Mapping data to graphics, amounts, and proportions

Week 7

AEM 2850 / 5850 : R for Business Analytics

Cornell Dyson

Fall 2025

Acknowledgements: Andrew Heiss, Claus Wilke

Announcements

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

No homework this week due to Fall Break 😎

The group project will be due Friday, Nov. 7 at 11:59pm

We will provide details on the group project in the next few weeks

We are grading Prelim 1 and will update you when we can

Questions before we get started?

Plan for this week

Tuesday

- Course progress
- Prologue
- Data, aesthetics,
  & the grammar of graphics
- Amounts
- Plotting amounts using ggplot
- example-07-1
- Reference: More examples

Thursday

- Proportions
- example-07-2
- 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)

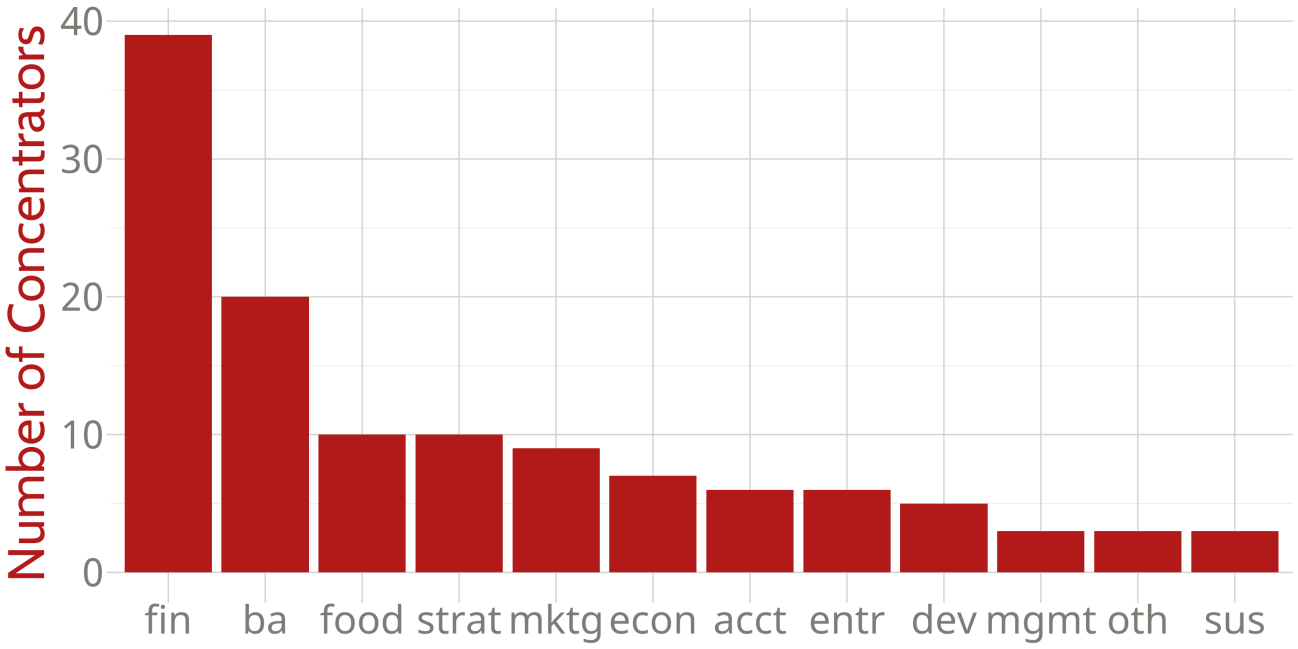
See aem2850.toddgerarden.com/schedule for details

Prologue

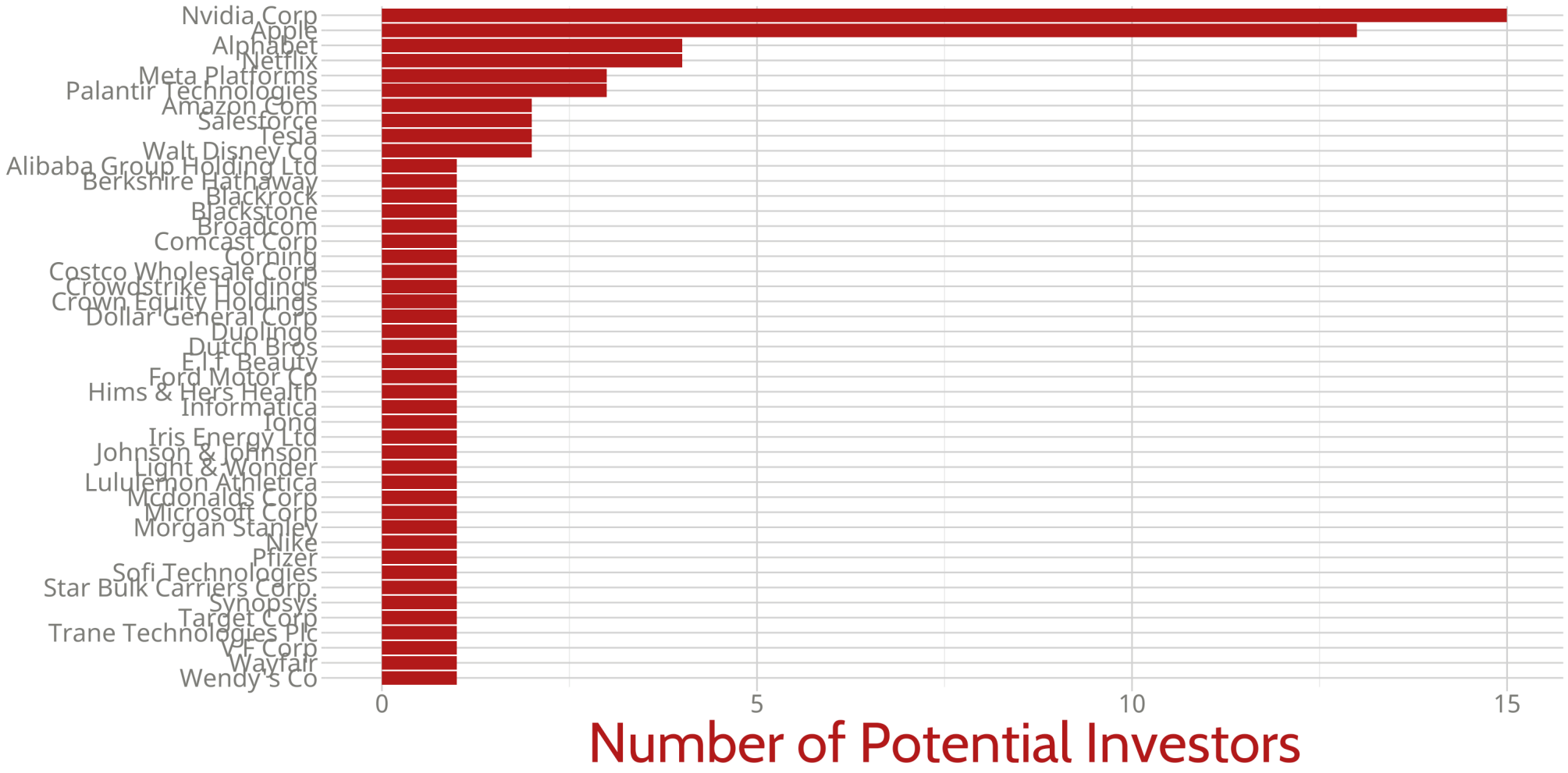
Remember our concentrations?

How might we visualize these amounts?

```
## # A tibble: 12 × 2
##    concentration count
##    <chr>         <int>
##  1 fin              39
##  2 ba               20
##  3 food             10
##  4 strat            10
##  5 mktg              9
##  6 econ              7
##  7 acct              6
##  8 entr              6
##  9 dev               5
## 10 sus               3
## 11 mgmt              3
## 12 oth               3
```

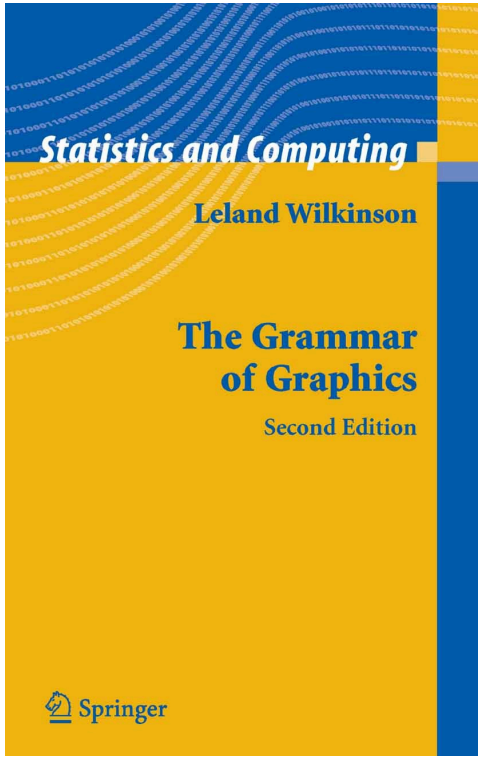


What are our favorite public companies?



Data, aesthetics, & the grammar of graphics

Mapping data to aesthetics



Data

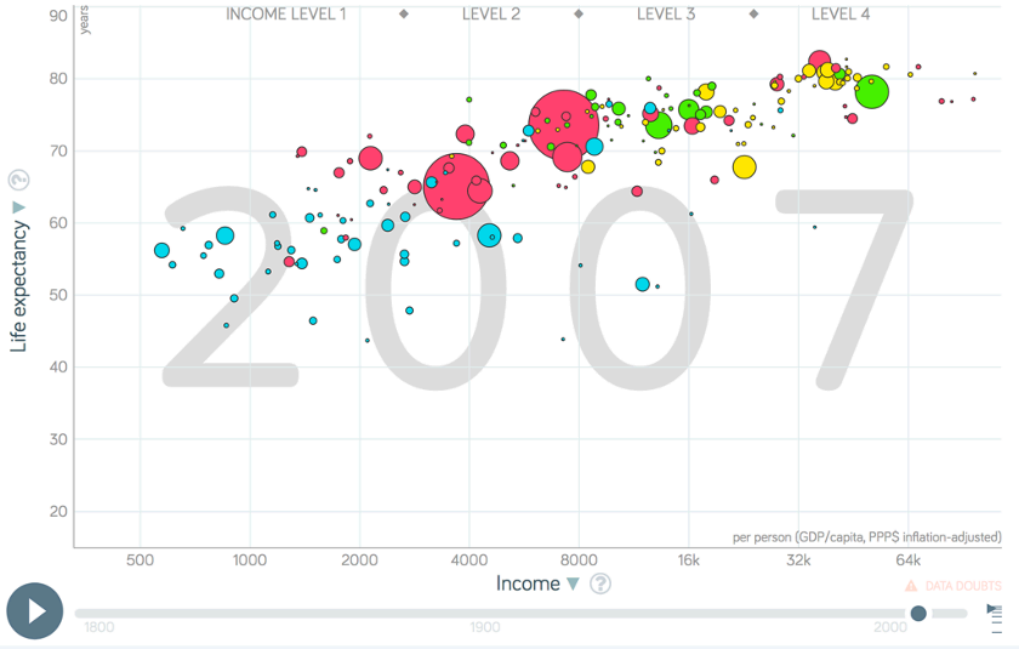
A column in a dataset

Aesthetics

Visual properties of a graph

Position, shape, color, etc.

An example: Health vs wealth



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

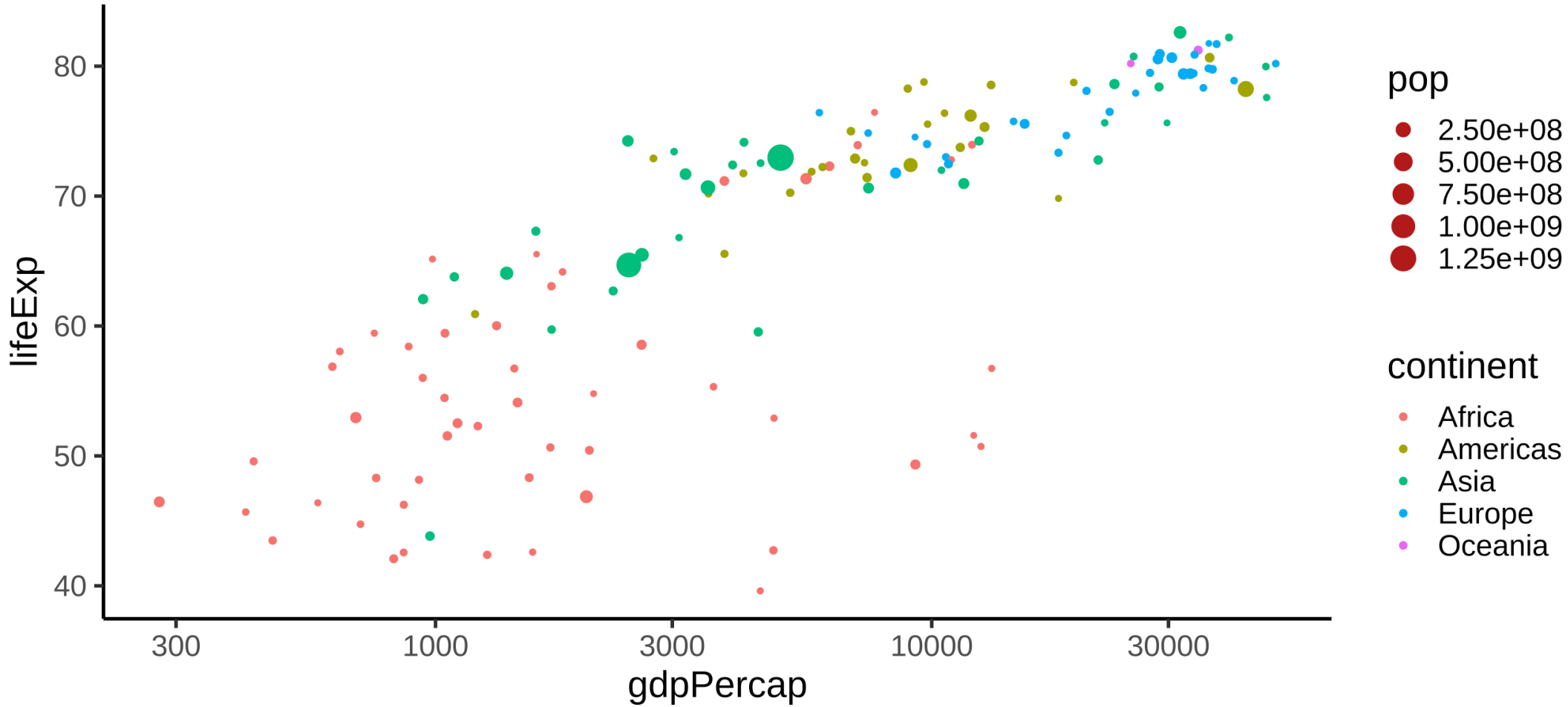
Mapping from `gapminder` to aesthetics

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



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

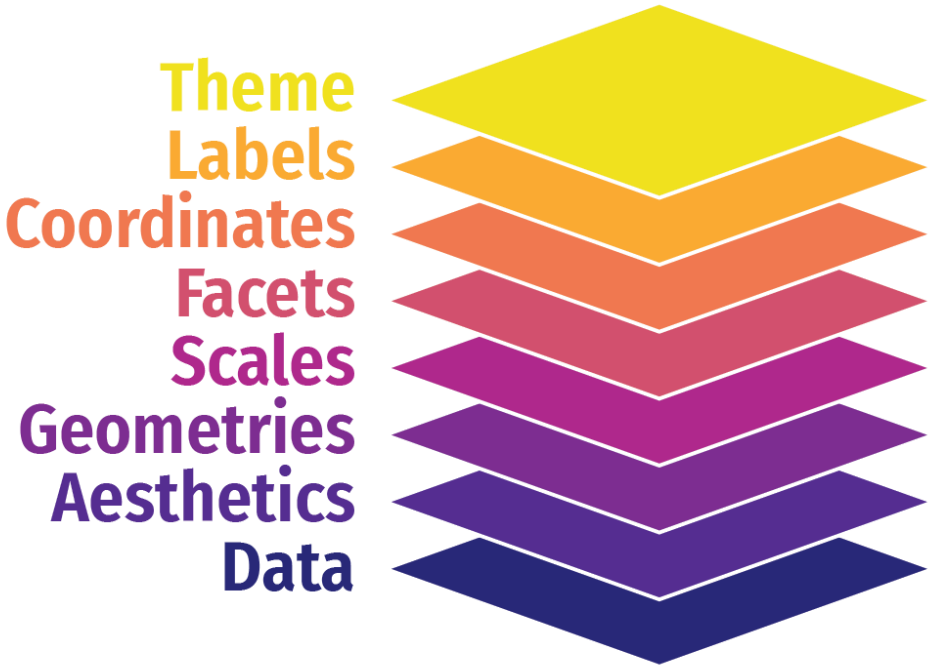
Treat the `+` the same as `|>`

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

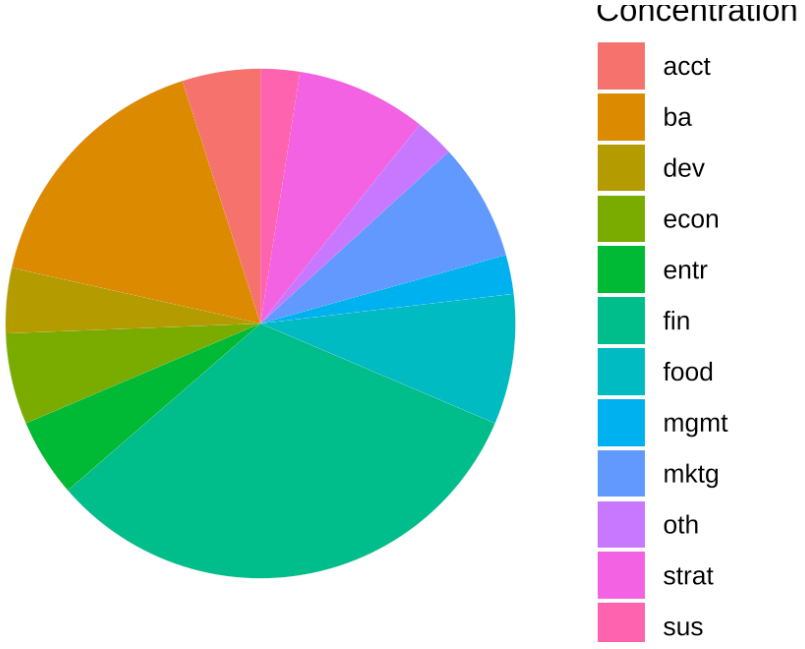
We will primarily learn by doing, though this slide deck contains a preview of additional layers in case you need some bedtime reading

