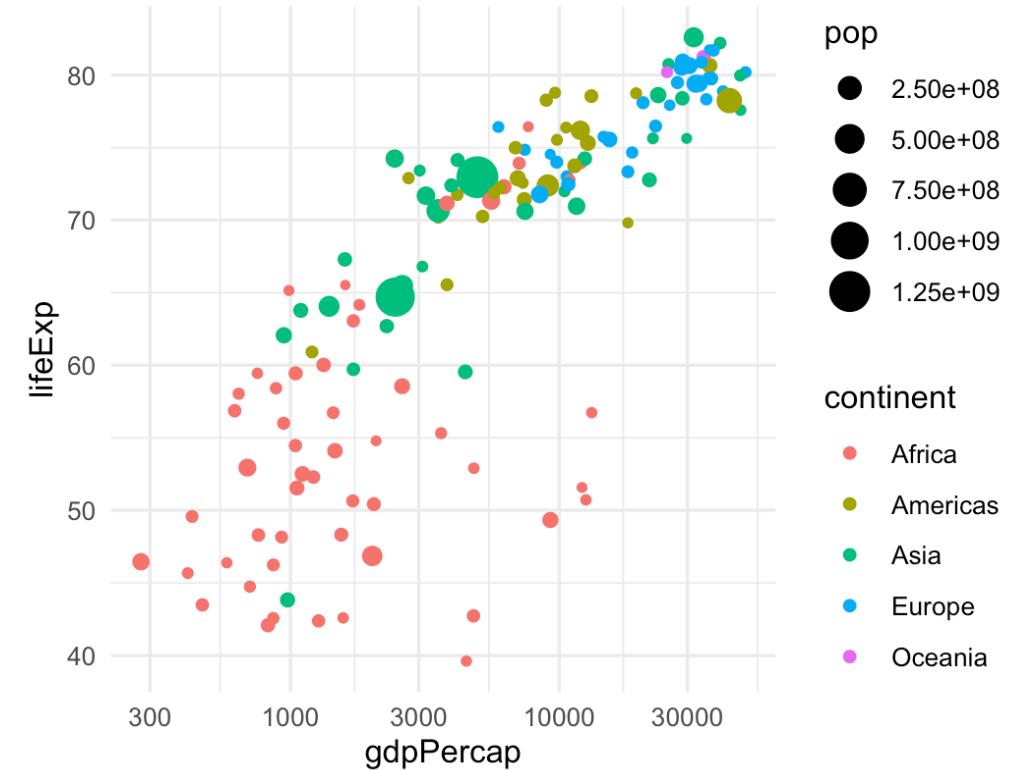
---

# Theme

theme\_dark()



theme\_minimal()



# Theme

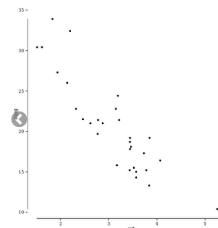
There are collections of pre-built themes online, like **the ggthemes package**

## ggthemes



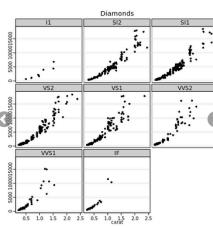
**theme\_wsj**

Wall Street Journal theme



**theme\_tufte**

Tufte Maximal Data, Minimal Ink Theme



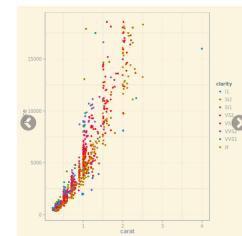
**theme\_stata**

Themes based on Stata graph schemes



**theme\_solid**

Theme with nothing other than a background color



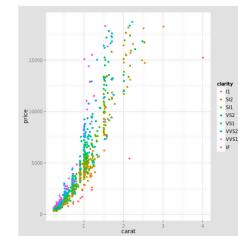
**theme\_solarized**

ggplot color themes based on the Solarized palette



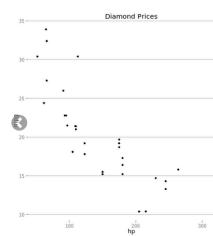
**theme\_map**

Clean theme for maps



**theme\_igray**

Inverse gray theme

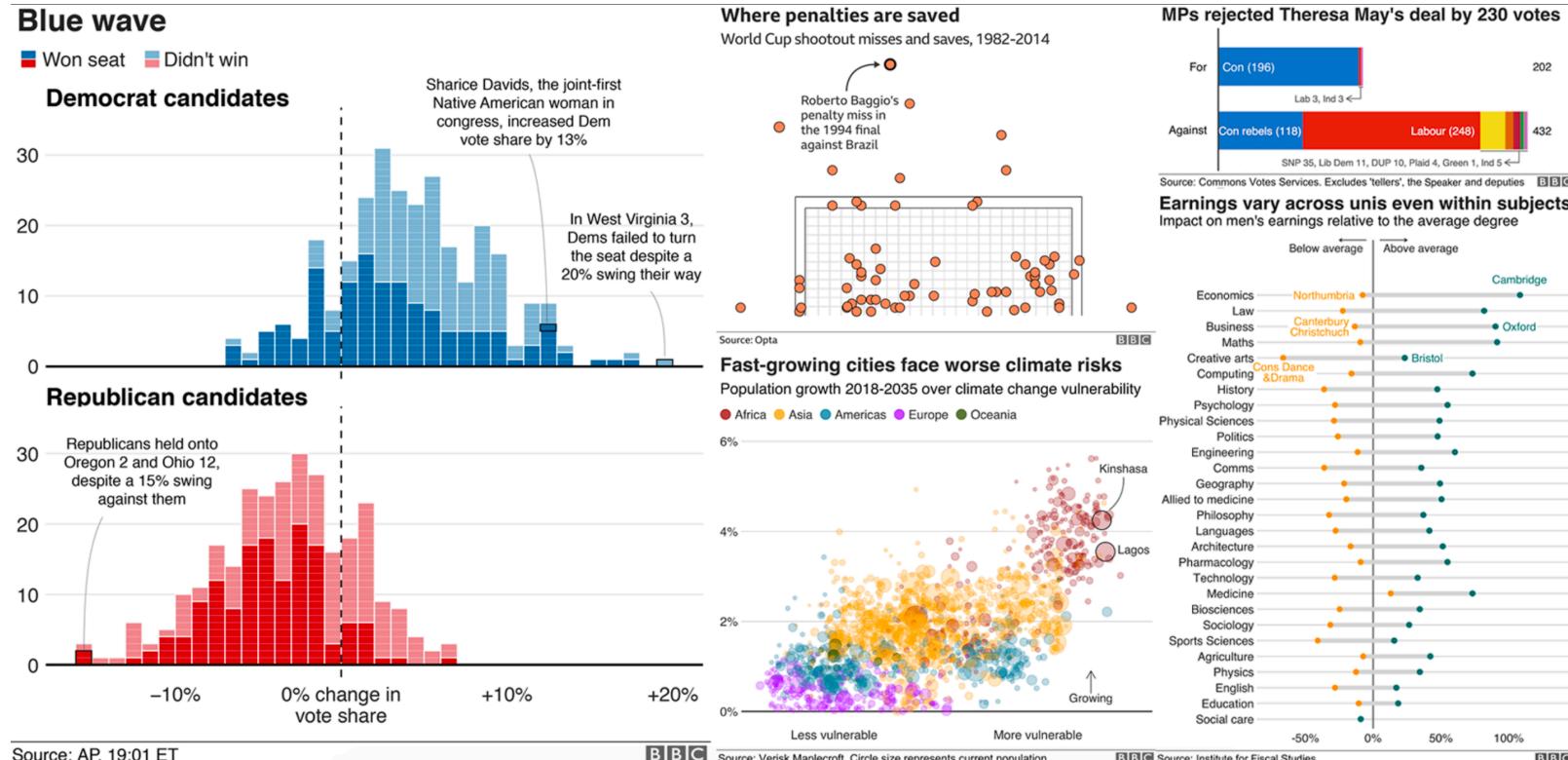


**theme\_hc**

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

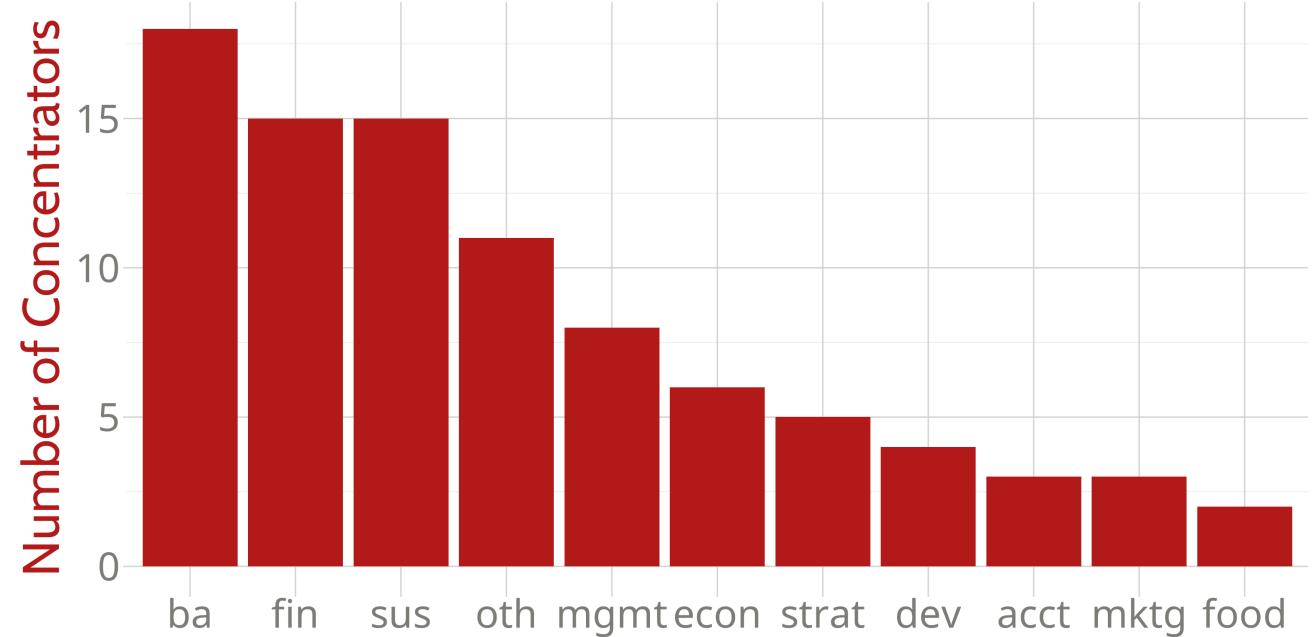
See [the ggplot2 documentation](#) for examples of everything you can do

