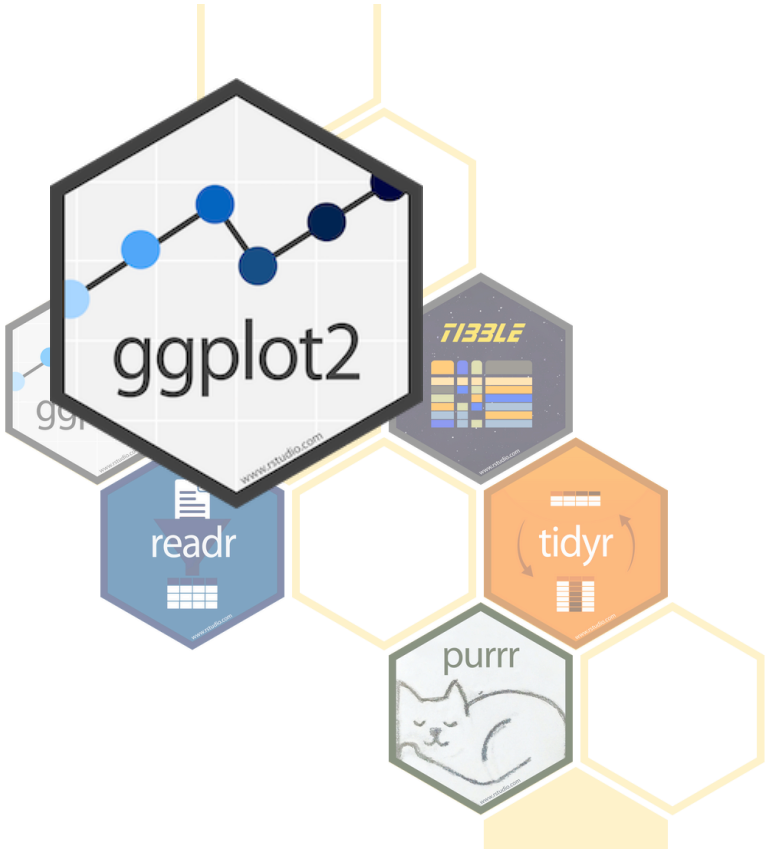
Data visualization

Overview

1. **The `ggplot` function**
2. **Mapping data to aesthetics**
3. **Different geoms**
4. **Scales**
5. **Facets**
6. **Coordinates**
7. **Themes**

The `ggplot` function

tidyverse



The `ggplot()` function

`ggplot()` from the `ggplot2` package is what we're gonna use for all our plots

It takes the following core arguments:

```
ggplot(data, aes()) + geometry + other_stuff
```

- **Data**: the values to plot
- **Mapping** (aes, for aesthetics): the structure of the plot
- **Geometry**: the type of plot

You can also use a pipe

```
data |>
ggplot(aes()) + geometry + other_stuff
```

The `ggplot()` function

Take for instance the gapminder data you've previously installed.

```
library(gapminder)

# The data() function in R is used to list, load,
# and access built-in or package-provided datasets.
data(gapminder)
```

Let's get a quick overview of the data again.

```
head(gapminder)
```

```
# A tibble: 6 × 6
  country     continent  year lifeExp      pop gdpPercap
  <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
1 Afghanistan Asia       1952    28.8  8425333      779.
2 Afghanistan Asia       1957    30.3  9240934      821.
3 Afghanistan Asia       1962    32.0 10267083      853.
4 Afghanistan Asia       1967    34.0 11537966      836.
5 Afghanistan Asia       1972    36.1 13079460      740.
6 Afghanistan Asia       1977    38.4 14880372      786.
```

What's the relationship between life expectancy and GDP per capita?

… we expect of course that higher GDP per capita leads to greater life expactancy.

What's the relationship between life expectancy and GDP per capita?

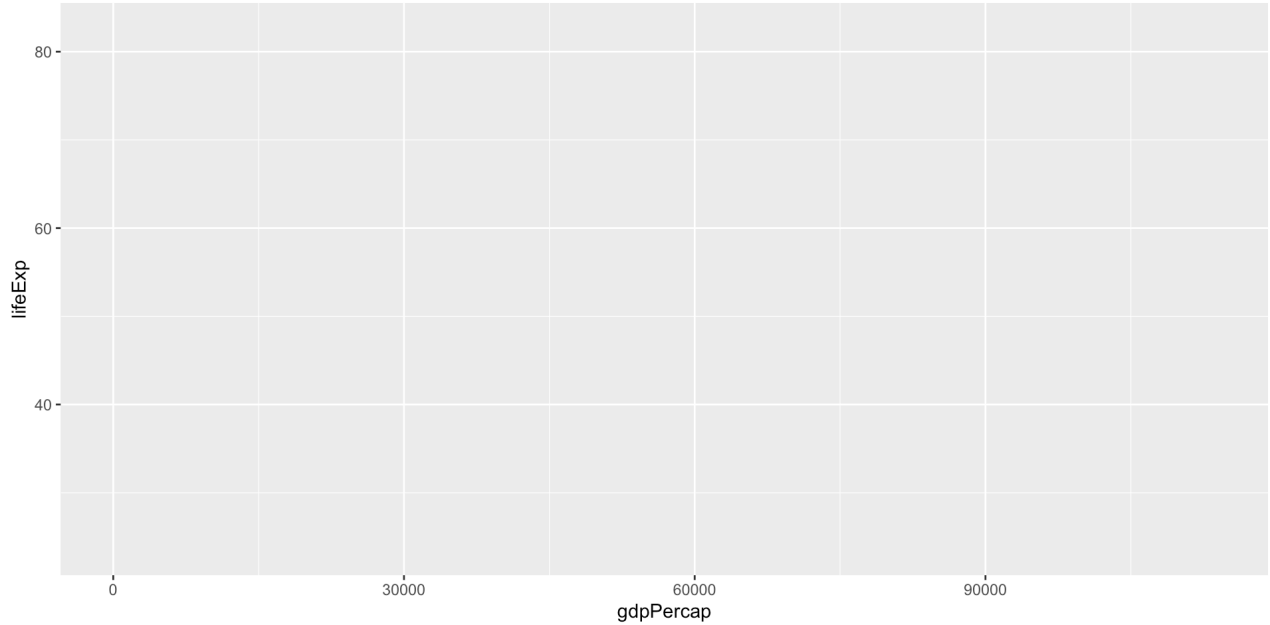
- We first assign the gapminder data to `ggplot()`
- The result is just an empty plot

```
1  ggplot(data = gapminder)
```