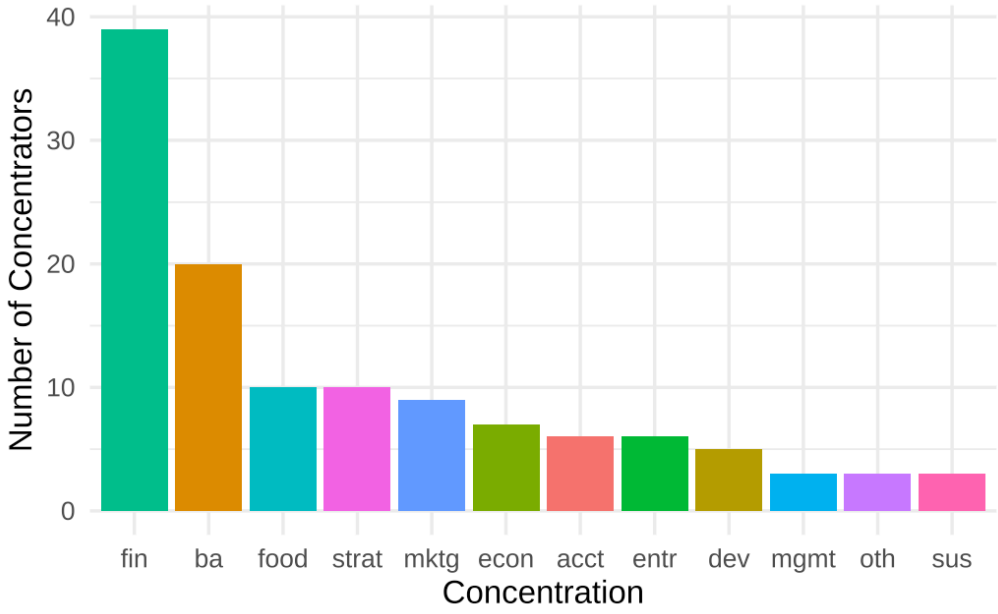


Amounts

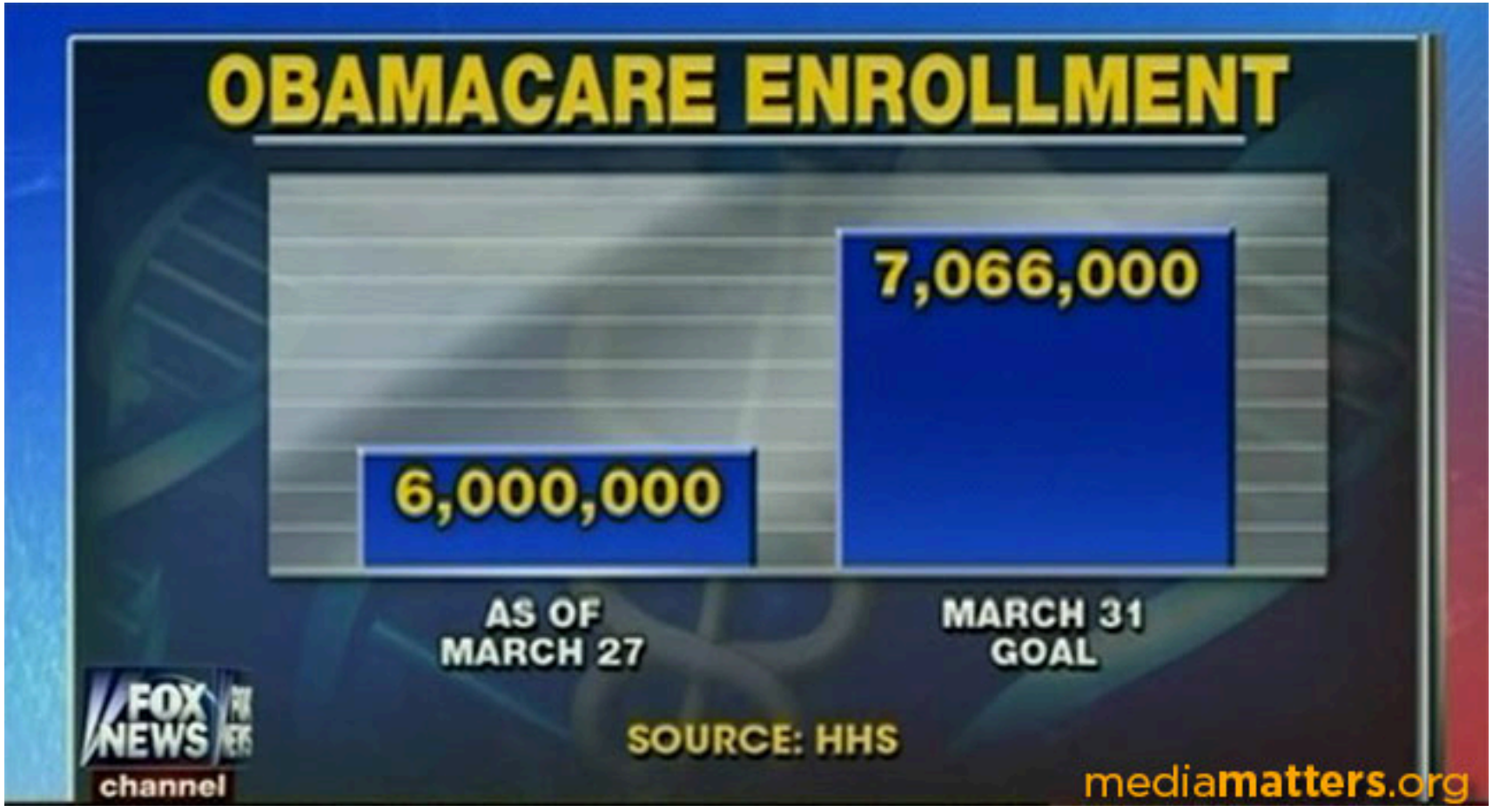
Yay bar plots!

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

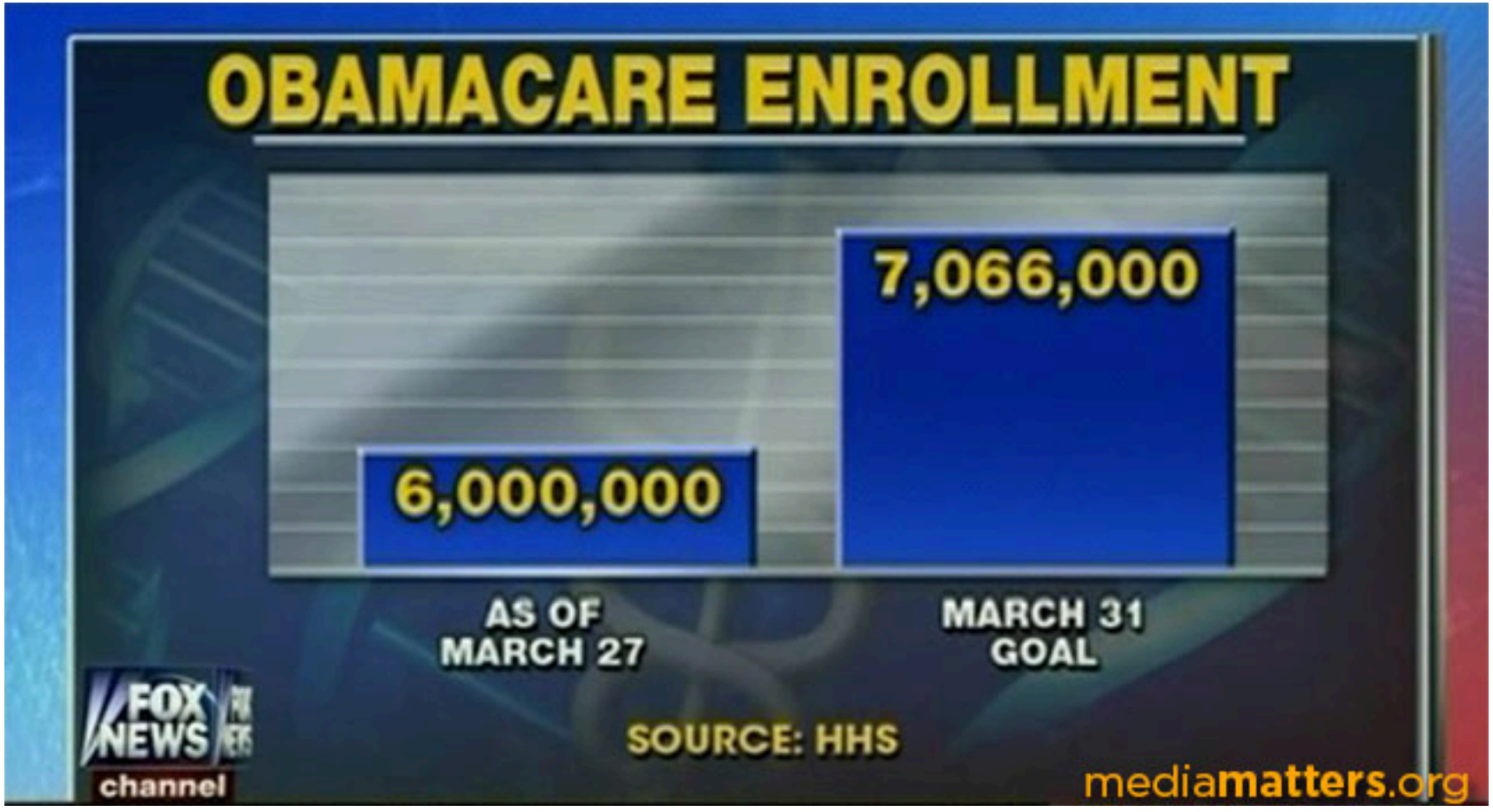




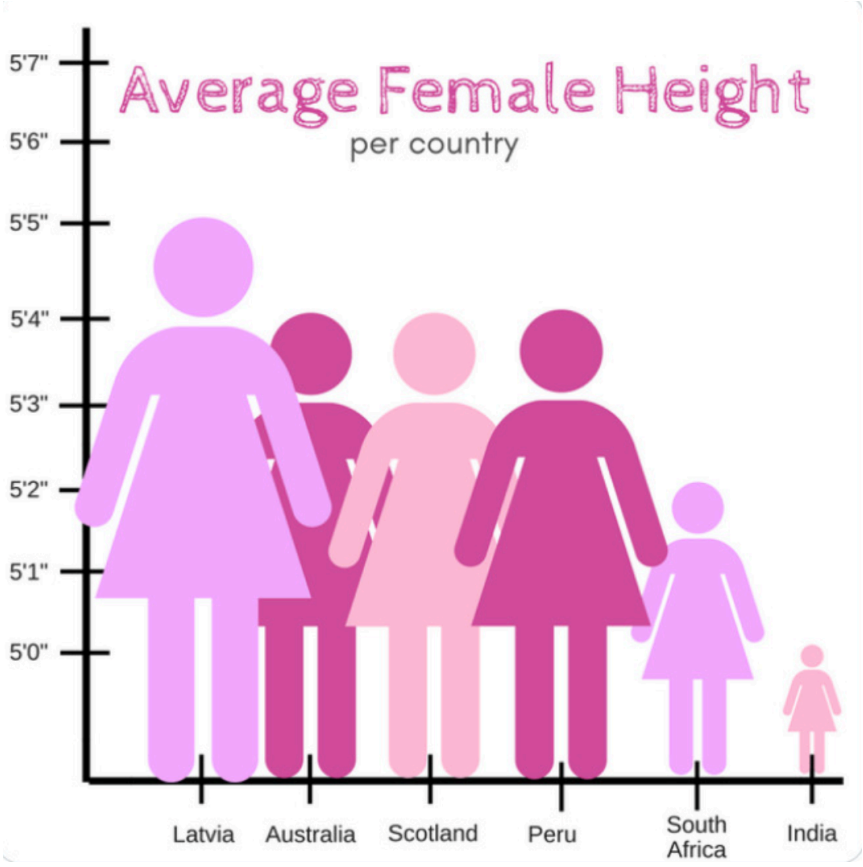
Oh no bar plots!



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!