# Prologue

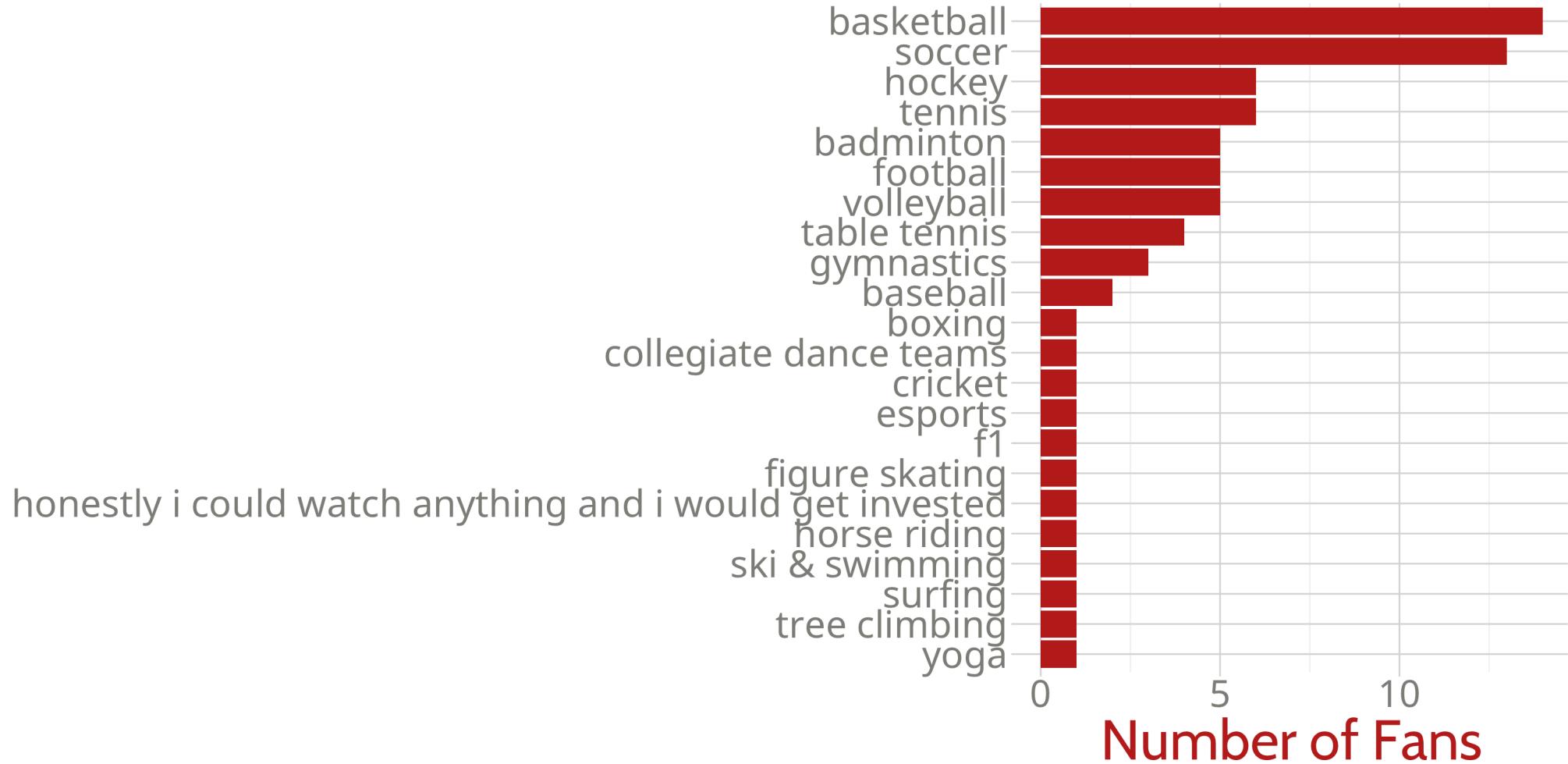
# Remember our concentrations?

How might we visualize these amounts?

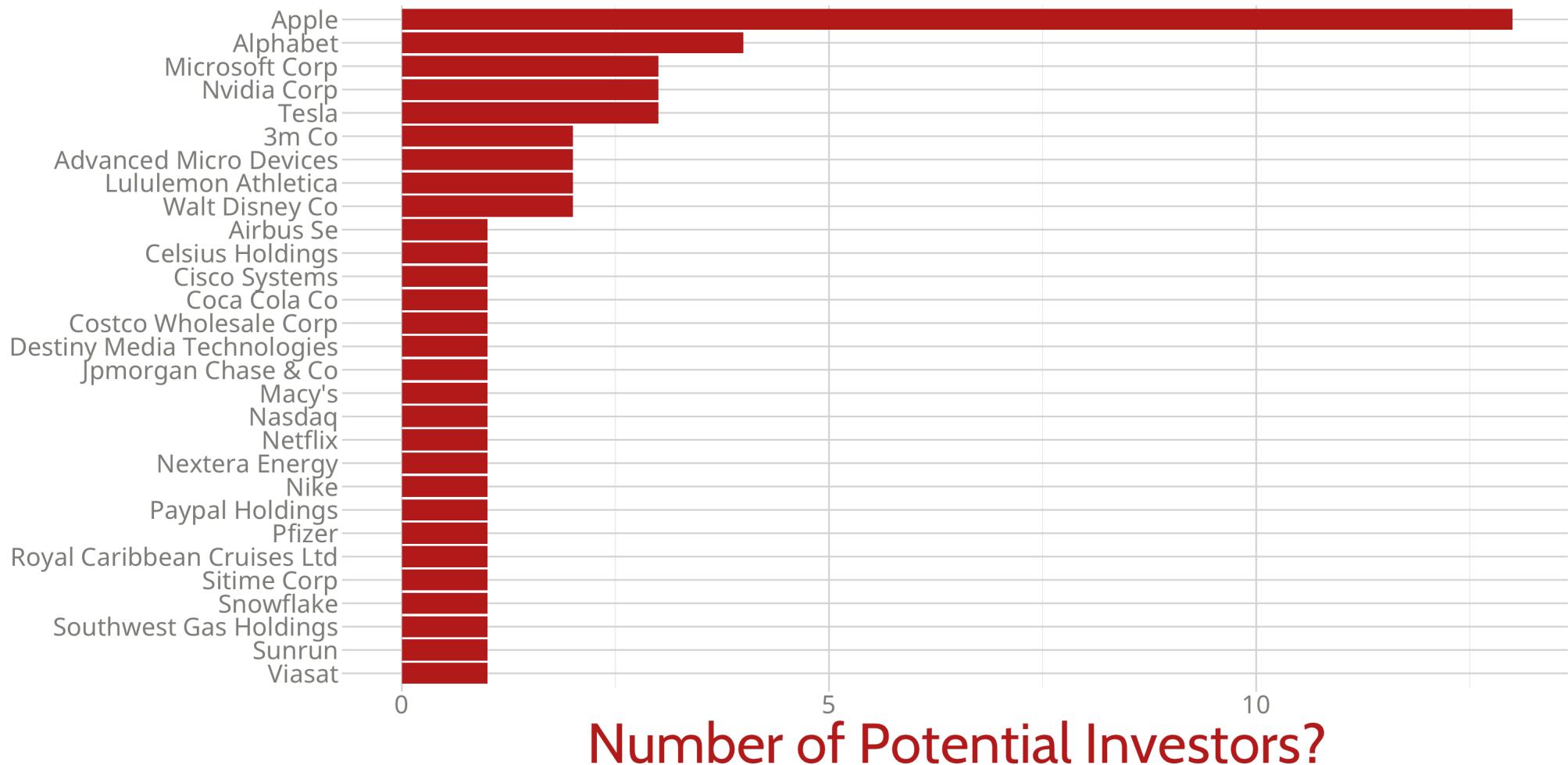
```
## # A tibble: 11 × 2
##   concentration count
##   <chr>          <int>
## 1 ba              18
## 2 sus             15
## 3 fin             15
## 4 oth              11
## 5 mgmt            8
## 6 econ             6
## 7 strat            5
## 8 dev              4
## 9 acct             3
## 10 mktg            3
## 11 food             2
```