What's the relationship between life expectancy and GDP per capita?

- Next, we map out the plot by adding the x and y axes

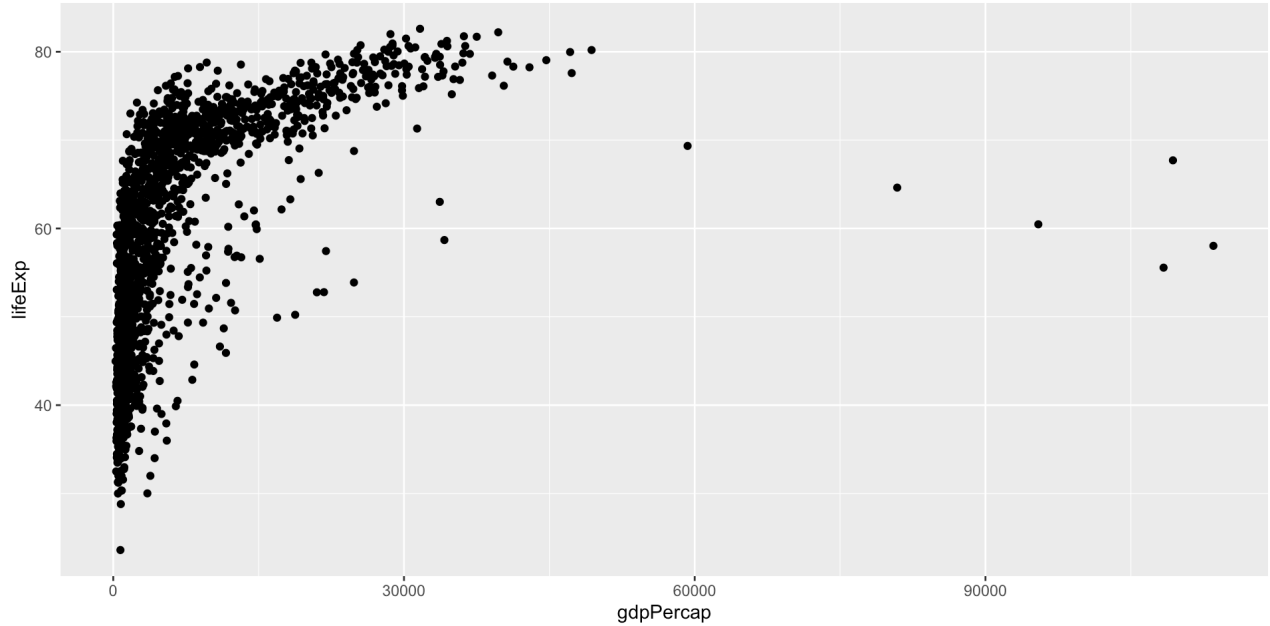
```
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp))
```



What's the relationship between life expectancy and GDP per capita?

- We then define how we want to plot our data
- In this case, let's go for the raw data points

```
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point()
```



Mapping data to aesthetics

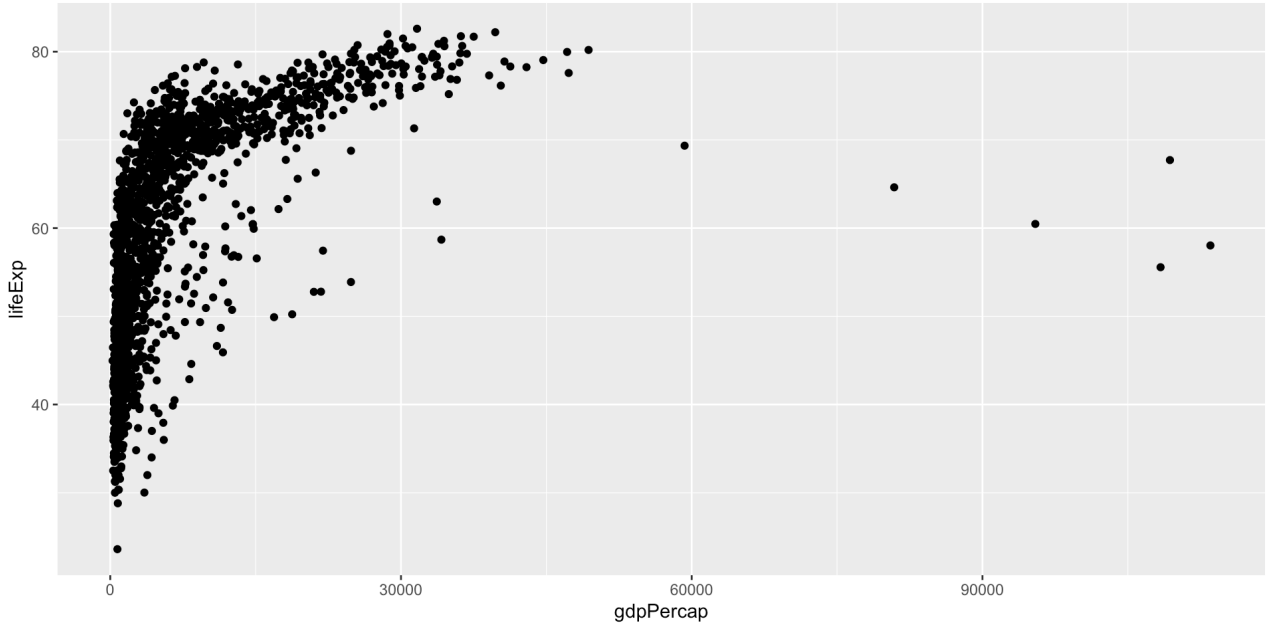
So far, only two variables appear in our plot (mapped onto the x and the y axis)