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

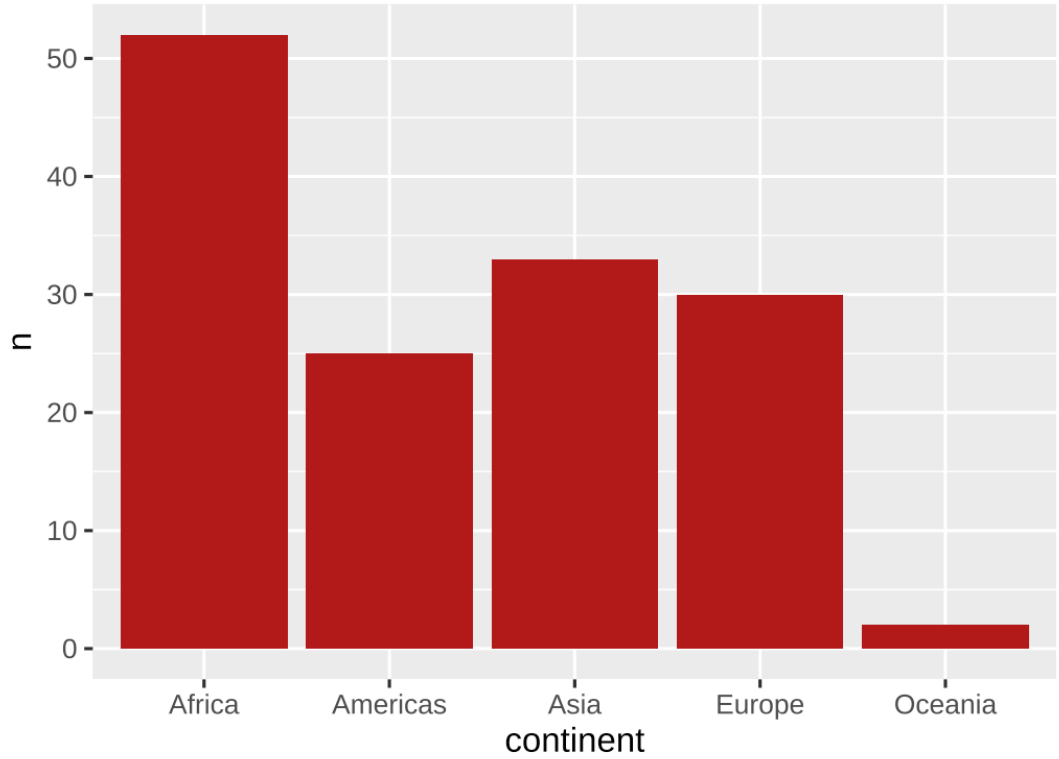
```
library(gapminder)
gapminder_continents <- gapminder |>
  filter(year == 2007) |>  # only look at 2007
  count(continent) |>      # count the number of countries per continent
  arrange(desc(n))         # sort by count, descending
gapminder_continents
```

```
## # 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?

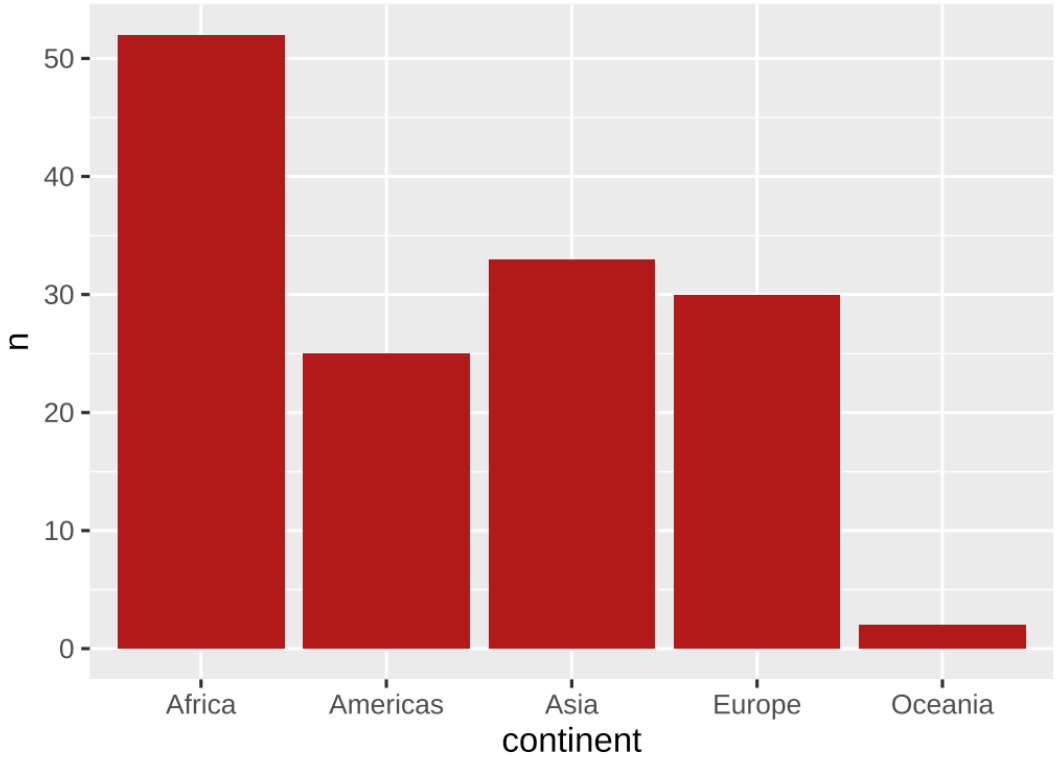


Start with a simple bar plot

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

Is "n" a good axis title?

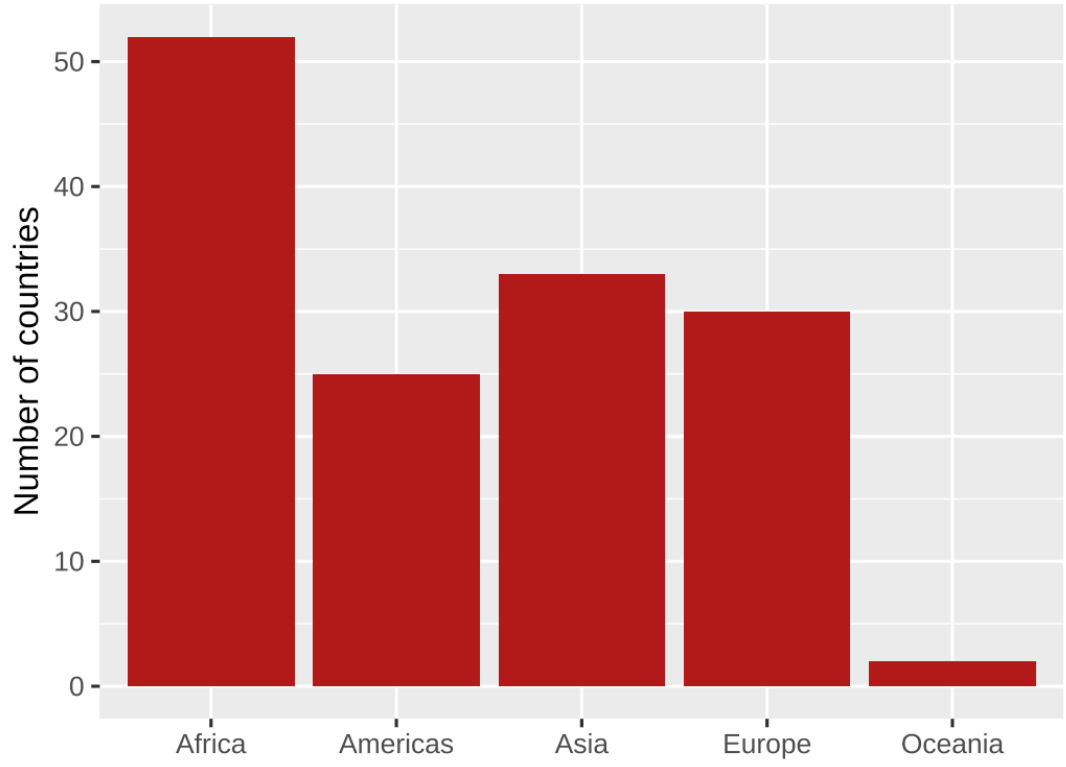
Do we need "continent" at all?



Add some labels

```
gapminder_continents |>
  ggplot(aes(x = continent,
             y = n)) +
  geom_col() +
  labs(x = NULL, y = "Number of countries")
```

Is alphabetical the best ordering?

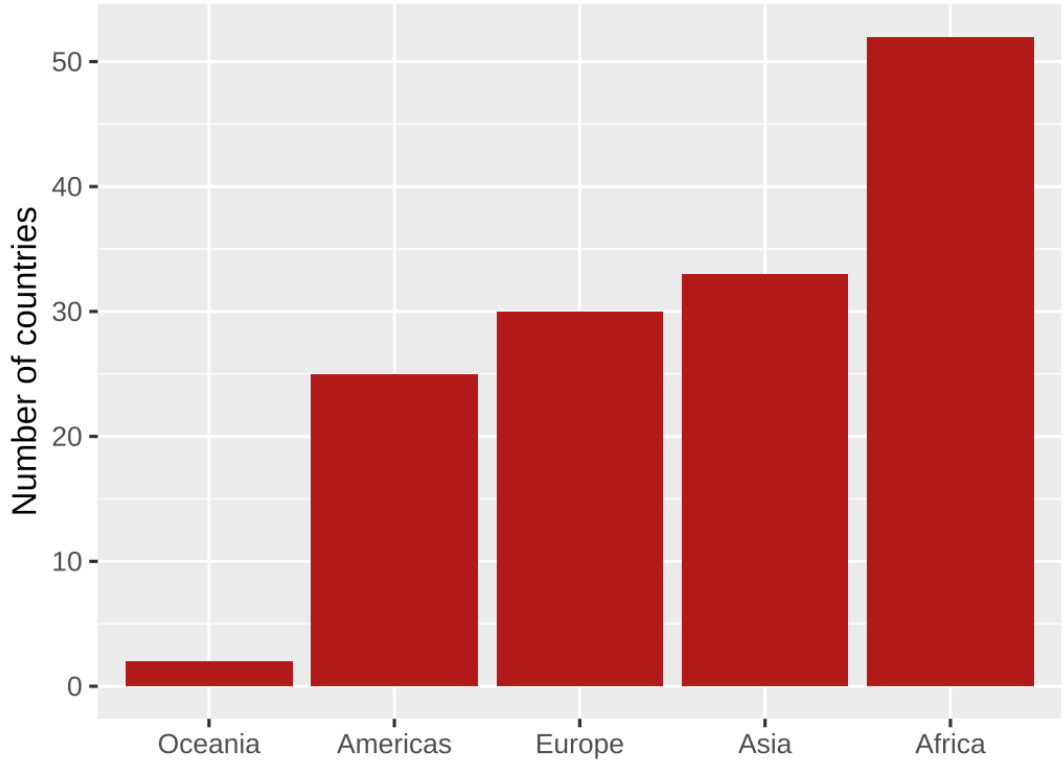


Order by data value

```
gapminder_continents |>
  ggplot(aes(x = fct_reorder(continent, n),
             y = n)) +
  geom_col() +
  labs(x = NULL, y = "Number of countries")
```

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

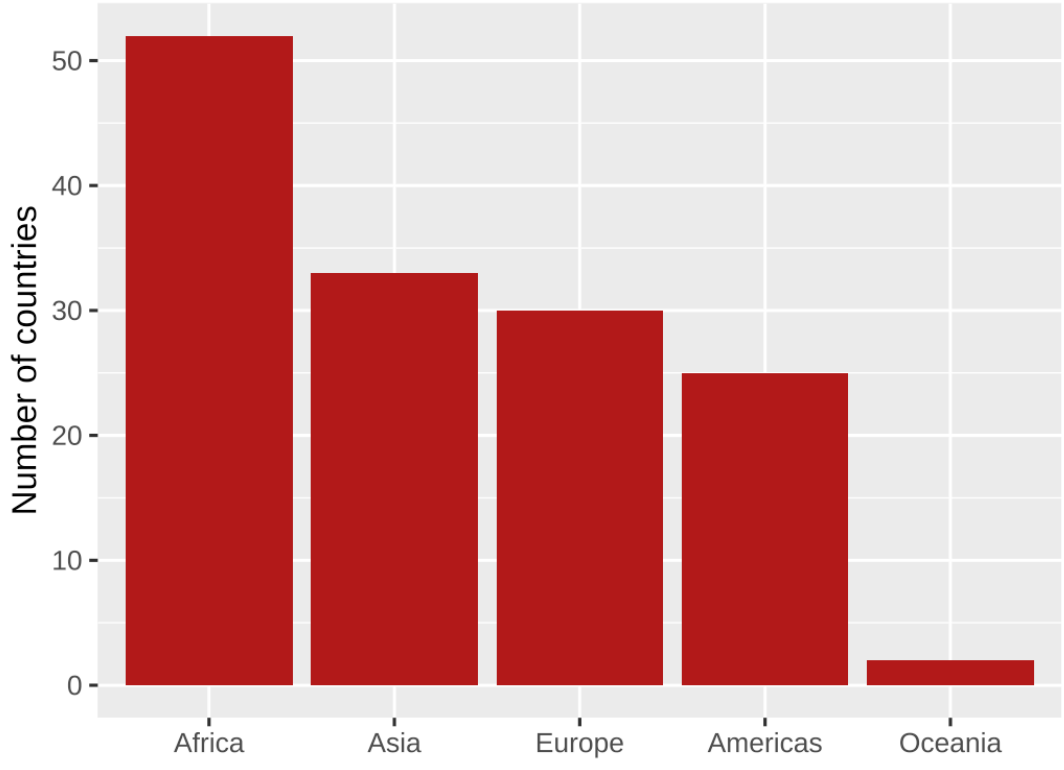
Factor is R's term for variables that have a fixed and known set of possible values



Order by data value, descending

```
gapminder_continents |>
  ggplot(aes(x = fct_reorder(continent, -n),
             y = n)) +
  geom_col() +
  labs(x = NULL, y = "Number of countries")
```

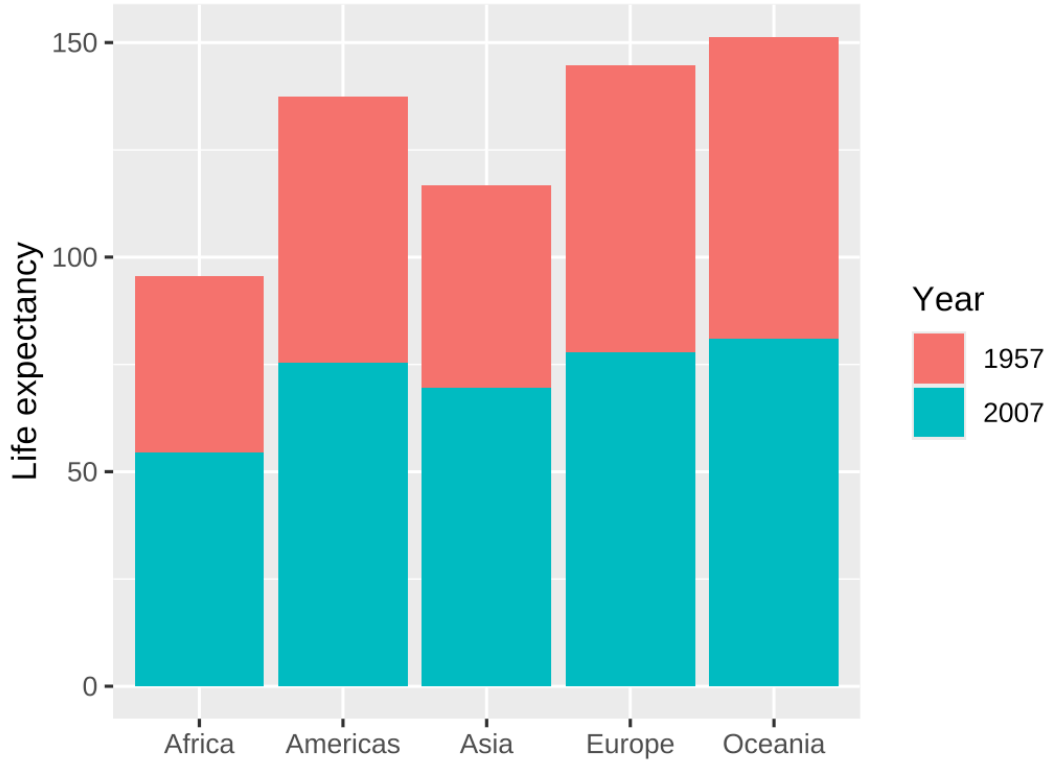
`fct_reorder(continent, -n)` means "reorder the factor variable continent by n, largest to smallest"



Higher dimensional datasets

`as_factor()` treats `year` as distinct categories, not a continuous measure

```
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() +
  labs(x = NULL,
       y = "Life expectancy", fill = "Year")
```

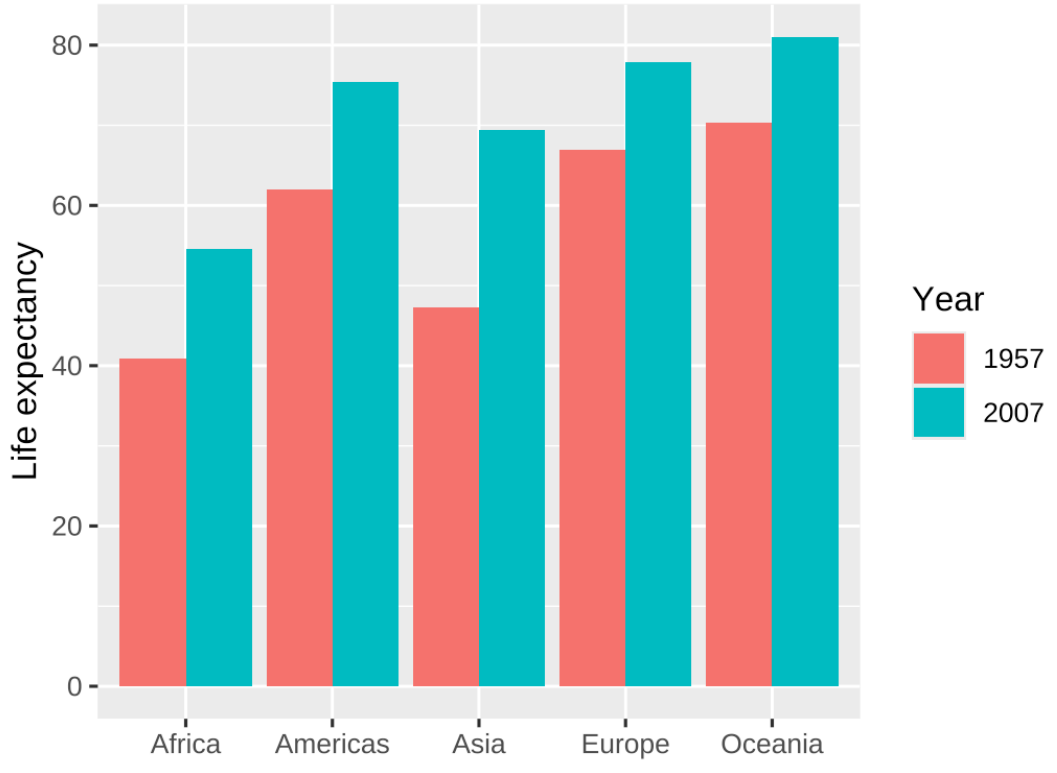


What is wrong with this visualization of life expectancy in two different years?

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")
```