# We could do the same thing with sports



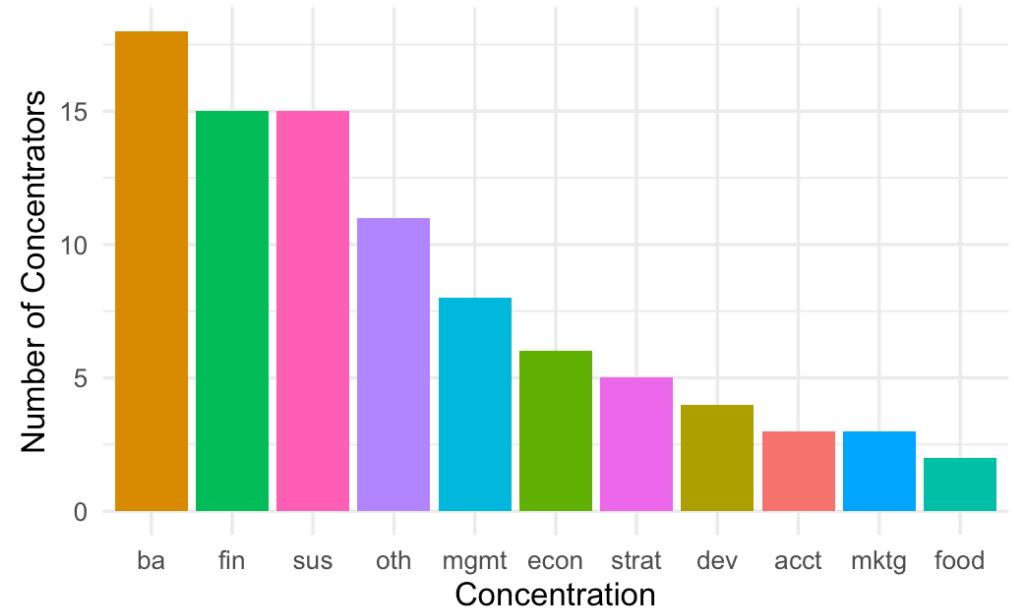
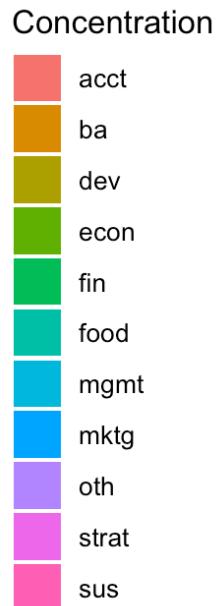
# What are our favorite public companies?



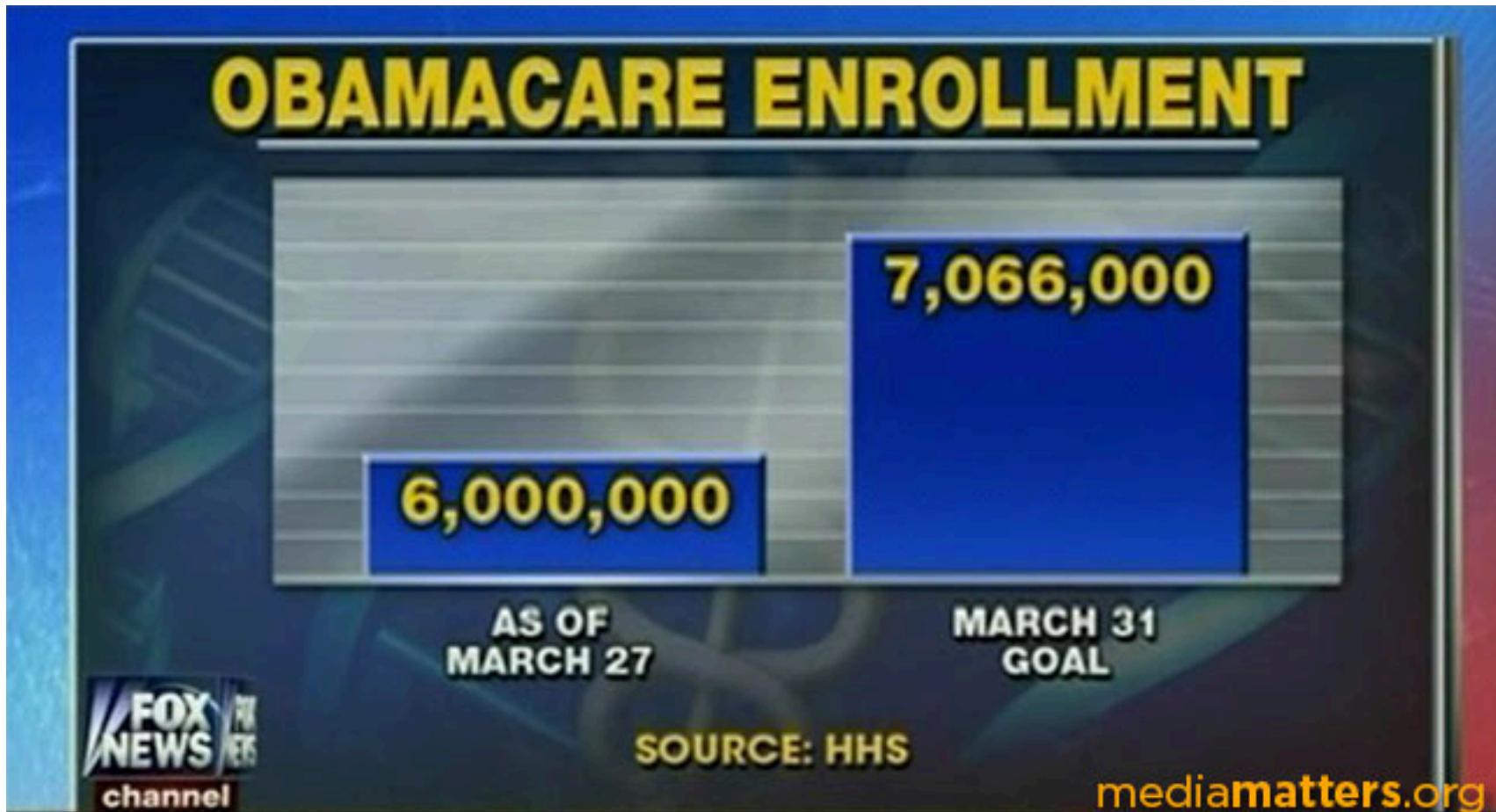
# Amounts

# Yay bar plots!

We are a lot better at visualizing line lengths than angles and areas



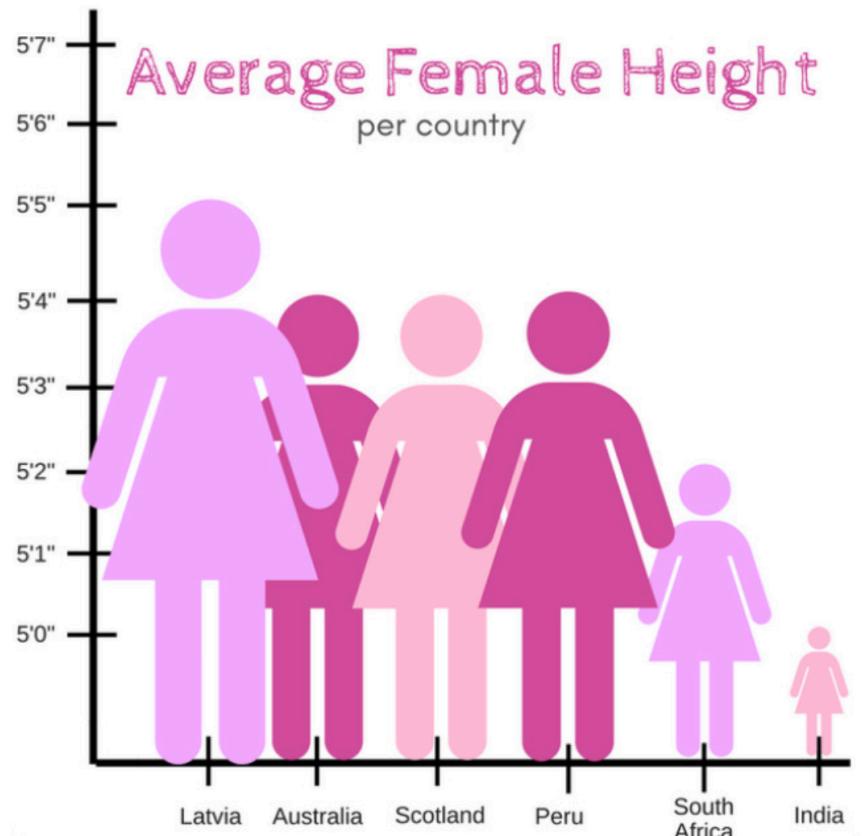
Oh no bar plots!