But we can add more variables to the plot, by assigning them to certain asthetics

Mapping data to aesthetics

- For example, we can display the variable continent as colors

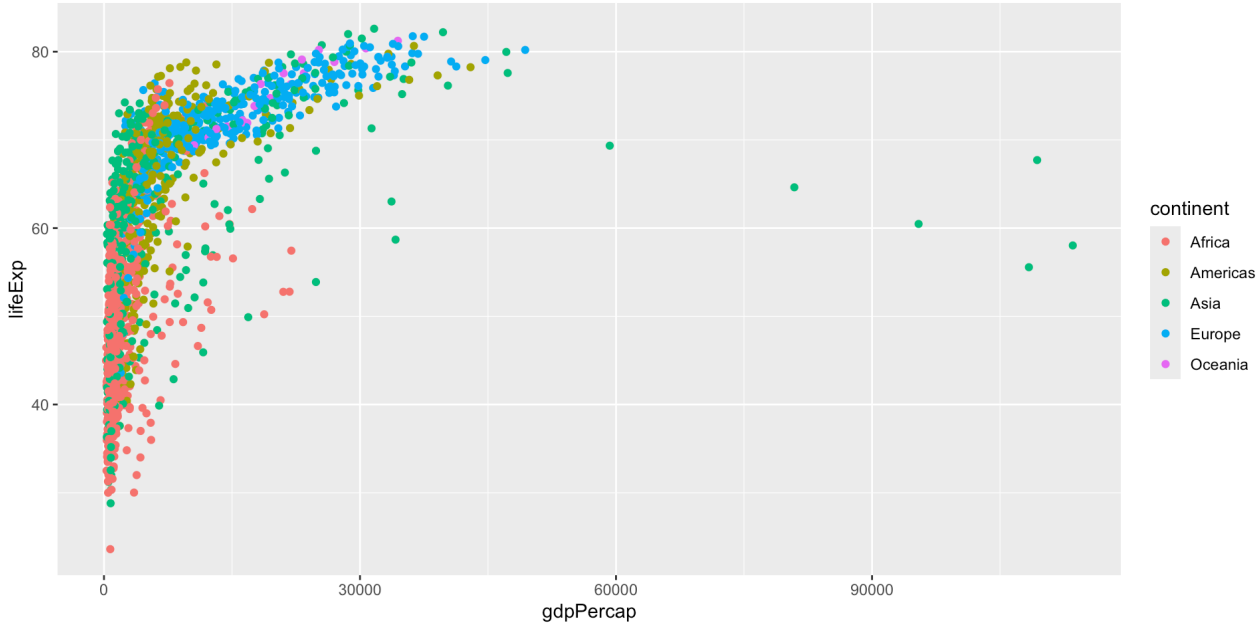
```
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point()
```



Mapping data to aesthetics

- For example, we can display the variable continent as colors
- Note that a legend gets added automatically to the plot