Facets

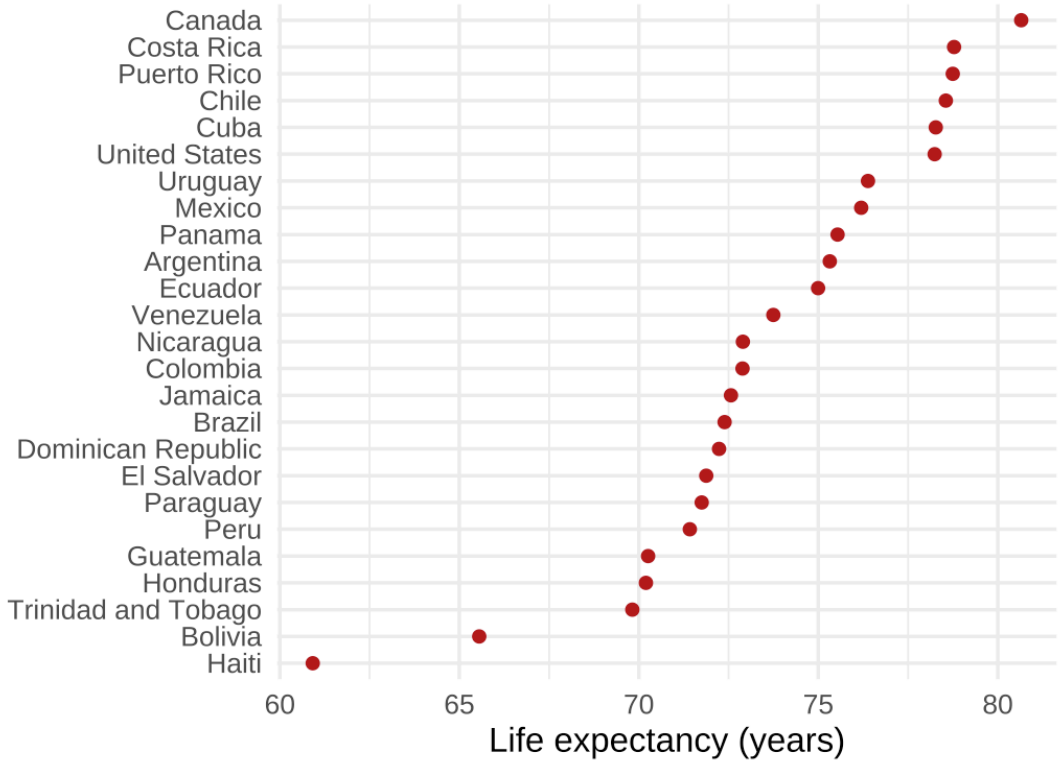
Use grouped bars or facets for higher dimensional datasets

```
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()
```



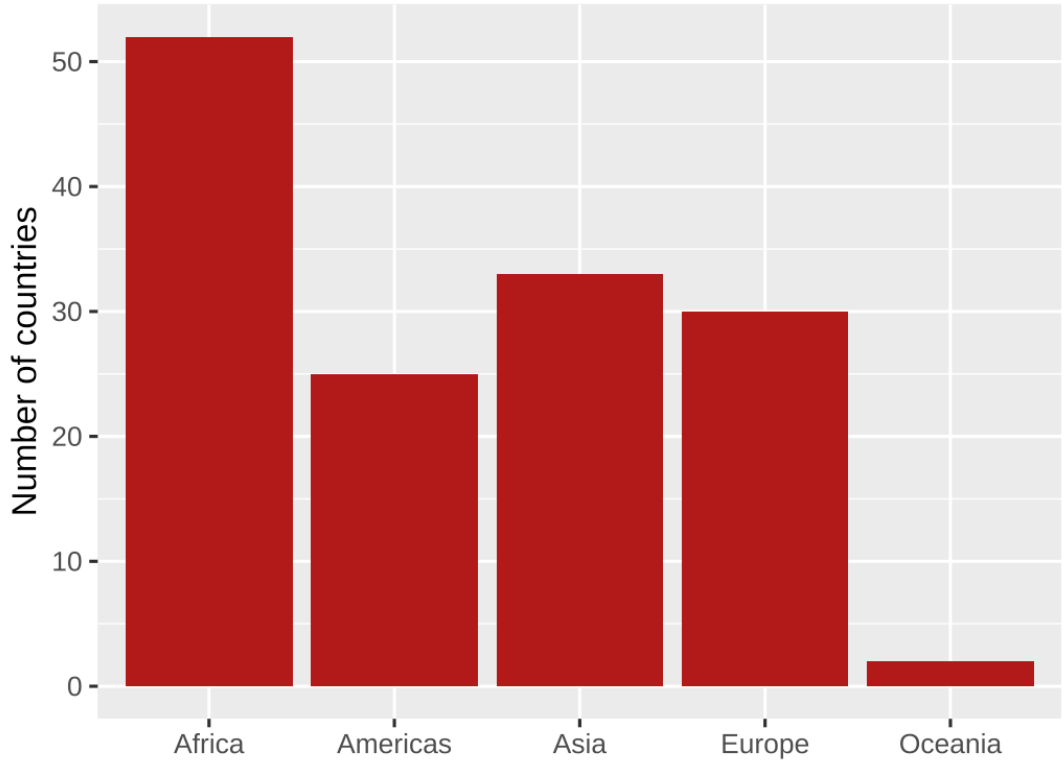
example-07-1: amounts-practice.R

Reference: More examples

Wait, what about geom_bar?

Use `geom_bar` to count and plot in one step

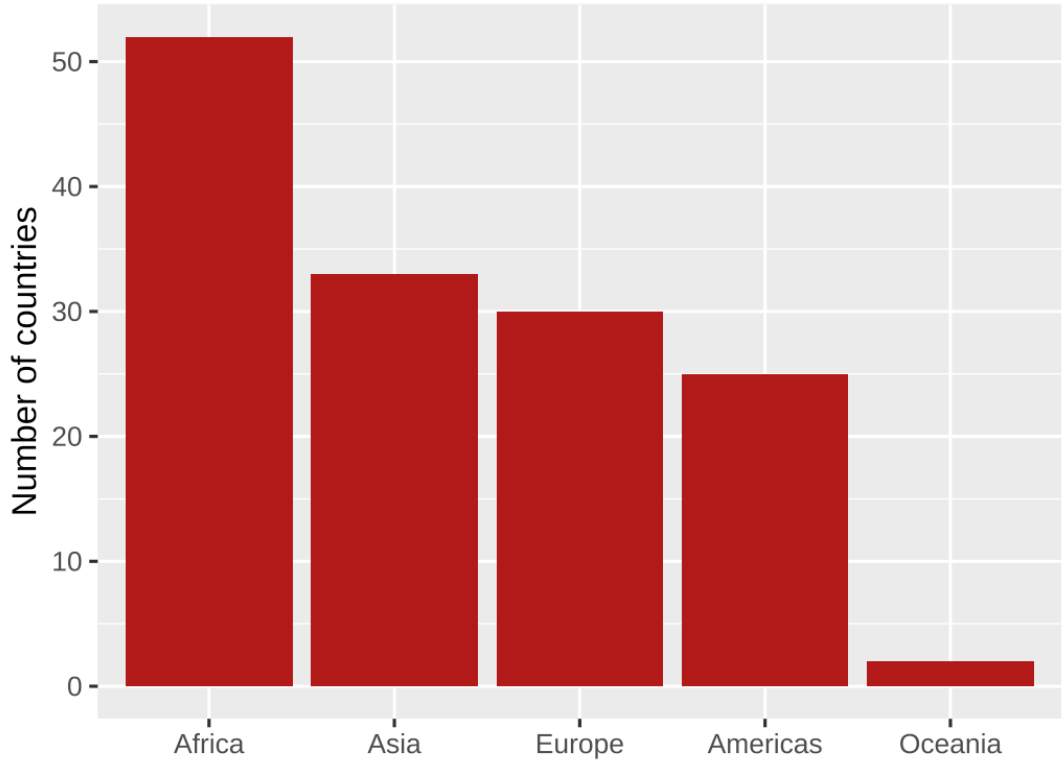
```
gapminder |> # old data: rows are countries
  filter(year == 2007) |>
  ggplot(aes(x = continent)) + # note: no y arg
  geom_bar() +
  labs(x = NULL, y = "Number of countries")
```



Wait, what about geom_bar?

Here we can reorder by frequency using `fct_infreq`

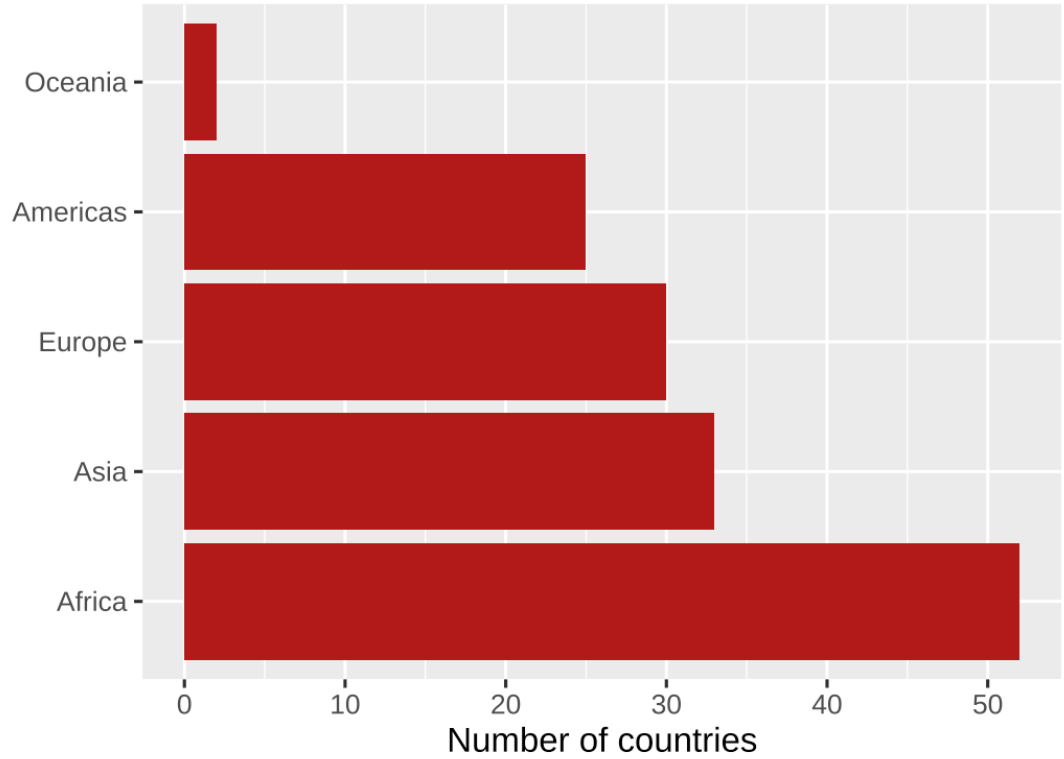
```
gapminder |>
  filter(year == 2007) |>
  ggplot(aes(x = fct_infreq(continent))) +
  geom_bar() +
  labs(x = NULL, y = "Number of countries")
```



We can also flip geom_col/bar axes

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

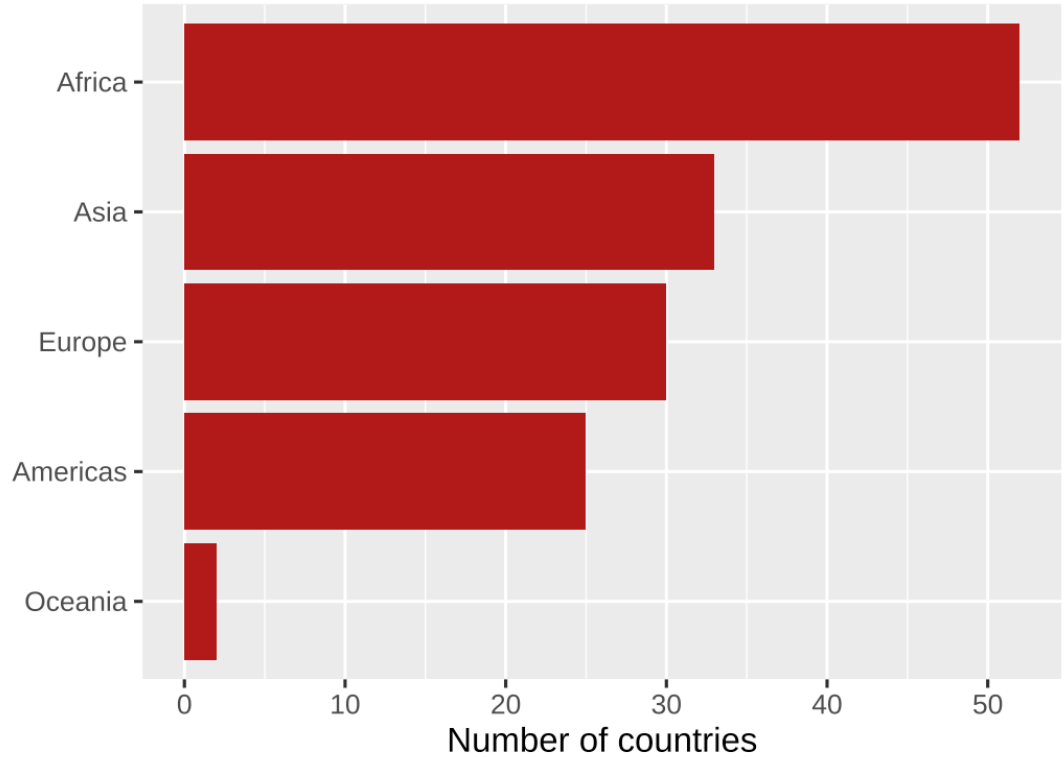
```
gapminder |>
  filter(year == 2007) |>
  ggplot(aes(y = fct_infreq(continent))) +
  geom_bar() +
  labs(x = "Number of countries", y = NULL)
```



We can also flip geom_col/bar axes

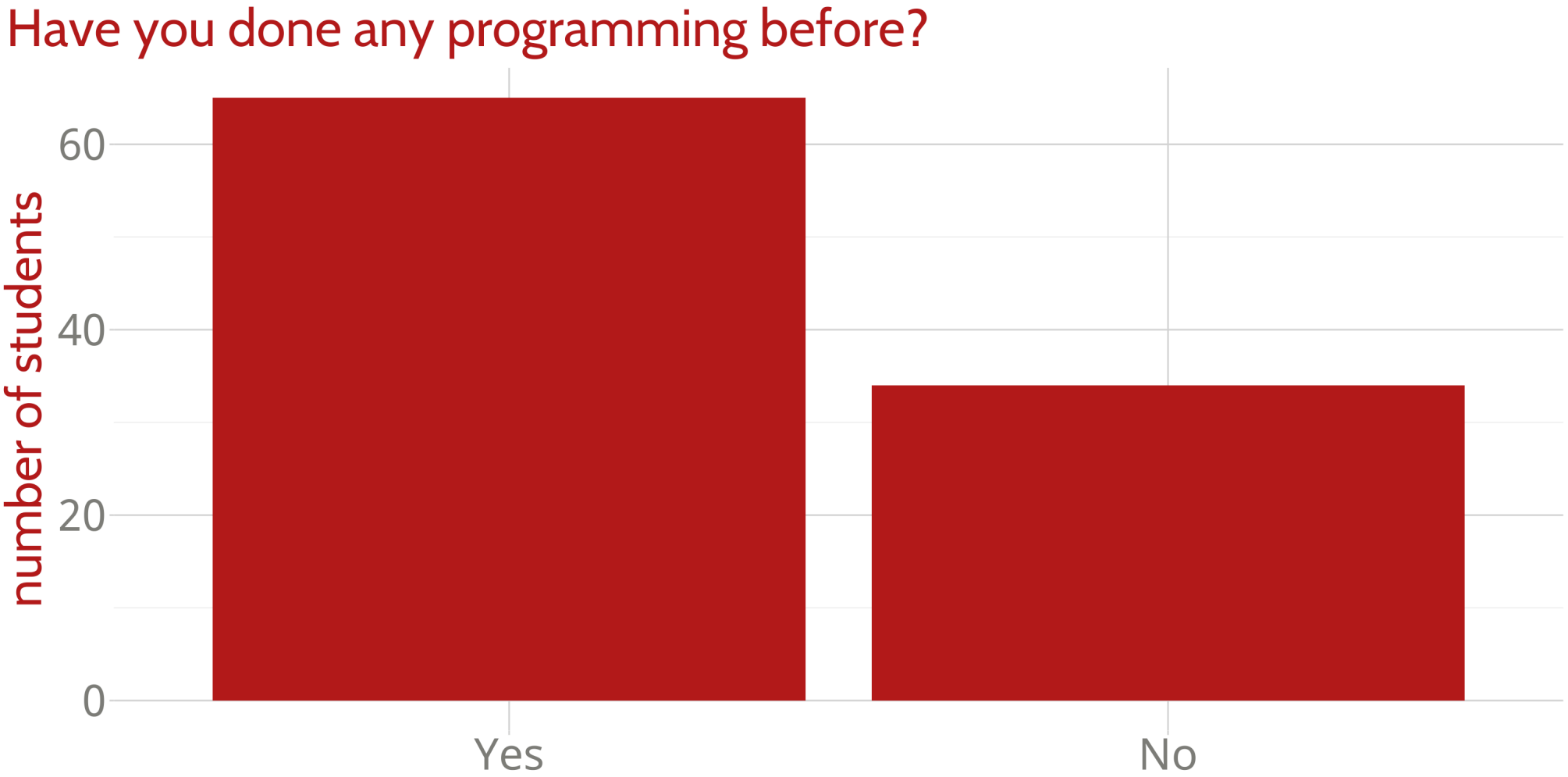
`fct_rev()` reverses the order of factors

```
gapminder |>
  filter(year == 2007) |>
  ggplot(aes(y = fct_rev(fct_infreq(continent)))) +
  geom_bar() +
  labs(x = "Number of countries", y = NULL)
```