# What went wrong?



# What went wrong?

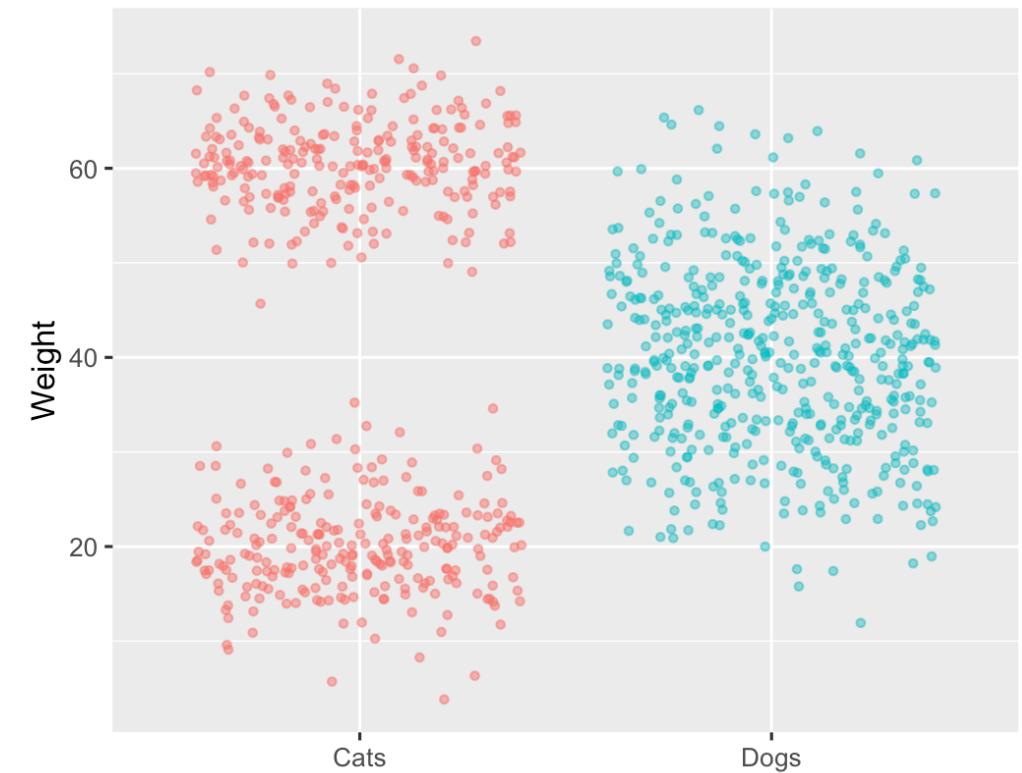
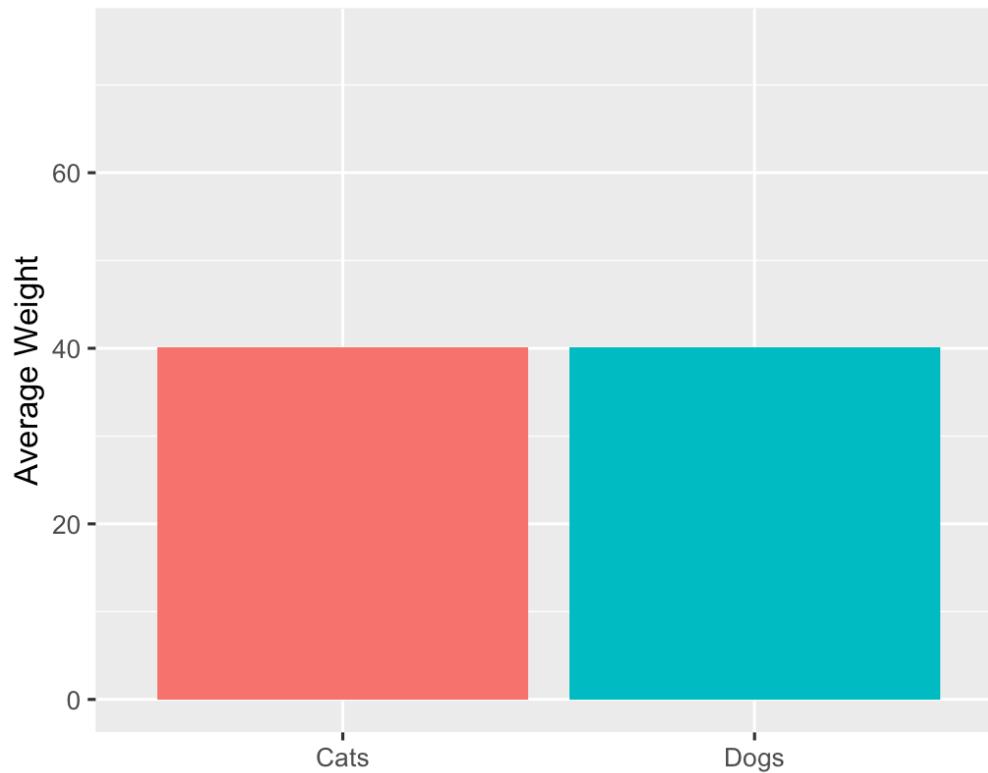


**At least two problems:**

1. truncated y axis
2. area scales faster than height

This terrible figure brought to you by a former AEM 2850 / 5850 student!

# Bar plots and summary statistics



# General rules for bar charts

Useful when the length of the bar is all that matters

Bar charts should always start at zero

- Or: don't use bars!

Don't use bars for summary statistics. You throw away too much information.

- We will come back to visualizing distributions / uncertainty next week

# Plotting amounts using ggplot

# Plotting amounts using ggplot

We'll use a summarized version of the gapminder dataset for examples

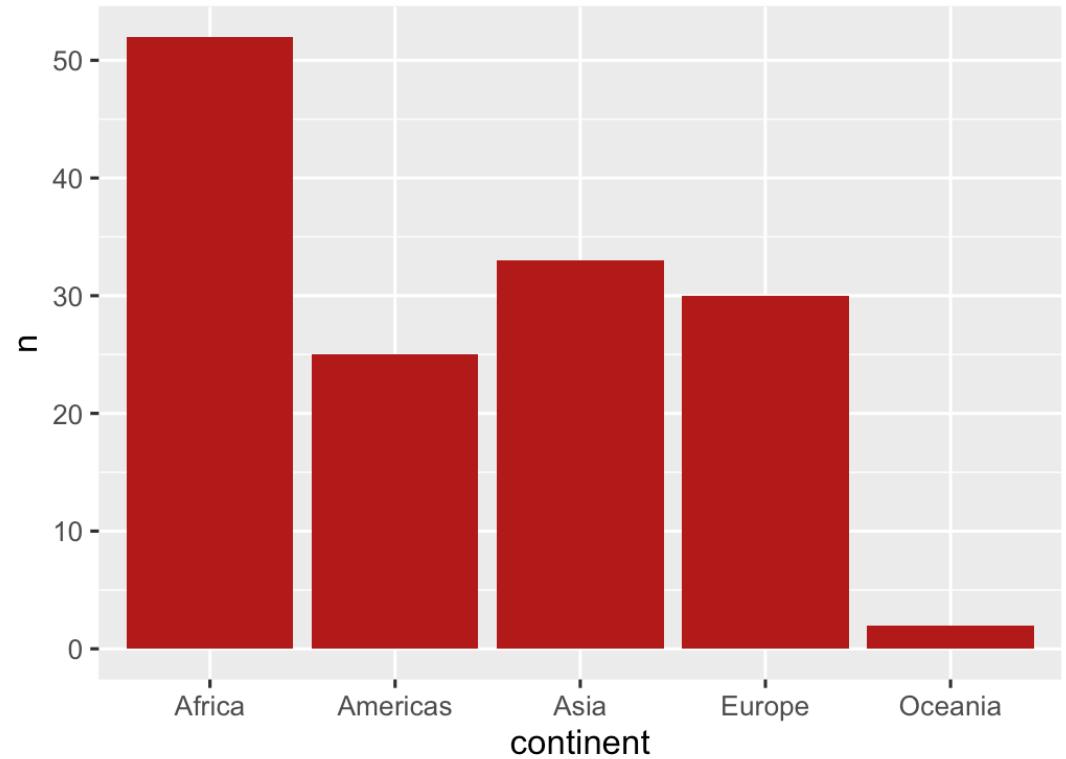
```
library(gapminder)
gapminder_continents <- gapminder |>
  filter(year == 2007) |> # only look at 2007
  count(continent) |>      # count the number of countries per continent
  arrange(desc(n))          # sort by count, descending
gapminder_continents
```

```
## # A tibble: 5 × 2
##   continent     n
##   <fct>     <int>
## 1 Africa       52
## 2 Asia         33
## 3 Europe       30
## 4 Americas     25
## 5 Oceania      2
```

# Start with a simple bar plot

```
gapminder_continents |>  
  ggplot(aes(x = continent, # continent to x  
             y = n)) +          # n countries to y  
  geom_col()               # add bars
```

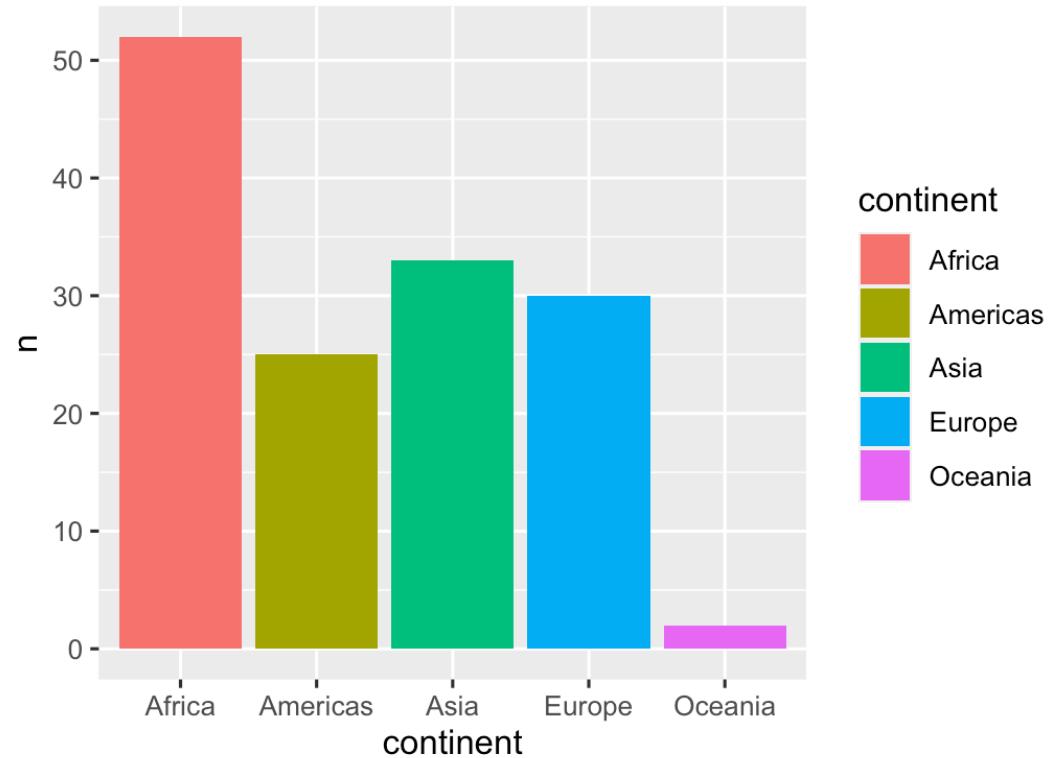
How could we improve this?