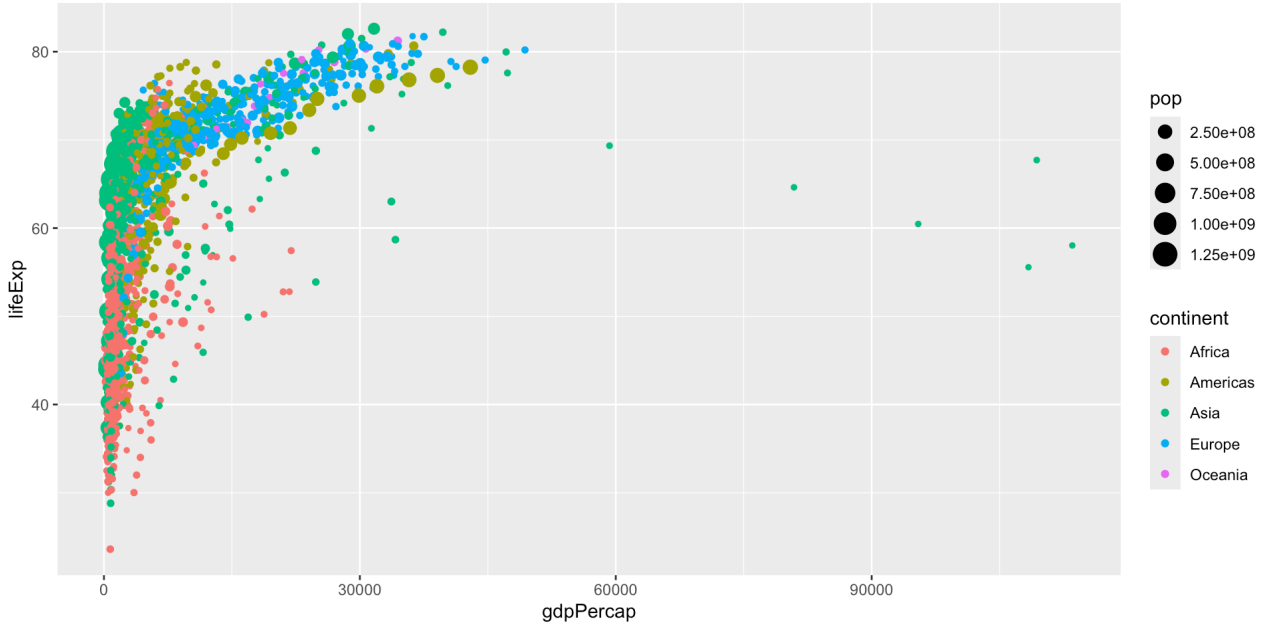
```
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp, color = continent)) +
  geom_point()
```



Mapping data to aesthetics

- We could further display population size by mapping it to the size aesthetic

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



Grammatical Layers

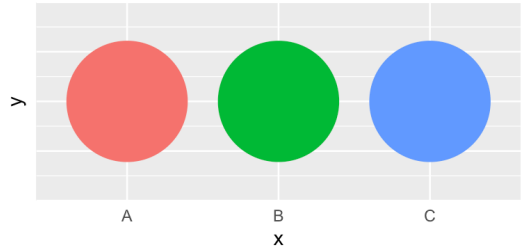
So far we know about data, aesthetics, and geometries

Think of these components as **layers**

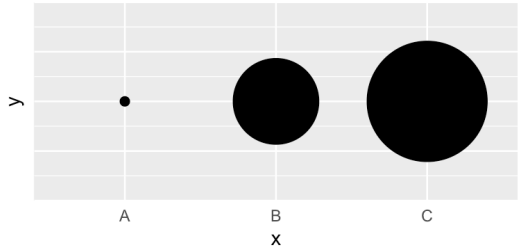
We add them to foundational `ggplot()` with `+`

Possible aesthetics

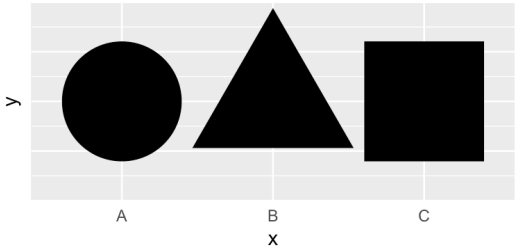
`color` (discrete)



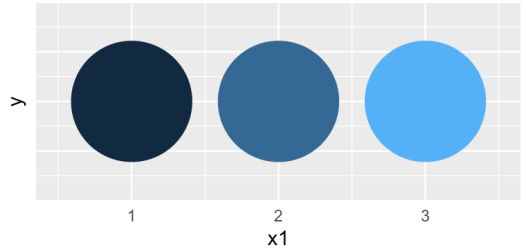
`size`



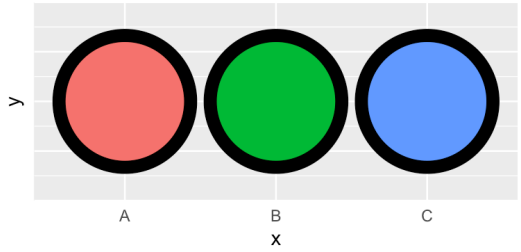
`shape`



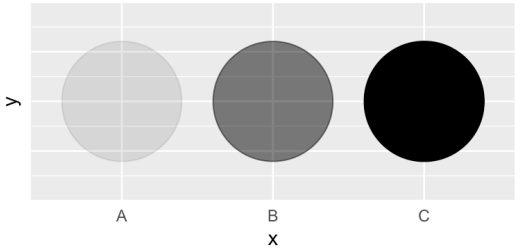
`color` (continuous)