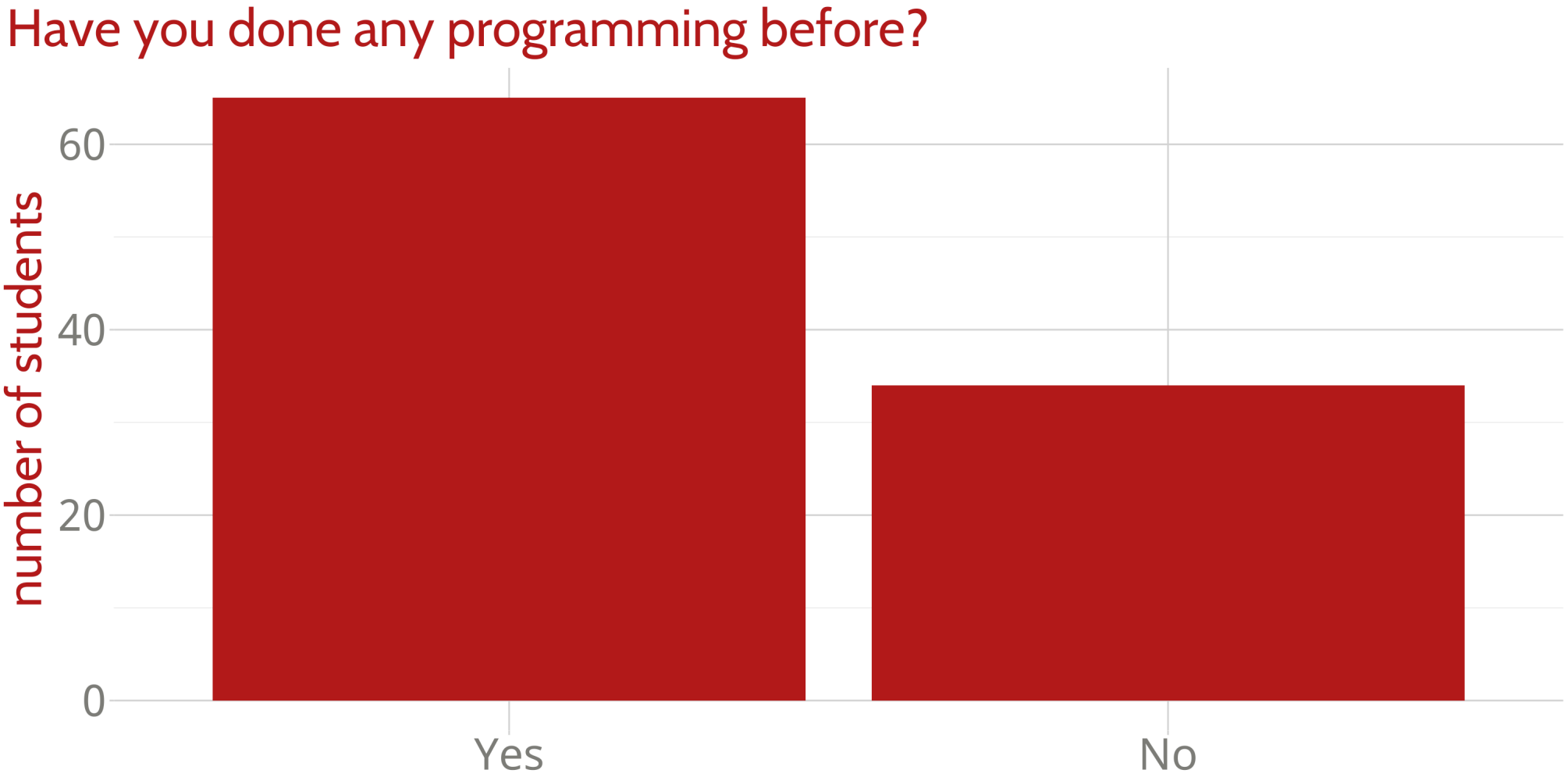


Proportions

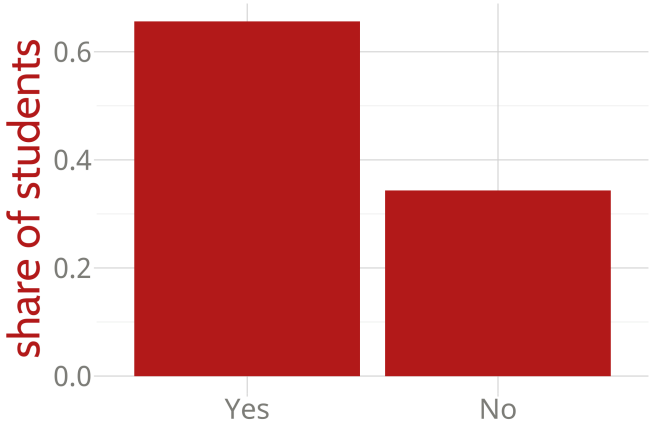
Last week we plotted amounts

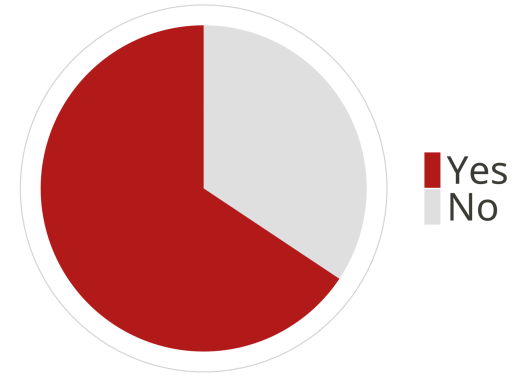


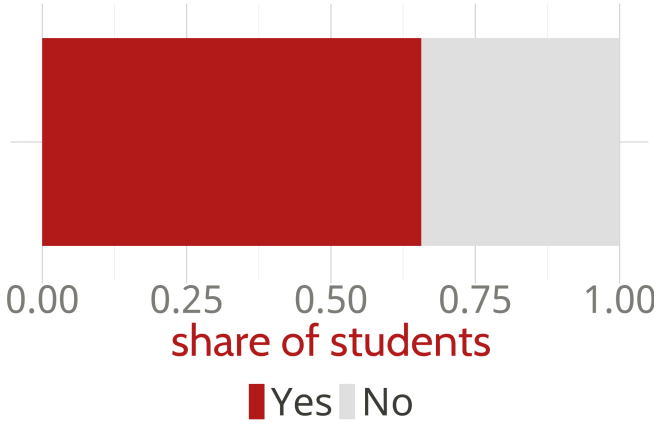
How else could we visualize these data?



Have you done any programming before?







Which do you think is best?

Does it depend on what you want to communicate?

Pros and cons of different approaches

	Pie chart	Stacked bars	Side-by-side bars
Allows easy comparison of relative proportions			

Pros and cons of different approaches

	Pie chart	Stacked bars	Side-by-side bars
Allows easy comparison of relative proportions	✖	✖	✔

Pros and cons of different approaches

	Pie chart	Stacked bars	Side-by-side bars
Allows easy comparison of relative proportions	✖	✖	✔
Shows data as proportions of a whole			

Pros and cons of different approaches

	Pie chart	Stacked bars	Side-by-side bars
Allows easy comparison of relative proportions	✖	✖	✔
Shows data as proportions of a whole	✔	✔	✖

Pros and cons of different approaches

	Pie chart	Stacked bars	Side-by-side bars
Allows easy comparison of relative proportions	✖	✖	✔
Shows data as proportions of a whole	✔	✔	✖
Emphasizes simple fractions (1/2, 1/3, ...)			

Pros and cons of different approaches

	Pie chart	Stacked bars	Side-by-side bars
Allows easy comparison of relative proportions	✖	✖	✔
Shows data as proportions of a whole	✔	✔	✖
Emphasizes simple fractions (1/2, 1/3, ...)	✔	✖	✖

Pros and cons of different approaches

	Pie chart	Stacked bars	Side-by-side bars
Allows easy comparison of relative proportions	✖	✖	✔
Shows data as proportions of a whole	✔	✔	✖
Emphasizes simple fractions (1/2, 1/3, ...)	✔	✖	✖
Visually appealing for small datasets			

Pros and cons of different approaches

	Pie chart	Stacked bars	Side-by-side bars
Allows easy comparison of relative proportions	✖	✖	✔
Shows data as proportions of a whole	✔	✔	✖
Emphasizes simple fractions (1/2, 1/3, ...)	✔	✖	✖
Visually appealing for small datasets	✔	✖	✔

Pros and cons of different approaches

	Pie chart	Stacked bars	Side-by-side bars
Allows easy comparison of relative proportions	✖	✖	✔
Shows data as proportions of a whole	✔	✔	✖
Emphasizes simple fractions (1/2, 1/3, ...)	✔	✖	✖
Visually appealing for small datasets	✔	✖	✔
Works well for a large number of subsets			

Pros and cons of different approaches

	Pie chart	Stacked bars	Side-by-side bars
Allows easy comparison of relative proportions	✖	✖	✔
Shows data as proportions of a whole	✔	✔	✖
Emphasizes simple fractions (1/2, 1/3, ...)	✔	✖	✖
Visually appealing for small datasets	✔	✖	✔
Works well for a large number of subsets	✖	✖	✔

Pros and cons of different approaches

	Pie chart	Stacked bars	Side-by-side bars
Allows easy comparison of relative proportions	✖	✖	✔
Shows data as proportions of a whole	✔	✔	✖
Emphasizes simple fractions (1/2, 1/3, ...)	✔	✖	✖
Visually appealing for small datasets	✔	✖	✔
Works well for a large number of subsets	✖	✖	✔
Works well for time series and similar			

Pros and cons of different approaches

	Pie chart	Stacked bars	Side-by-side bars
Allows easy comparison of relative proportions	✖	✖	✔
Shows data as proportions of a whole	✔	✔	✖
Emphasizes simple fractions (1/2, 1/3, ...)	✔	✖	✖
Visually appealing for small datasets	✔	✖	✔
Works well for a large number of subsets	✖	✖	✔
Works well for time series and similar	✖	✔	✖

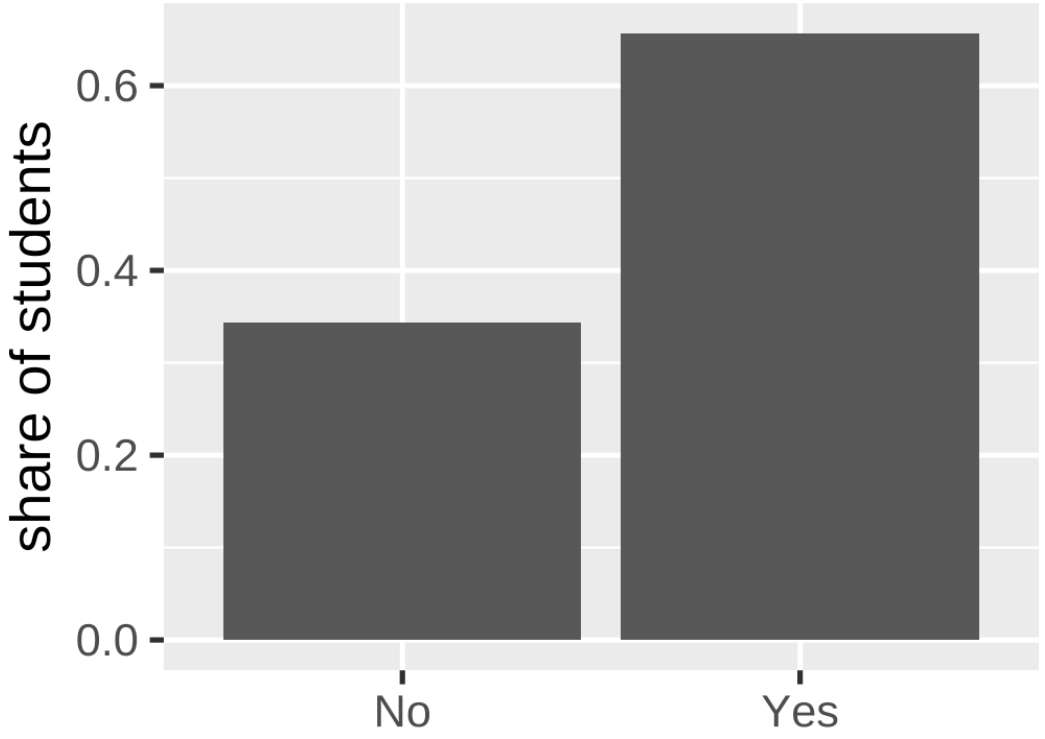
No one visualization fits all scenarios!

Side-by-side bars using ggplot

How could we use ggplot to visualize *proportions* using side-by-side bars?

We could do it manually:

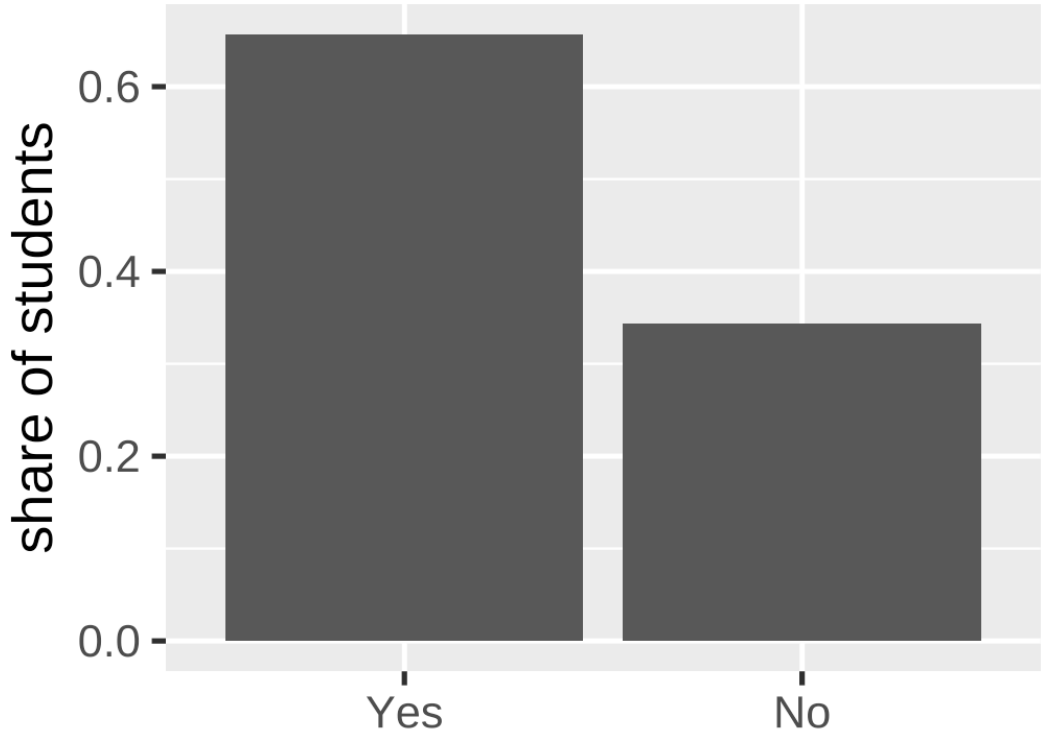
```
prior_programming |>
    count(prior_programming) |>
    mutate(share = n / sum(n)) |>
    ggplot(aes(
      x = prior_programming,
      y = share
      )) +
    geom_col() +
    labs(x = NULL,
         y = "share of students")
```



How could we reverse the bars' order?