# Add some color

```
gapminder_continents |>  
  ggplot(aes(x = continent,  
             y = n,  
             fill = continent)) + # color bars  
  geom_col()
```

Do we need the fill legend?

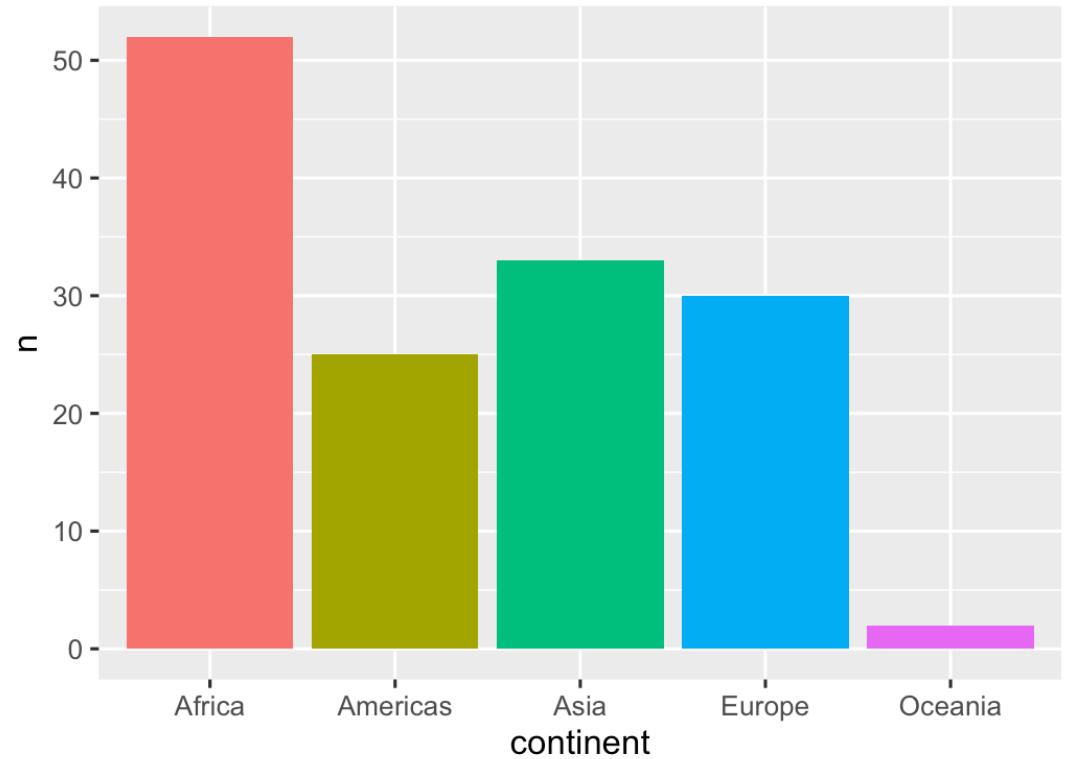


# Add some color

```
gapminder_continents |>  
  ggplot(aes(x = continent, y = n,  
             fill = continent)) +  
  geom_col() +  
  guides(fill = "none") # omit fill legend
```

Is "n" a good axis title?

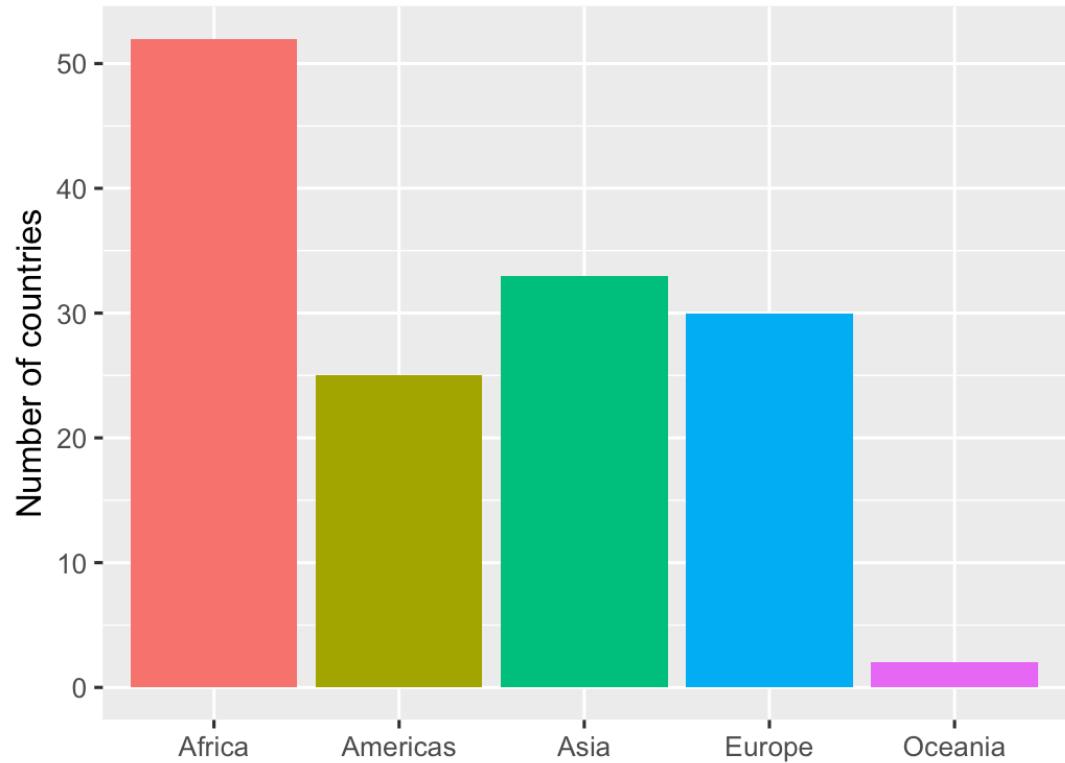
Do we need "continent" at all?



# Add some labels

```
gapminder_continents |>  
  ggplot(aes(x = continent, y = n,  
             fill = continent)) +  
  geom_col() +  
  guides(fill = "none") +  
  labs(x = NULL, y = "Number of countries")
```

Is alphabetical the best ordering?

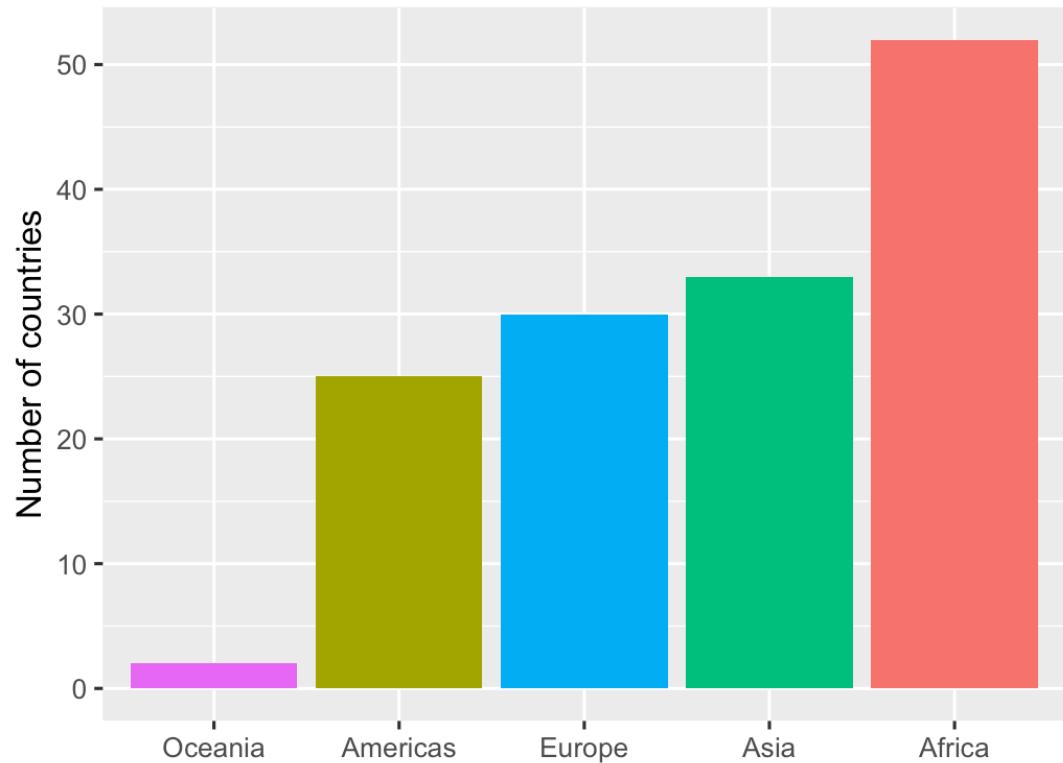


# Order by data value

```
gapminder_continents |>  
  ggplot(aes(x = fct_reorder(continent, n),  
             y = n, fill = continent)) +  
  geom_col() +  
  guides(fill = "none") +  
  labs(x = NULL, y = "Number of countries")
```