`fill`



`alpha`



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

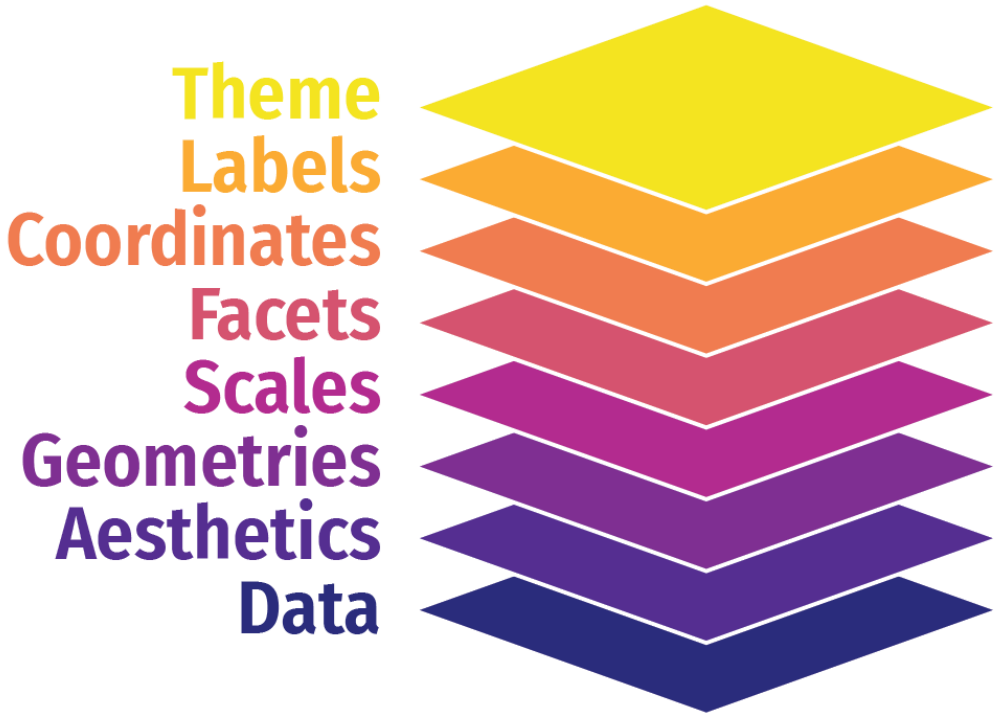
There are dozens of possible geoms and
each class session will cover different ones.