Side-by-side bars using ggplot

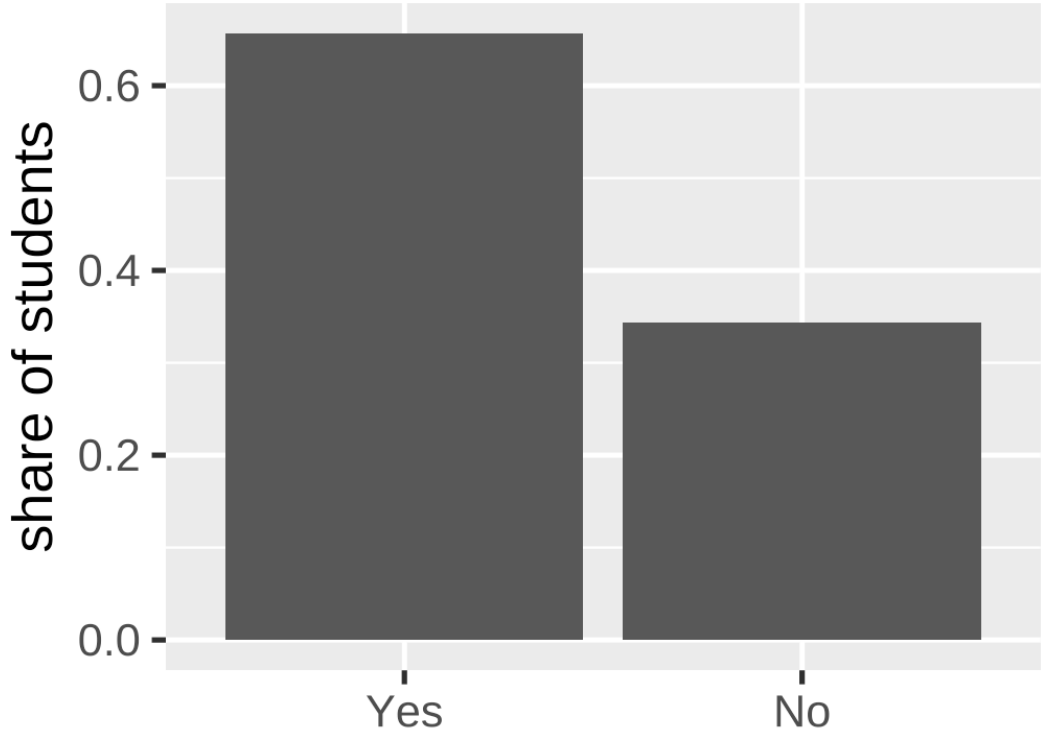
```
prior_programming |>
    count(prior_programming) |>
    mutate(share = n / sum(n)) |>
    ggplot(aes(
      x = fct_reorder(
        prior_programming,
        -share
        ),
      y = share
      )) +
    geom_col() +
    labs(x = NULL,
         y = "share of students")
```



Side-by-side bars using ggplot

`fct_rev()` also works well since there are only two categories:

```
prior_programming |>
    count(prior_programming) |>
    mutate(share = n / sum(n)) |>
    ggplot(aes(
      x = fct_rev(prior_programming),
      y = share
      )) +
    geom_col() +
    labs(x = NULL,
         y = "share of students")
```

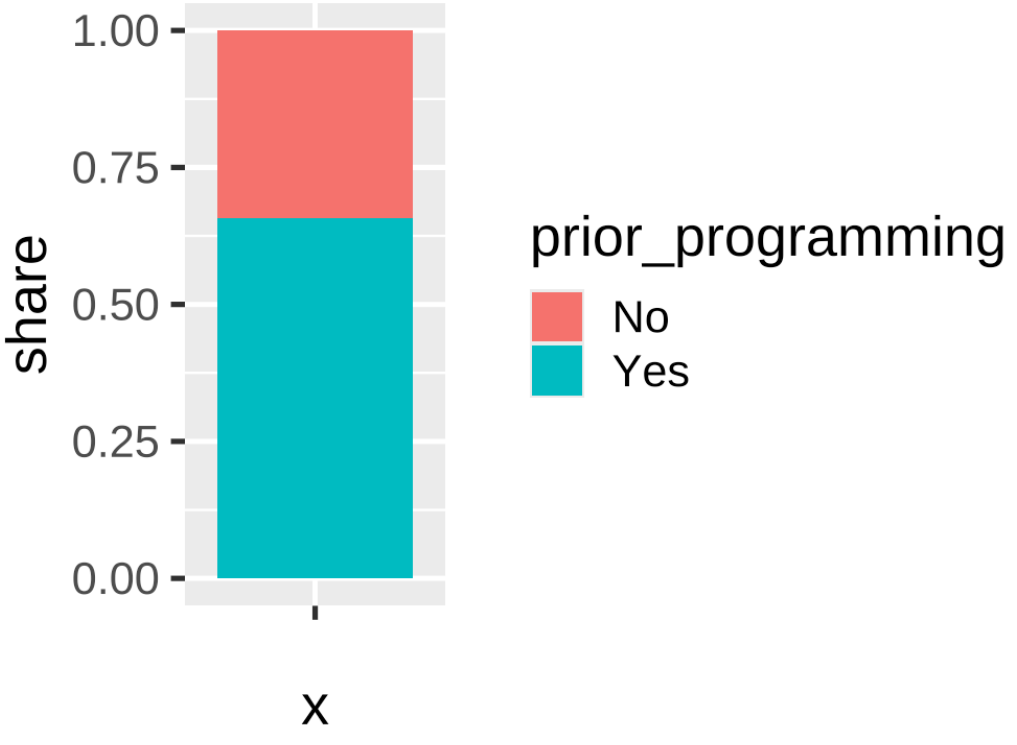


Stacked bars using ggplot

How could we use ggplot to visualize *proportions* using stacked bars?

Again, we could do it manually:

```
prior_programming |>
    count(prior_programming) |>
    mutate(share = n / sum(n)) |>
    ggplot(aes(
        x = "",     # provide dummy to x
        y = share, # plot shares on y
        fill = prior_programming
    )) +
    geom_col()
```



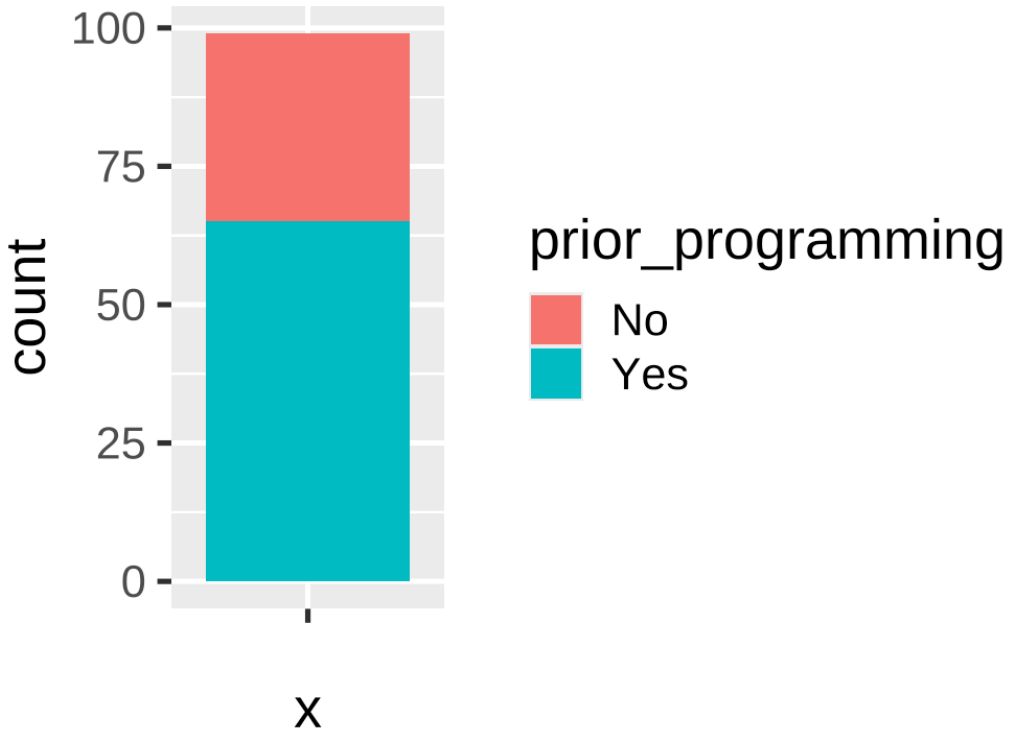
By default, `geom_col` stacks bars if they fall in the same place (`x`)

Stacked bars using ggplot

Alternatively, we could use `geom_bar()` to count and plot the data for us

```
prior_programming |>
  ggplot(aes(
    x = "",
    fill = prior_programming
  )) +
  geom_bar()
```

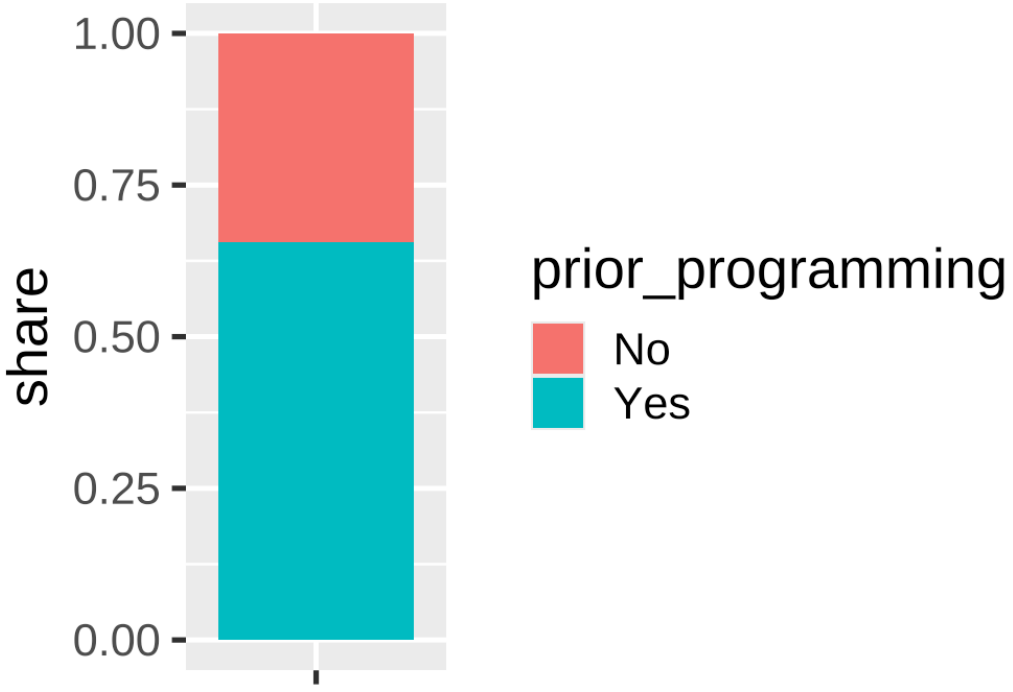
But this gives us *counts*. We want *shares*!



Stacked bars using ggplot

The argument `position = "fill"` scales everything to sum to 1

```
prior_programming |>
  ggplot(aes(
    x = "",
    fill = prior_programming
  )) +
  geom_bar(position = "fill") +
  labs(x = NULL, y = "share")
```

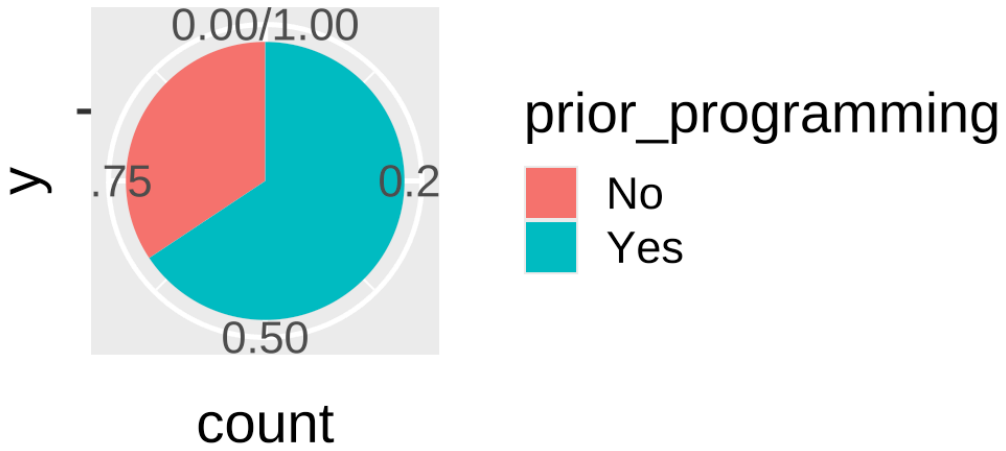


Pie charts using ggplot

How could we use ggplot to visualize *proportions* using a pie chart?

Pie charts are just stacked bars in polar coordinates

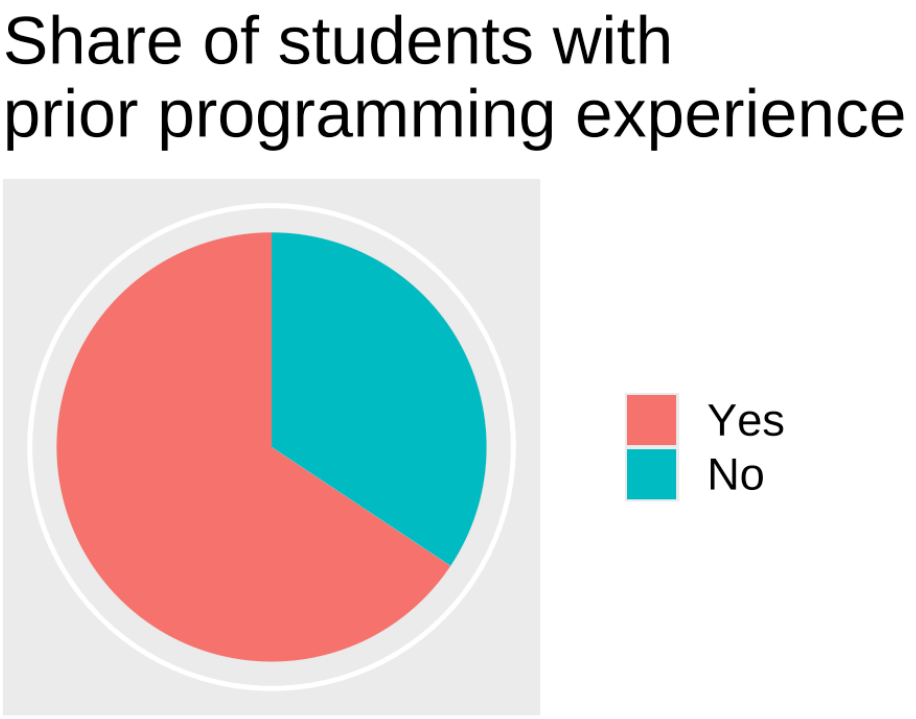
```
prior_programming |>
  ggplot(aes(
    y = "", # y, not x
    fill = prior_programming
  )) +
  geom_bar(position = "fill") +
  coord_polar() # convert to polar coordinates
```



Pie charts using ggplot

It takes some work to create a clean pie chart using ggplot

```
prior_programming |>
  ggplot(aes(
    y = "",
    fill = fct_rev(prior_programming)
  )) +
  geom_bar(position = "fill") +
  coord_polar() +
  scale_x_continuous(
    name = NULL, breaks = NULL
  ) +
  scale_y_discrete(
    name = NULL, breaks = NULL
  ) +
  labs(
    title = "Share of students with\nprior prog
    fill = NULL
  )
```



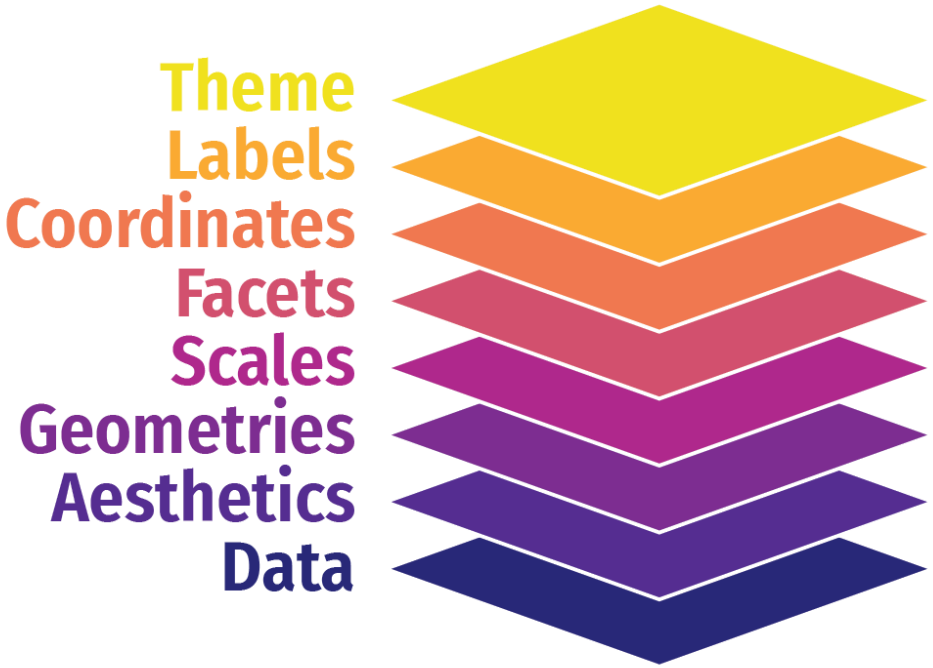
example-07-2: proportions-practice.R

Reference: Additional layers

Additional layers

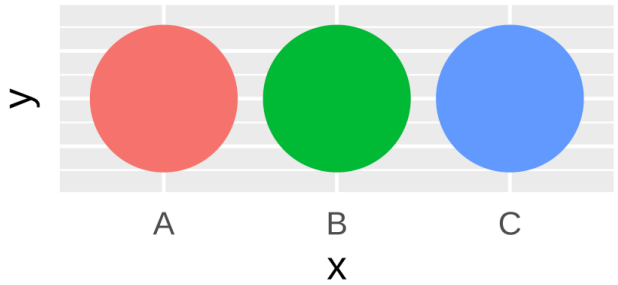
The next several slides contains a preview of additional layers

These are intended as a reference

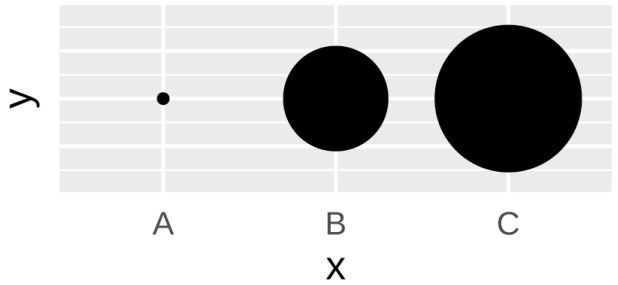


Aesthetics

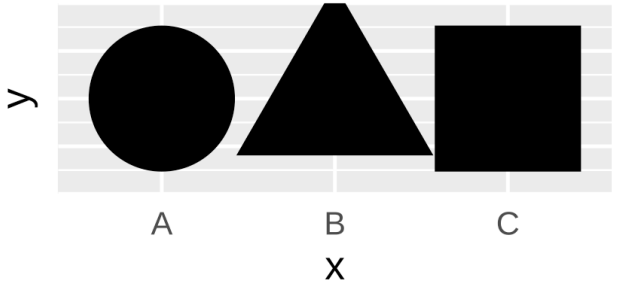
`color` (discrete)