`fct_reorder(continent, n)` means  
"reorder the factor variable continent  
by n, smallest to largest"

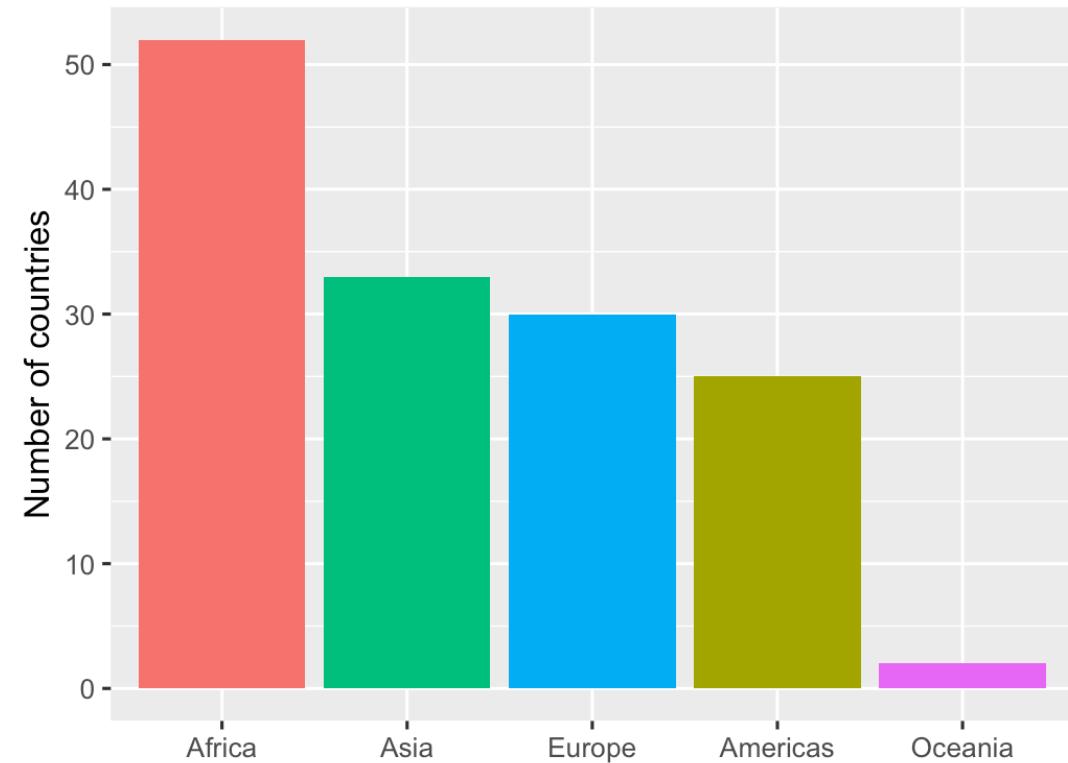
Factor is R's term for variables that  
have a fixed and known set of possible  
values



# Order by data value, descending

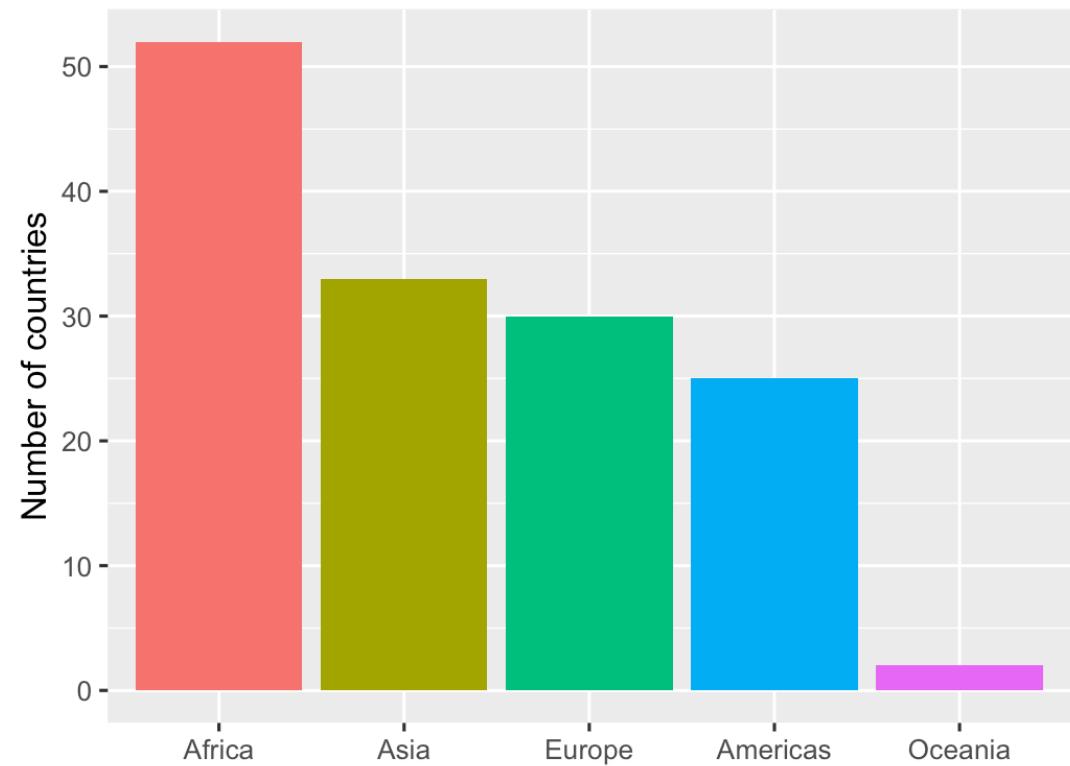
```
gapminder_continents |>  
  ggplot(aes(x = fct_reorder(continent, -n),  
             y = n, fill = continent)) +  
  geom_col() +  
  guides(fill = "none") +  
  labs(x = NULL, y = "Number of countries")
```

`fct_reorder(continent, -n)`  
means "reorder the factor variable  
continent by n, largest to smallest"



# Another option is to encode order in the data

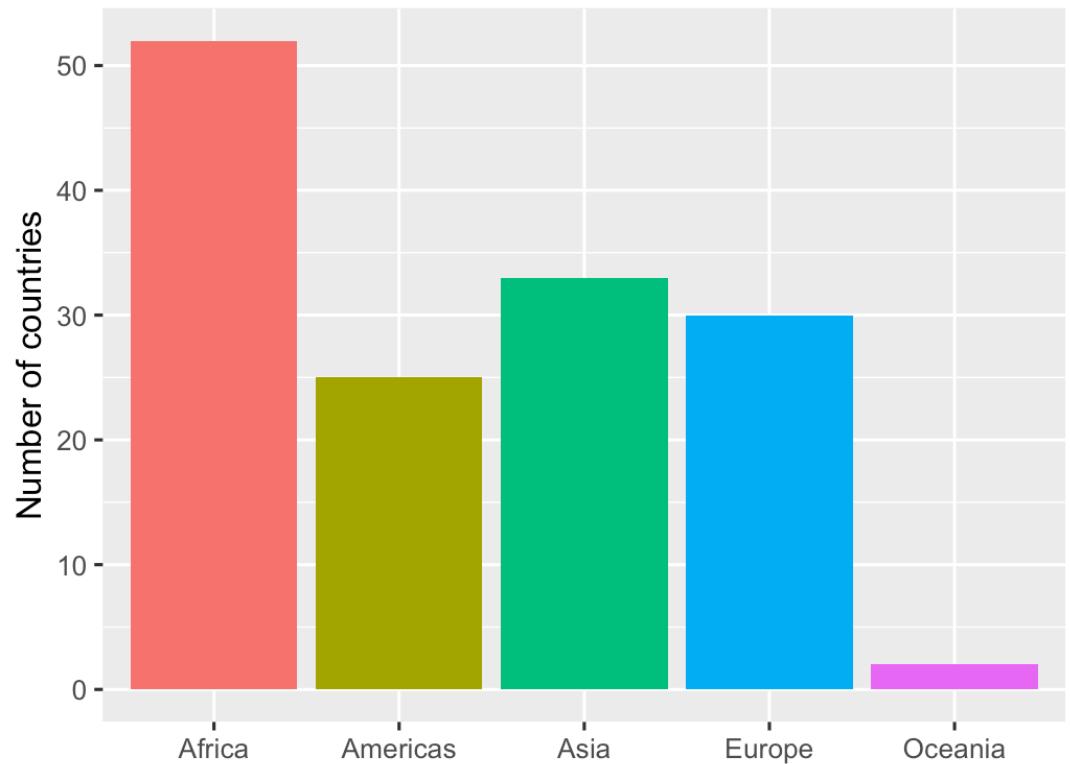
```
# use `fct_inorder` to order levels of continent  
# based on how they appear in the data  
gapminder_ordered <- gapminder_continents |>  
  arrange(desc(n)) |> # sort by count, descending  
  mutate(continent = fct_inorder(continent))  
gapminder_ordered |>  
  ggplot(aes(x = continent,  
             y = n,  
             fill = continent)) +  
  geom_col() +  
  guides(fill = "none") +  
  labs(x = NULL, y = "Number of countries")
```