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

Additional Layers

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

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



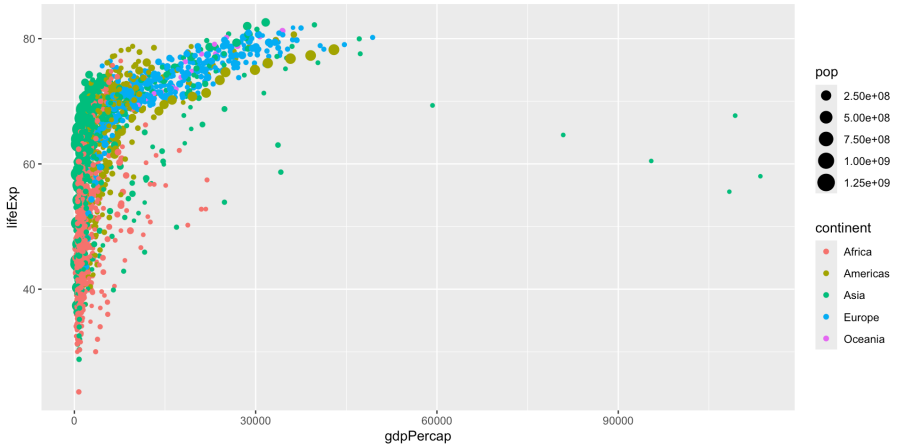
Scales

Scales change how variables are mapped

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

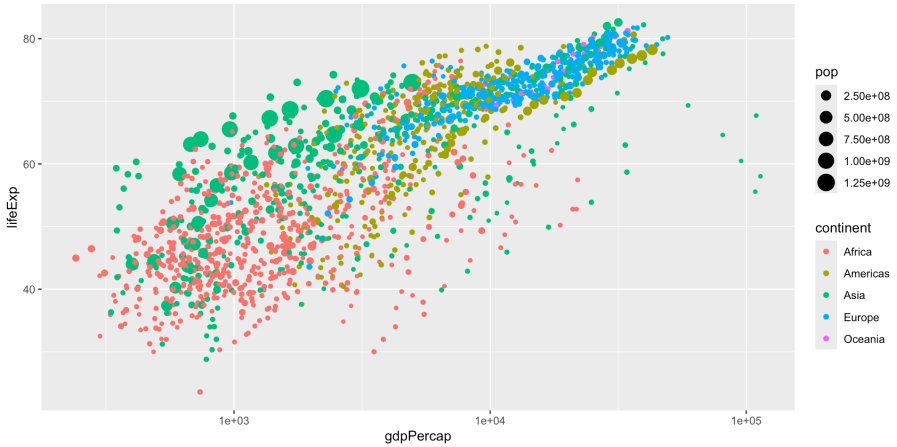
Scales

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



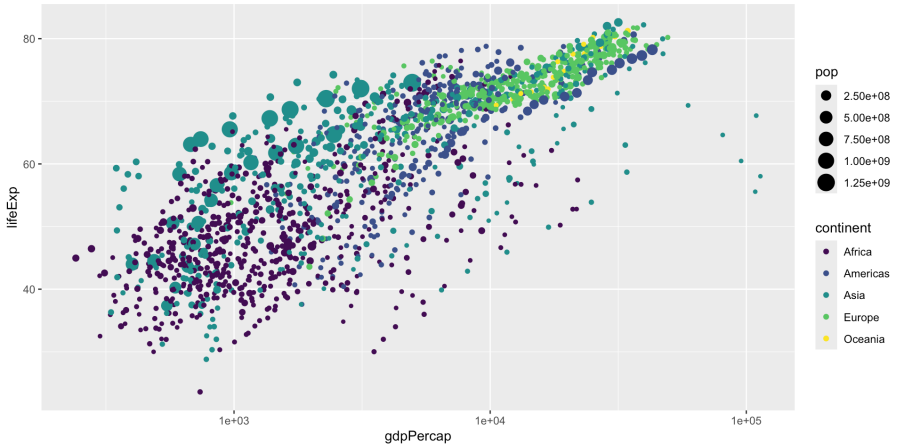
Scales

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



Scales

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



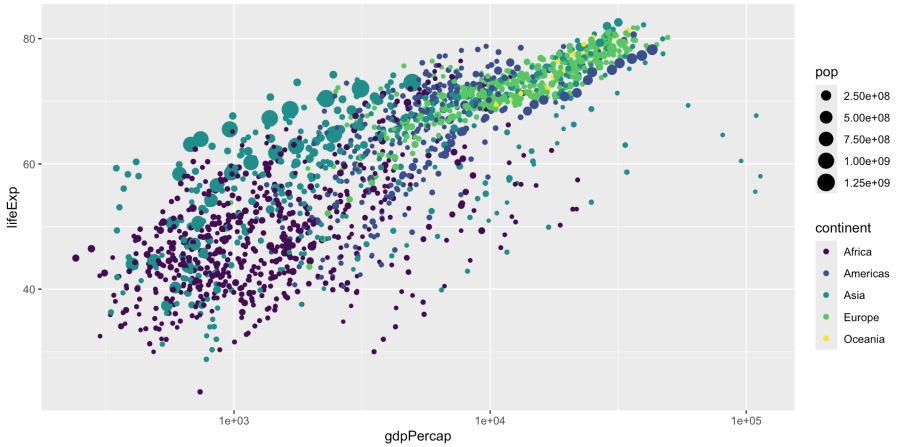
Facets

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

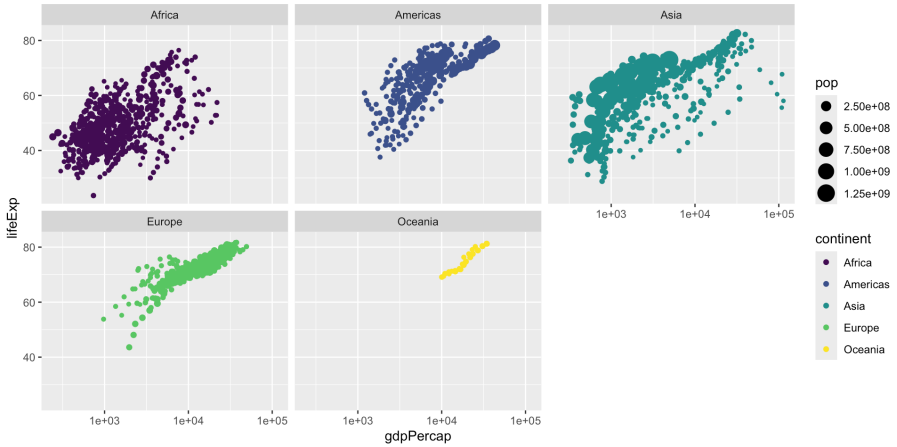
Facets

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



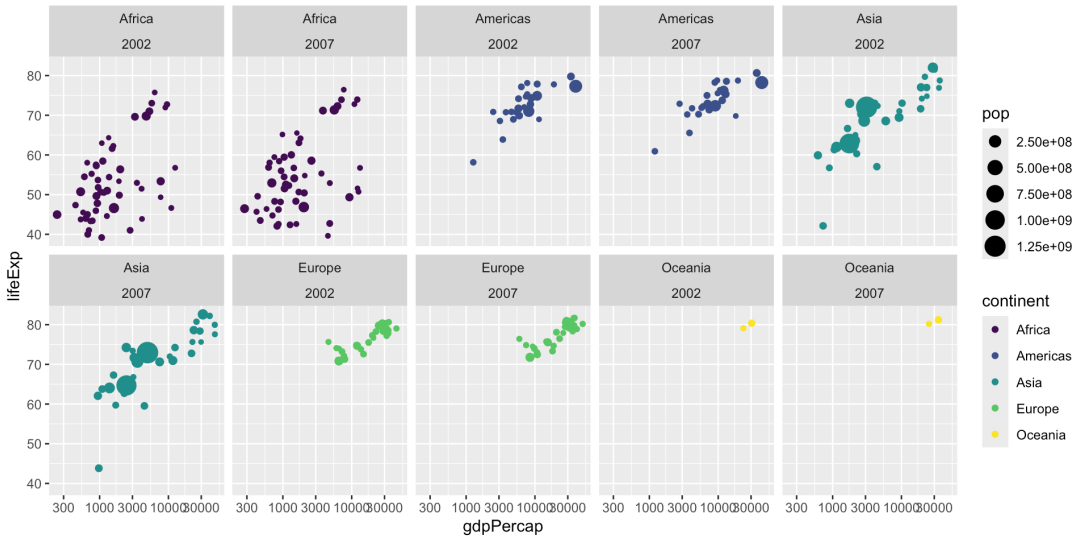
Facets

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



Facets

```
ggplot(data = gapminder |>
         filter(year %in% c(2002, 2007)
       mapping = aes(x = gdpPercap, y =
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  scale_color_viridis_d() +
  facet_wrap(vars(continent,
                  year), nrow = 2)
```