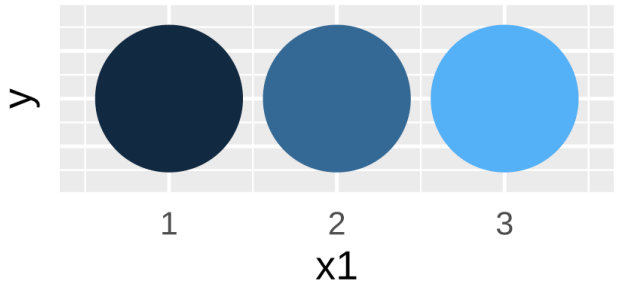
`size`



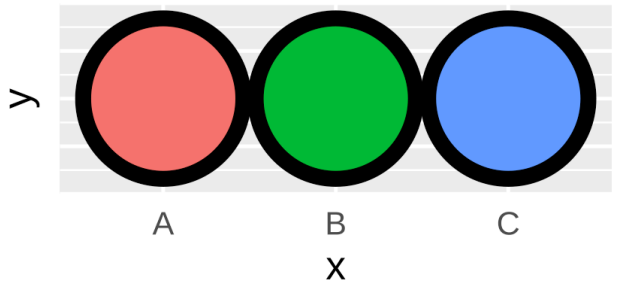
`shape`



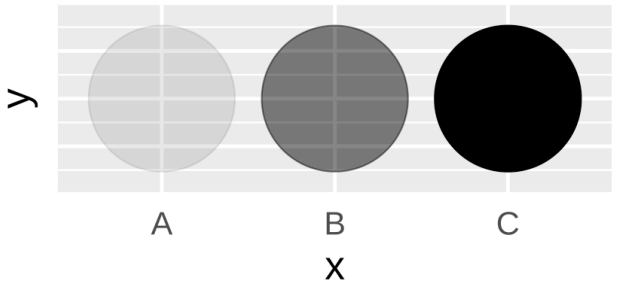
`color` (continuous)



`fill`



`alpha` (opacity)



Geometries

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

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

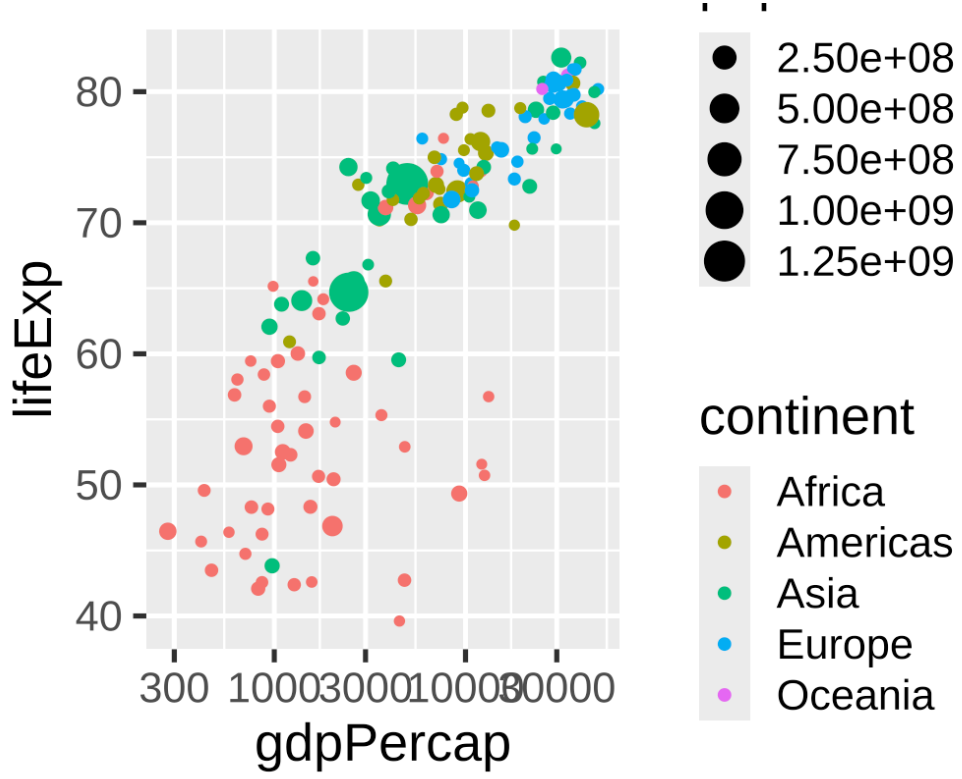
Scales

Scales change the properties of the variable mapping

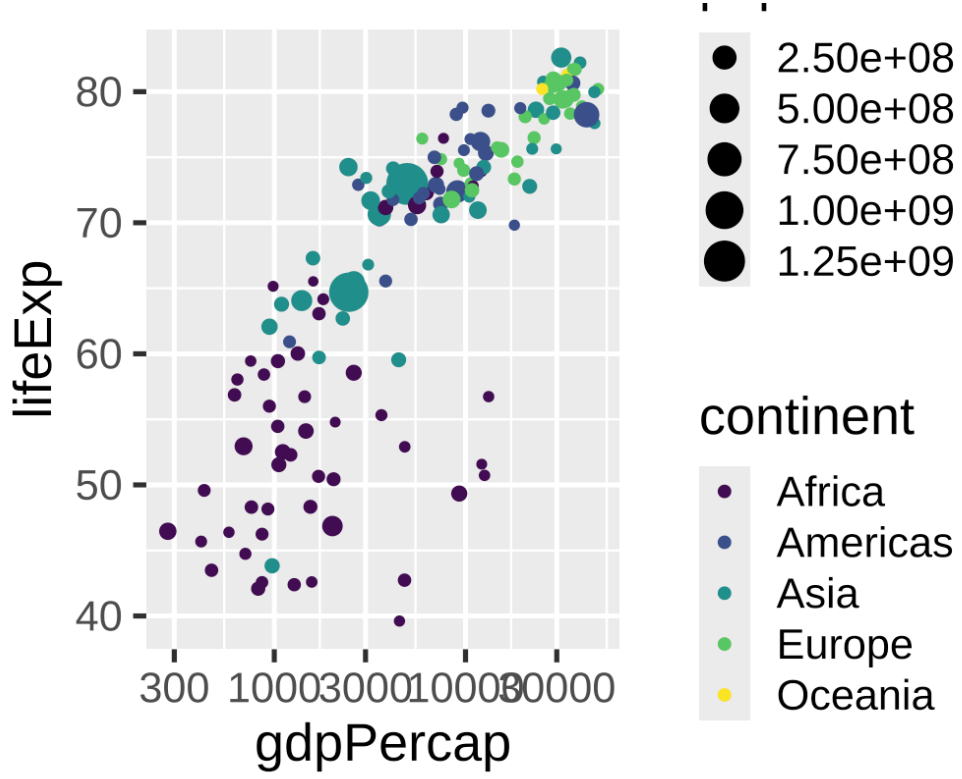
Example layer	What it does
`scale_x_continuous()`	Make the x-axis continuous
`scale_x_continuous(breaks = 1:5)`	Manually specify axis ticks
`scale_x_log10()`	Log the x-axis
`scale_color_gradient()`	Use a gradient
`scale_fill_viridis_d()`	Fill with discrete viridis colors

Scales

`scale_x_log10()`



`scale_color_viridis_d()`



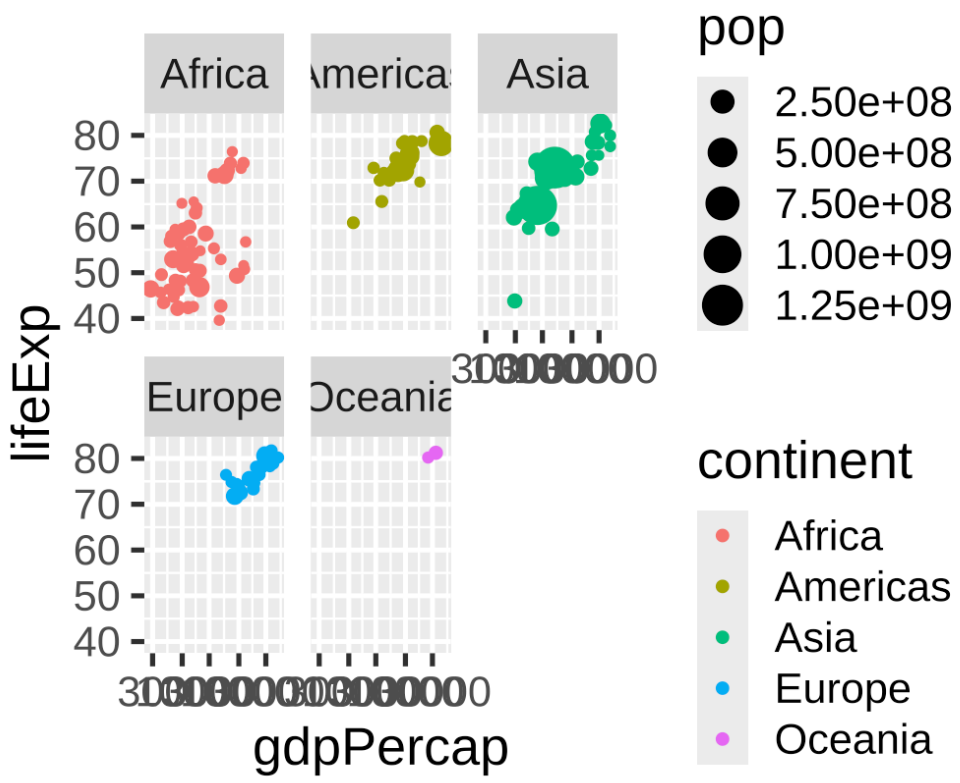
Facets

Facets show subplots for different subsets of data

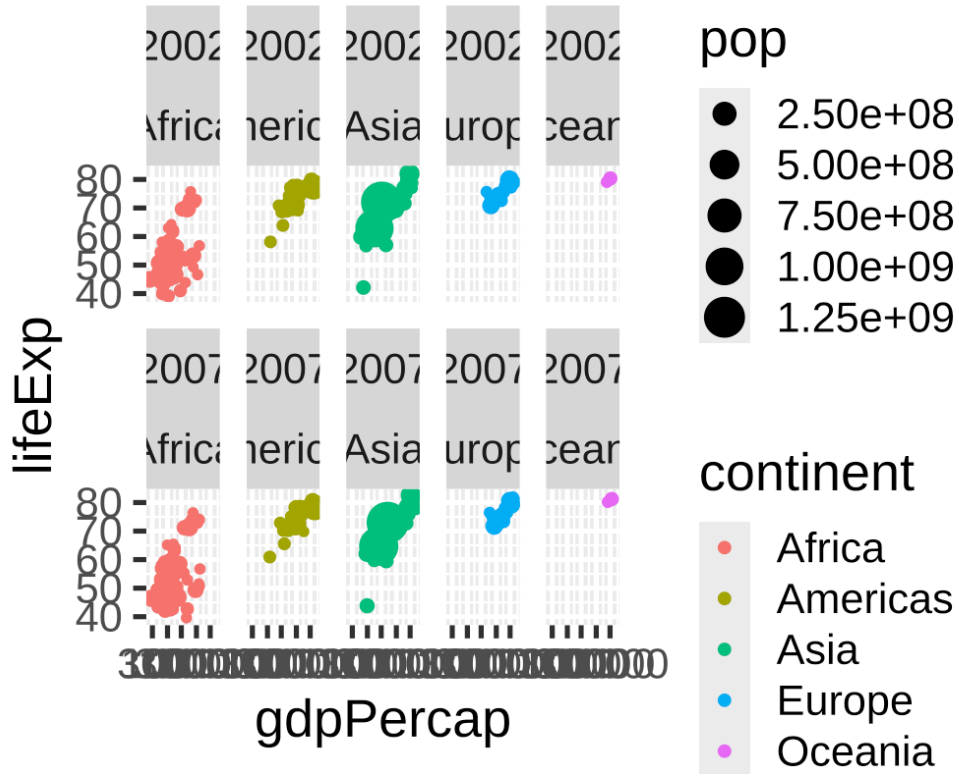
Example layer	What it does
`facet_wrap(vars(continent))`	Plot for each continent
`facet_wrap(vars(continent, year))`	Plot for each continent/year
`facet_wrap(..., ncol = 1)`	Put all facets in one column
`facet_wrap(..., nrow = 1)`	Put all facets in one row

Facets

`facet_wrap(vars(continent))`



`facet_wrap(vars(continent, year))`



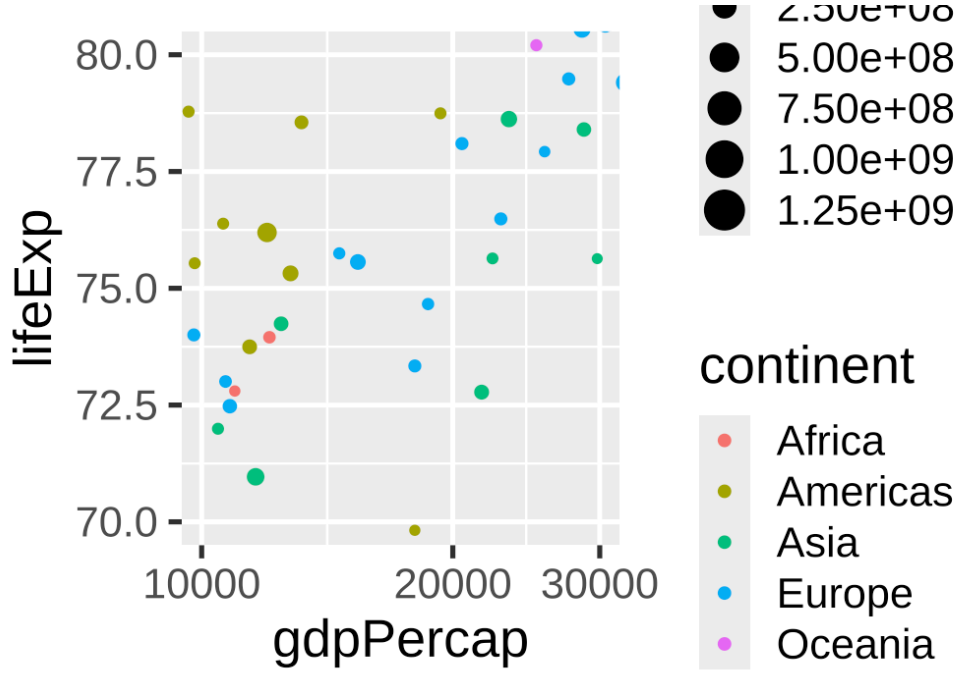
Coordinates

Change the coordinate system

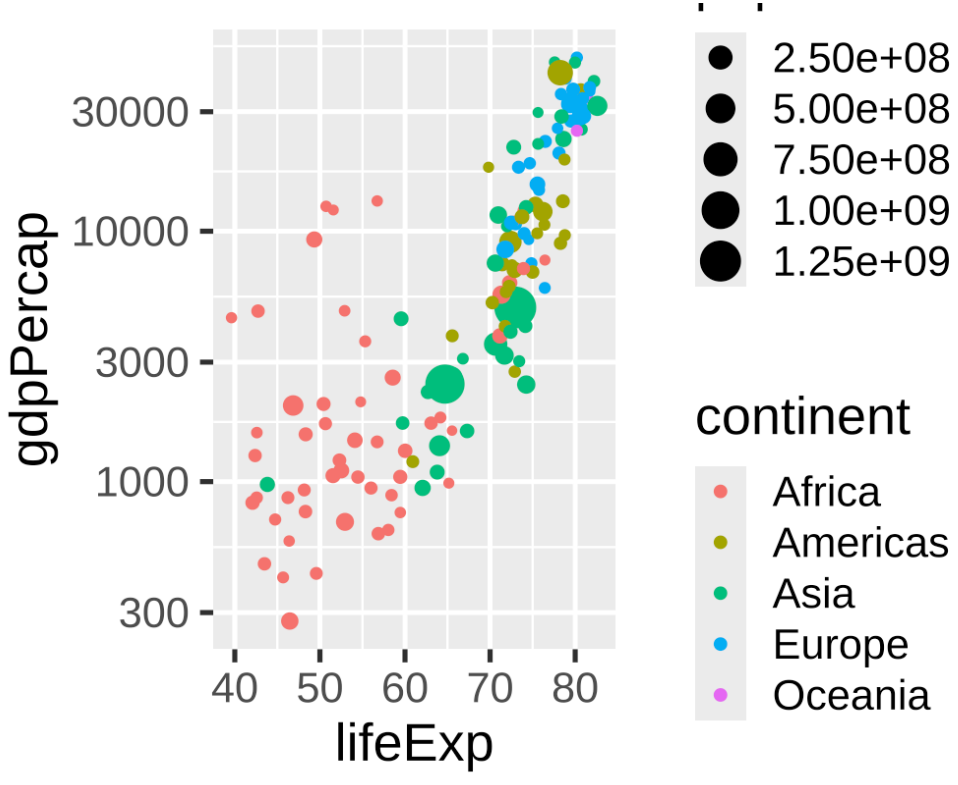
Example layer	What it does
`coord_cartesian(ylim = c(1, 10))`	Zoom in where y is 1–10
`coord_flip()`	Switch x and y
`coord_polar()`	Use polar coordinates