# Wait, what about geom\_bar?

Use `geom_bar` to count and plot in one step

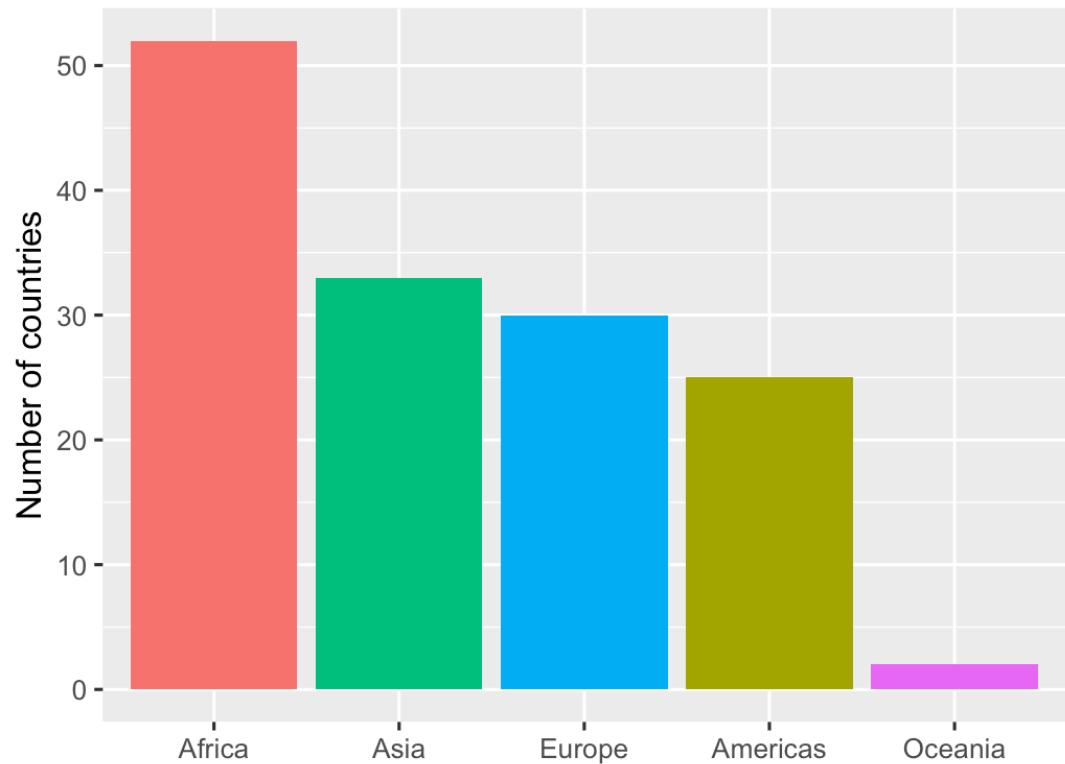
```
gapminder |> # old data: rows are countries
  filter(year == 2007) |>
  ggplot(aes(x = continent, # note: no y aesthe
             fill = continent)) +
  geom_bar() +
  guides(fill = "none") +
  labs(x = NULL, y = "Number of countries")
```



# Wait, what about geom\_bar?

Here we can reorder by frequency using `fct_infreq`

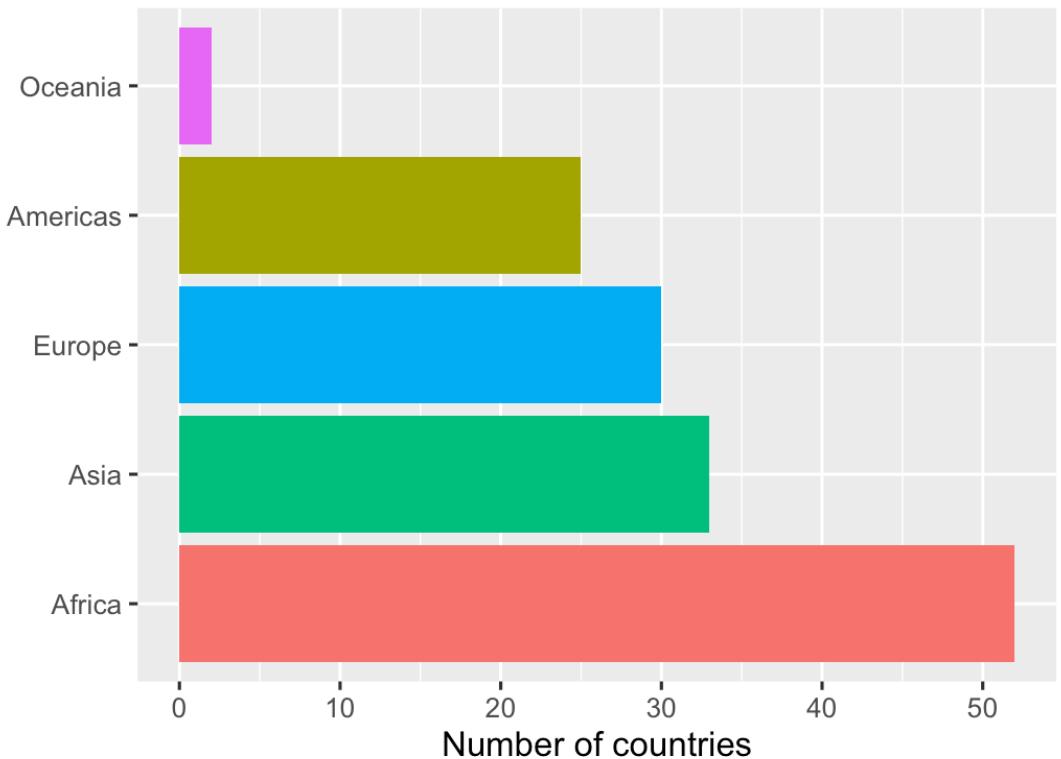
```
gapminder |>
  filter(year == 2007) |>
  ggplot(aes(x = fct_infreq(continent),
             fill = continent)) +
  geom_bar() +
  guides(fill = "none") +
  labs(x = NULL, y = "Number of countries")
```



# We can also flip geom\_col/bar axes

Simply use `y =` instead of `x =` for the aesthetic mapping

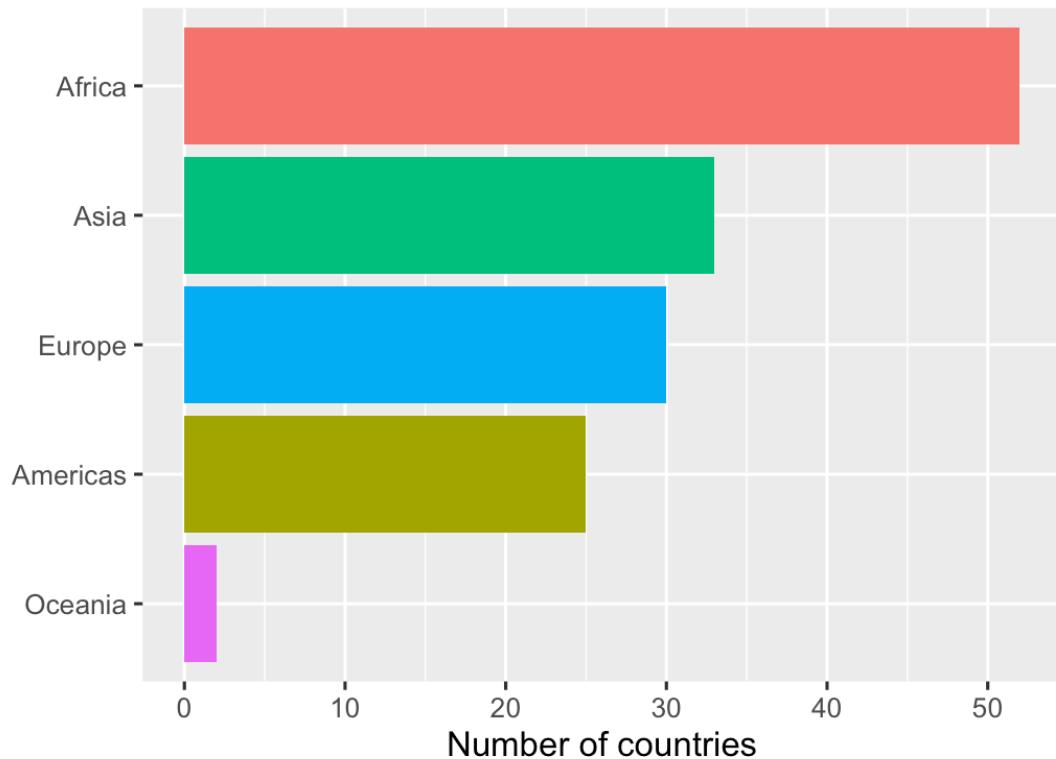
```
gapminder |>
  filter(year == 2007) |>
  ggplot(aes(y = fct_infreq(continent),
             fill = continent)) +
  geom_bar() +
  guides(fill = "none") +
  labs(x = "Number of countries", y = NULL)
```