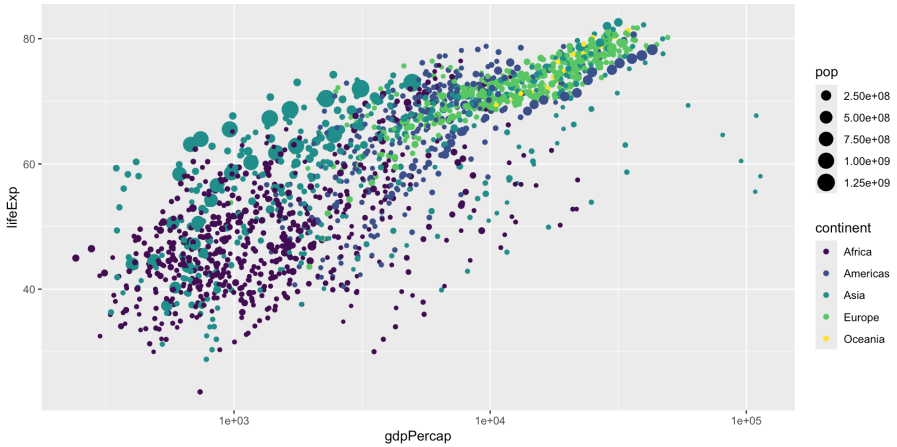
Coordinates

Change the coordinate system

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

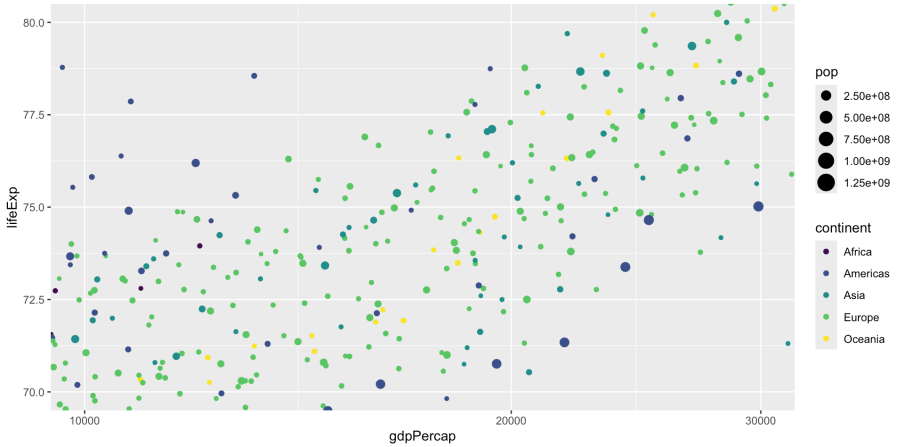
Coordinates

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



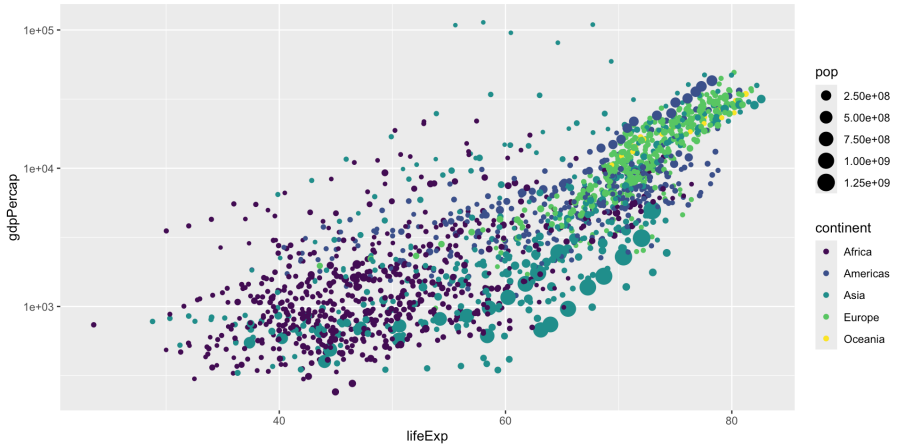
Coordinates

```
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp, c
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  scale_color_viridis_d() +
  coord_cartesian(ylim = c(70, 80),
                  xlim = c(10000, 30000))
```



Coordinates

```
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp, c
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  scale_color_viridis_d() +
  coord_flip()
```