Coordinates

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




```
coord_flip()
```



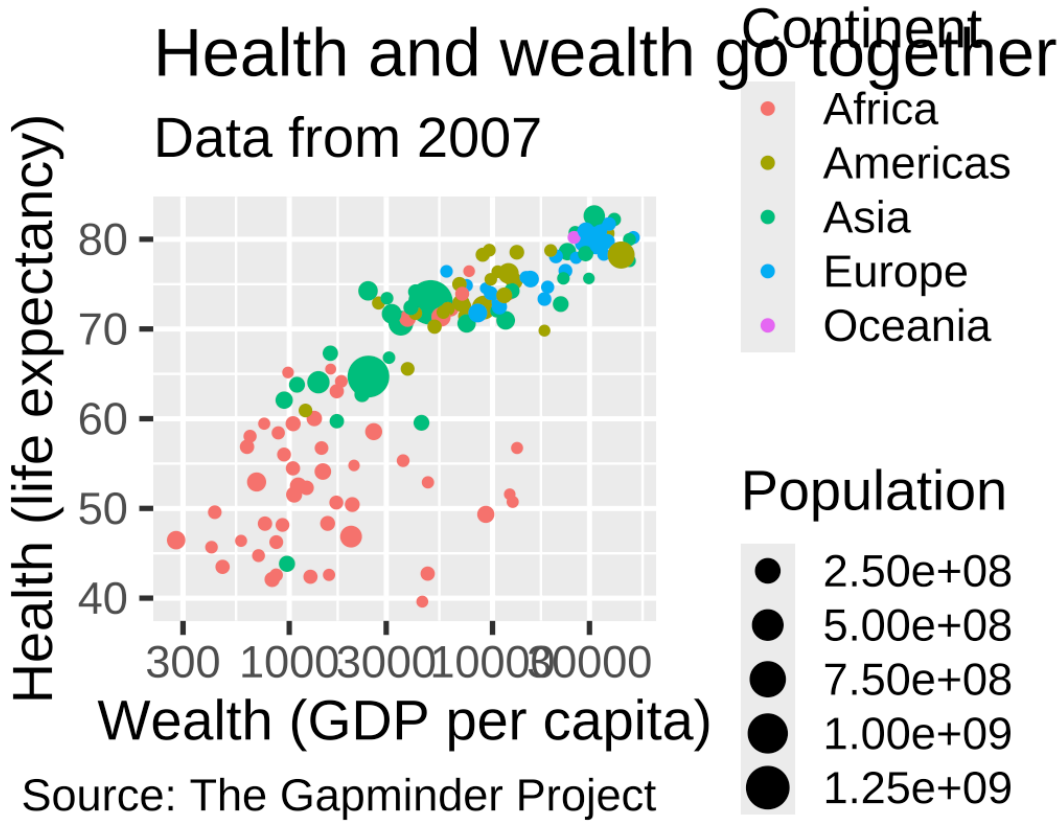
Labels

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

```
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

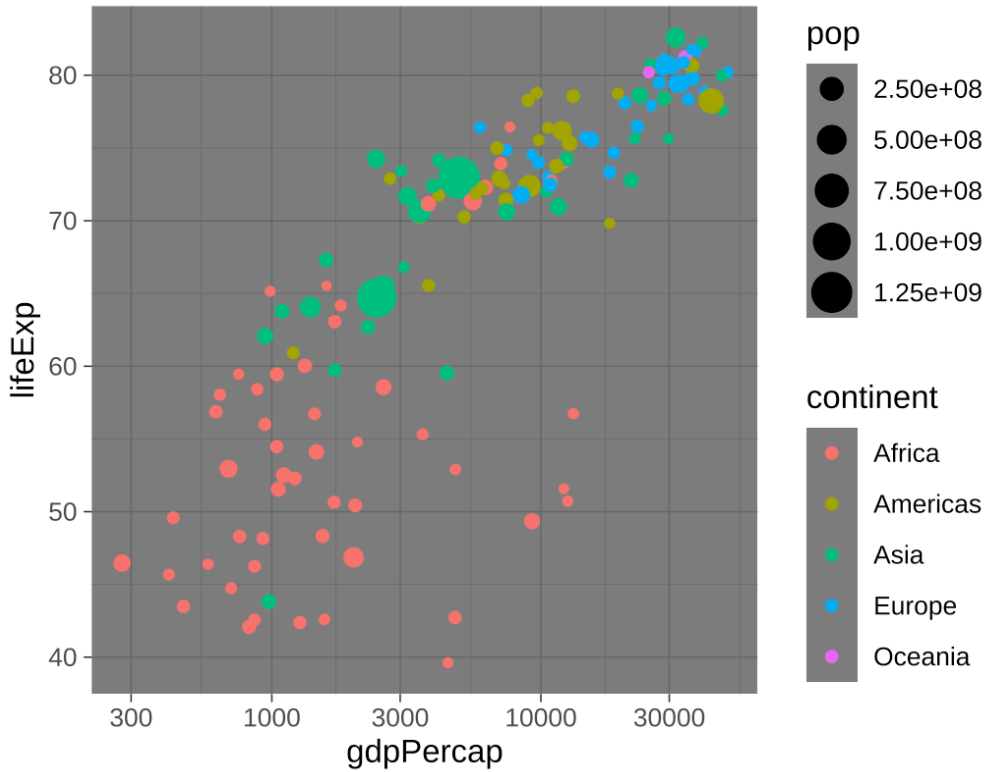
`theme()` can be used to change the appearance of anything in a plot

Lots of themes built in and available from other packages

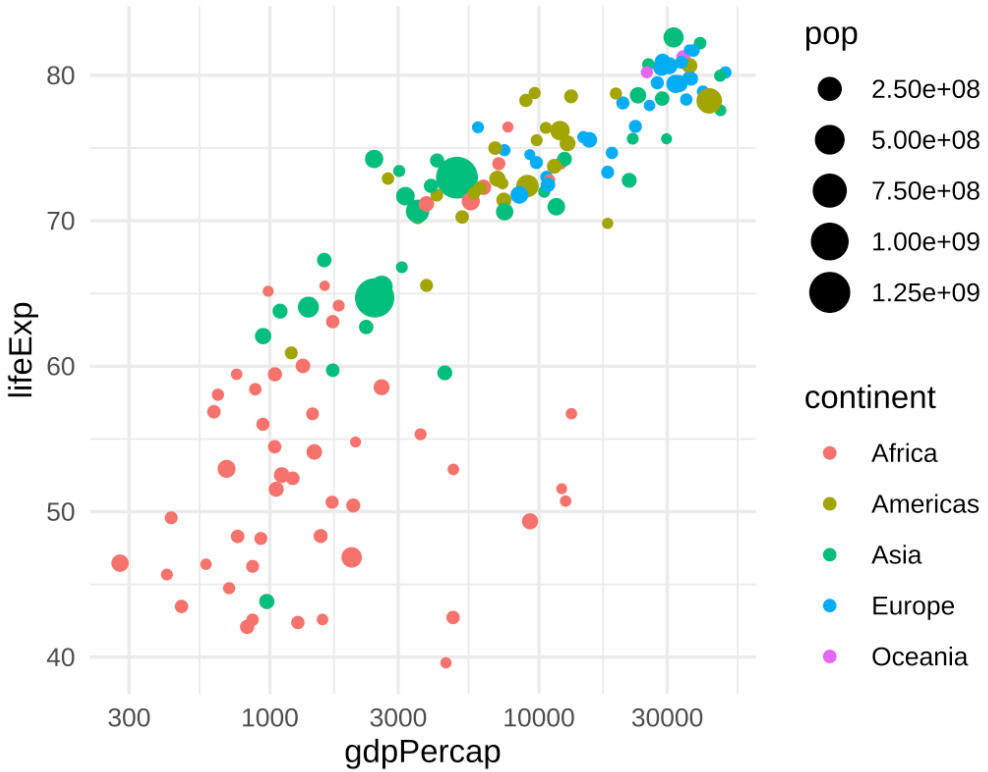
Example layer	What it does
`theme_grey()`	Default grey background
`theme_bw()`	Black and white
`theme_dark()`	Dark
`theme_minimal()`	Minimal

Theme

`theme_dark()`

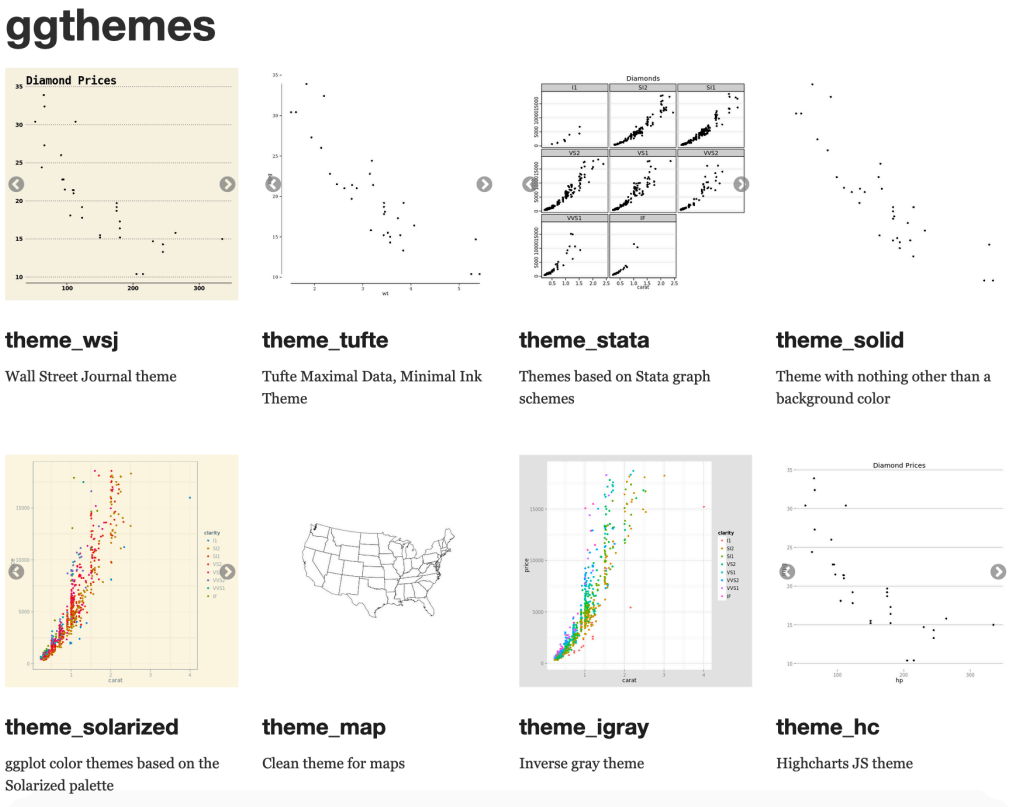


`theme_minimal()`



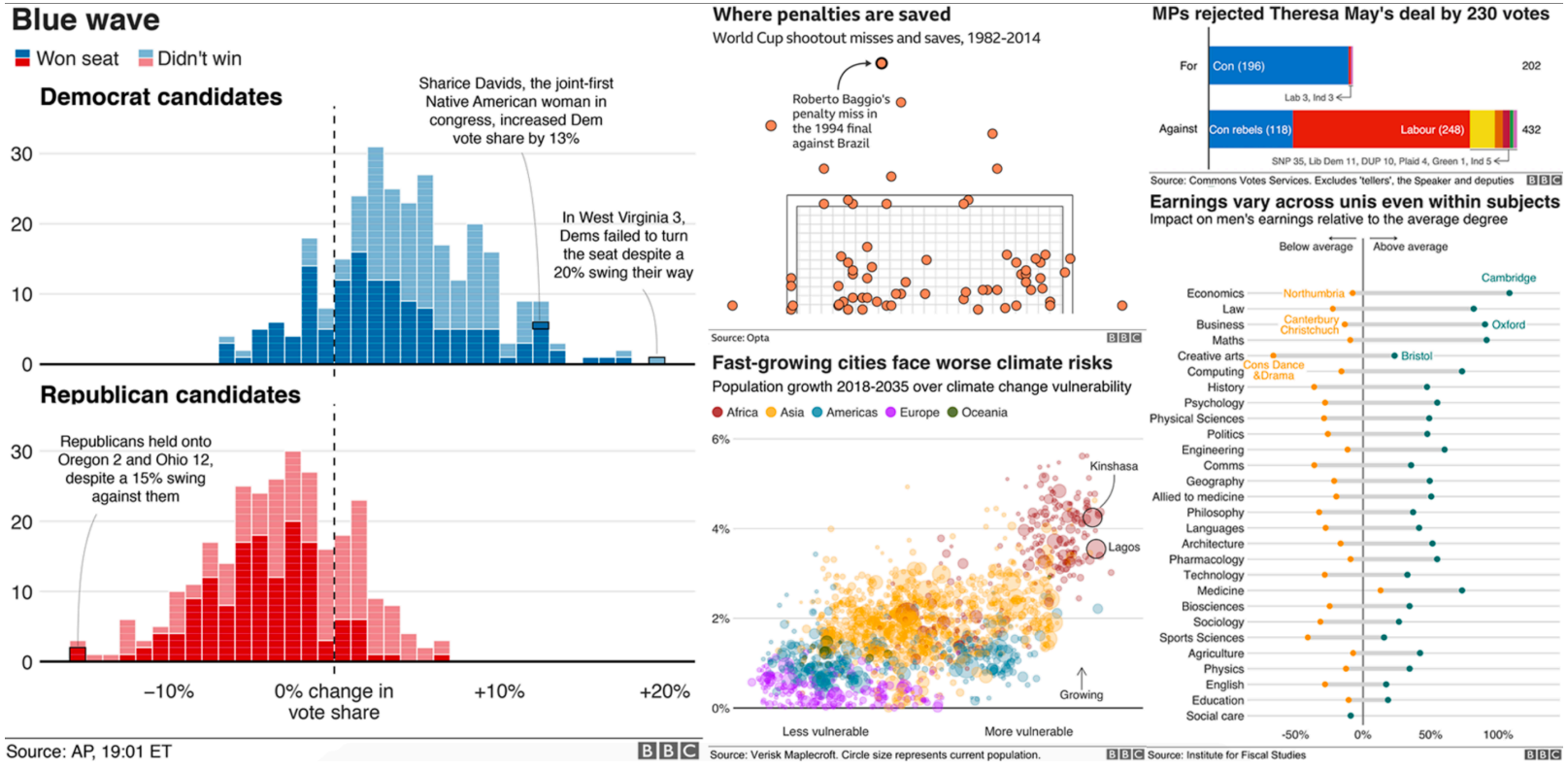
Theme

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



Theme

Organizations often make their own custom themes, like the BBC

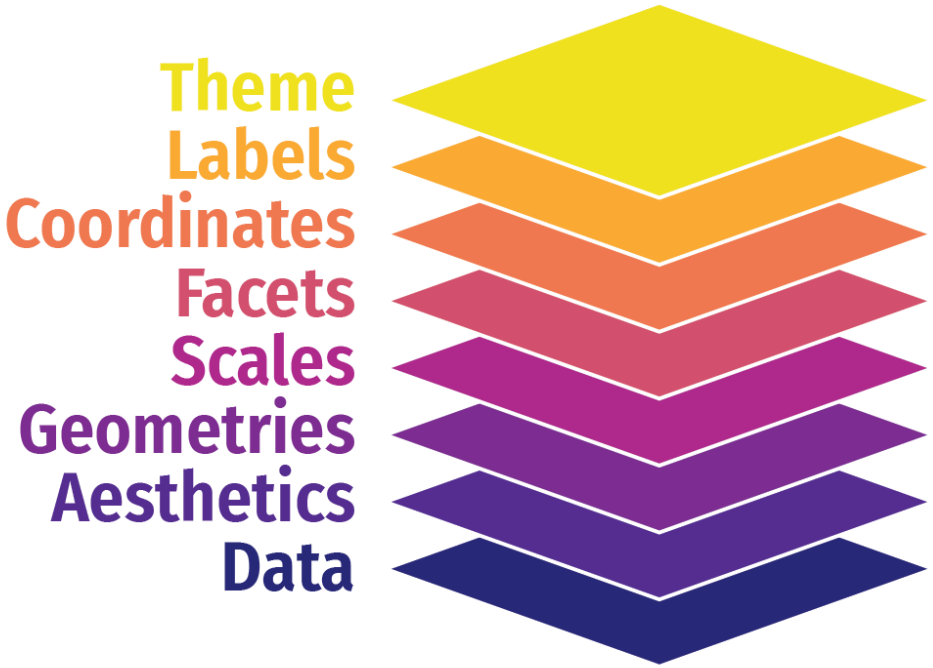


Theme

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