# We can also flip geom\_col/bar axes

fct\_rev() reverses the order of factors

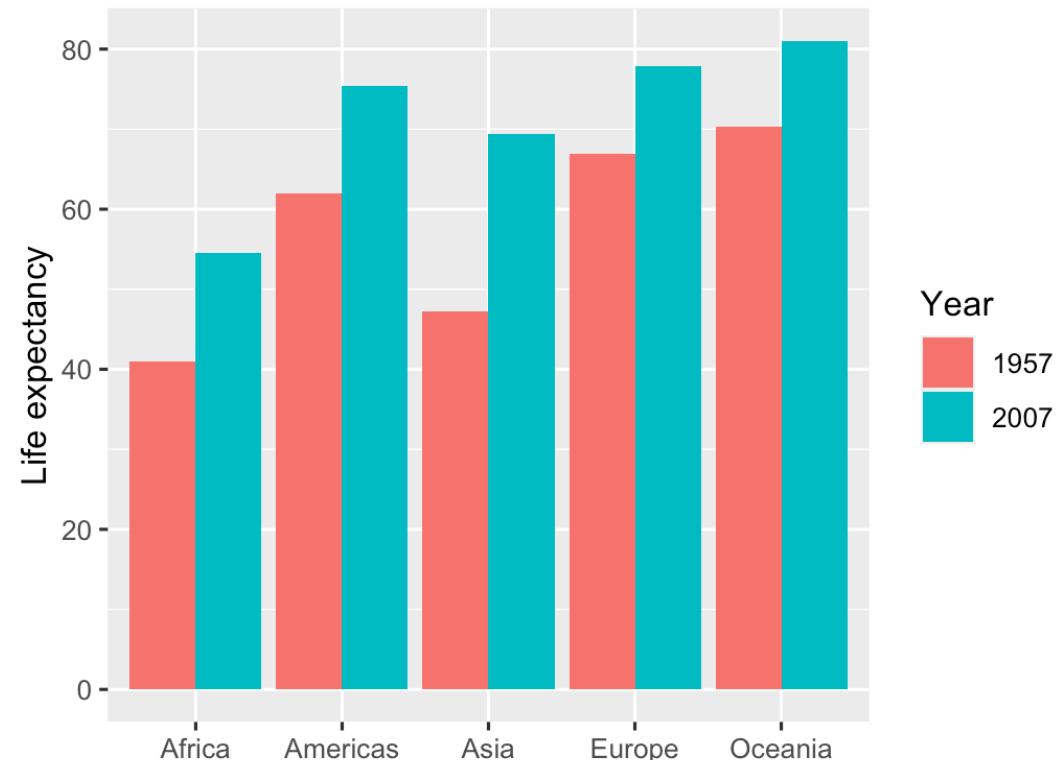
```
gapminder |>
  filter(year == 2007) |>
  ggplot(aes(y = fct_rev(fct_infreq(continent)),
             fill = continent)) +
  geom_bar() +
  guides(fill = "none") +
  labs(x = "Number of countries", y = NULL)
```



# Grouped bar charts

Use grouped bars or facets for higher dimensional datasets

```
gapminder |>
  filter(year==1957 | year==2007) |>
  group_by(continent, year) |>
  summarize(
    lifeExp = sum(lifeExp * pop) / sum(pop)
  ) |>
  ggplot(aes(x = continent,
             y = lifeExp,
             fill = as_factor(year))) +
  geom_col(position = "dodge") +
  labs(x = NULL,
       y = "Life expectancy", fill = "Year")
```

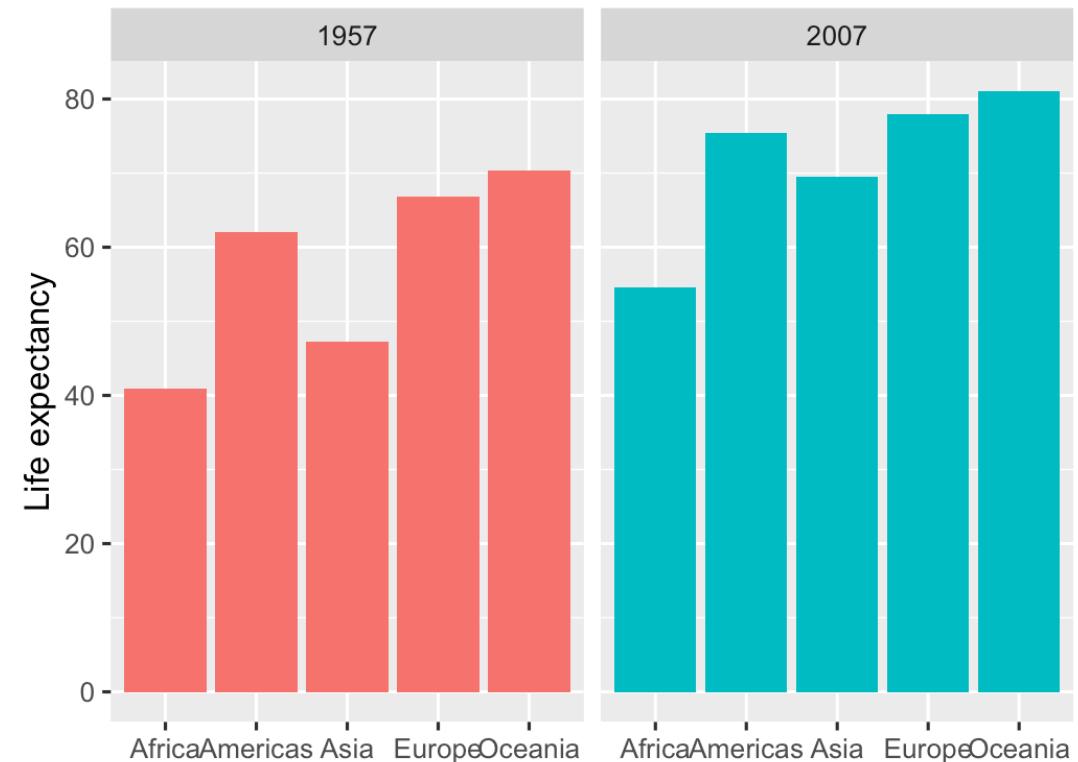


`as_factor()` specifies that we want to treat `year` as distinct categories, not a continuous measure

# Facets

Use grouped bars or facets for higher dimensional datasets

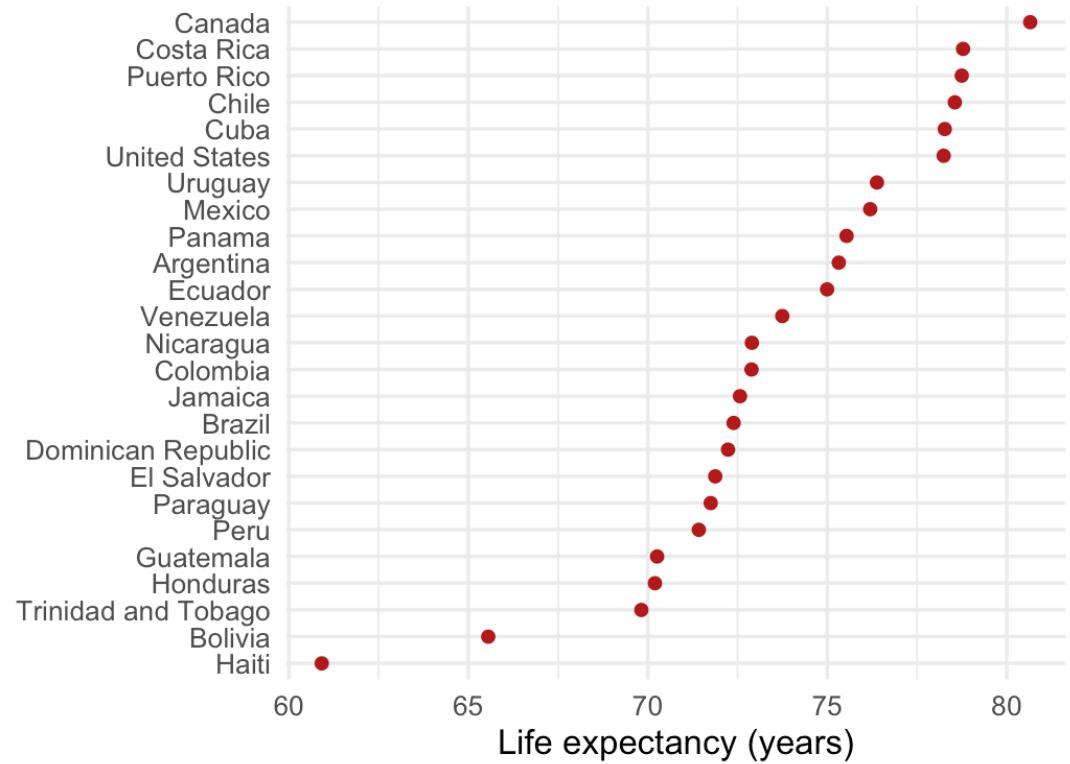
```
gapminder |>
  filter(year==1957 | year==2007) |>
  group_by(continent, year) |>
  summarize(
    lifeExp = sum(lifeExp * pop) / sum(pop)
  ) |>
  ggplot(aes(x = continent,
              y = lifeExp,
              fill = as_factor(year))) +
  geom_col() +
  facet_wrap(vars(year)) +
  guides(fill = "none") +
  labs(x = NULL,
       y = "Life expectancy")
```



# Alternative: Dots instead of bars

Dots are preferable if we want to truncate the axes

```
gapminder |>
  filter(year == 2007, continent == "Americas")
  ggplot(aes(x = lifeExp,
             y = fct_reorder(country, lifeExp)))
  geom_point() +
  guides(color = "none") +
  labs(x = "Life expectancy (years)", y = NULL)
  theme_minimal()
```



example-07-2:  
amounts-practice.R