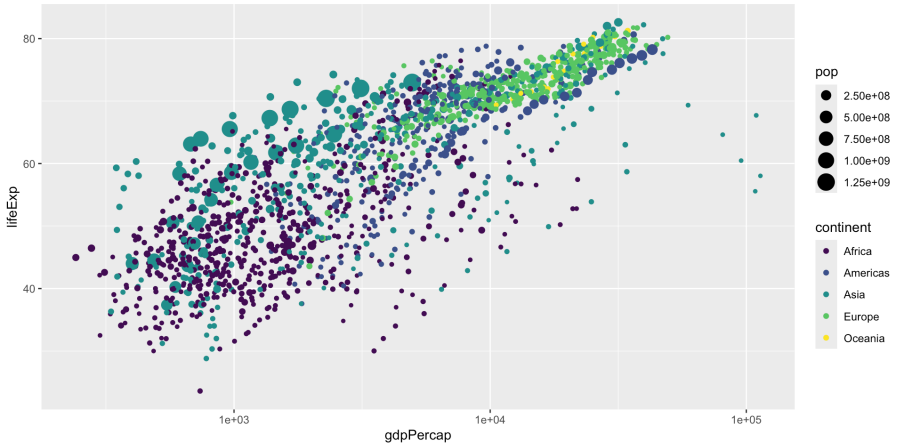


Labels

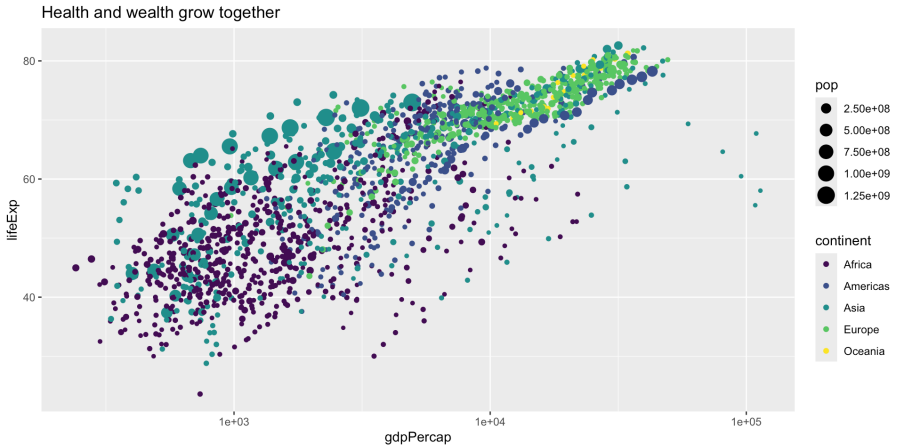
Add labels to the plot with a single `labs()` layer

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

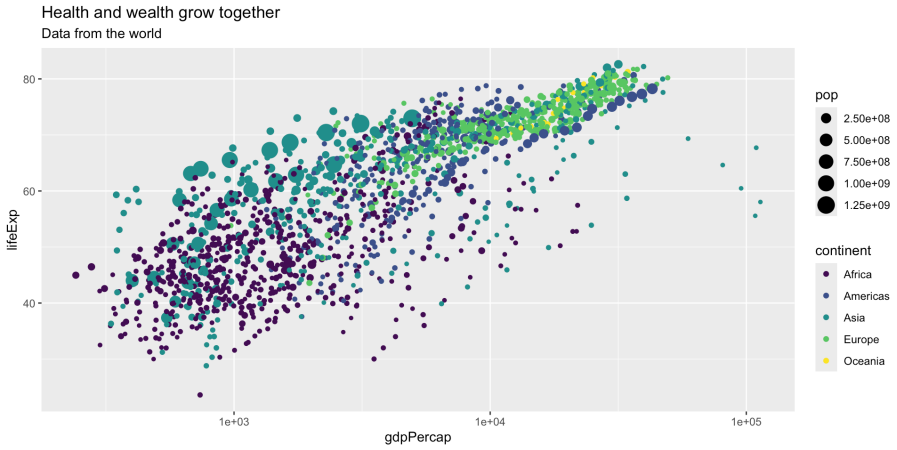
```
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp, c
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  scale_color_viridis_d()
```



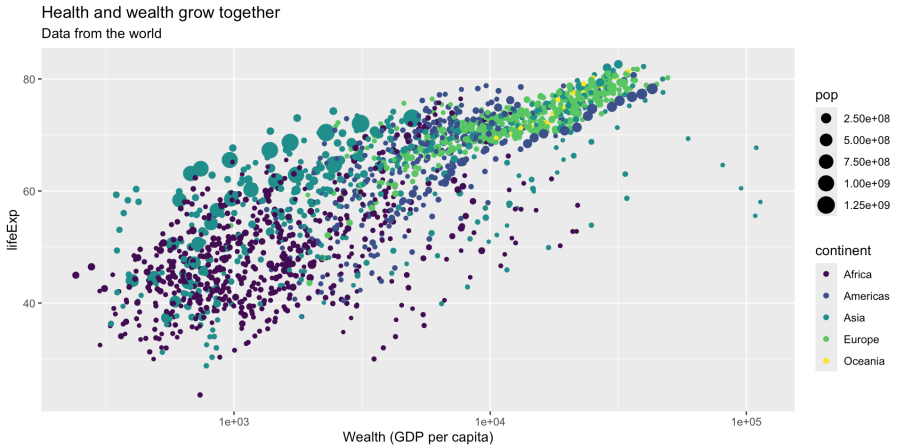
```
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp, c
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  scale_color_viridis_d() +
  labs(title = "Health and wealth grow together")
```



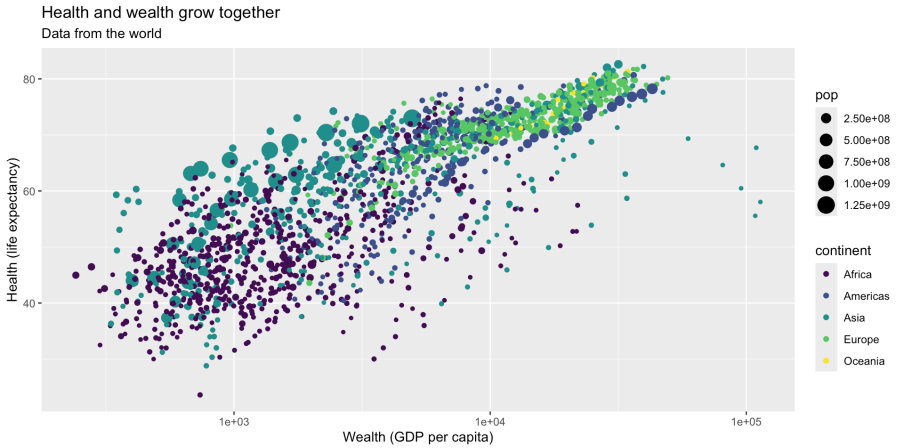
```
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp, c
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  scale_color_viridis_d() +
  labs(title = "Health and wealth grow together",
       subtitle = "Data from the world")
```



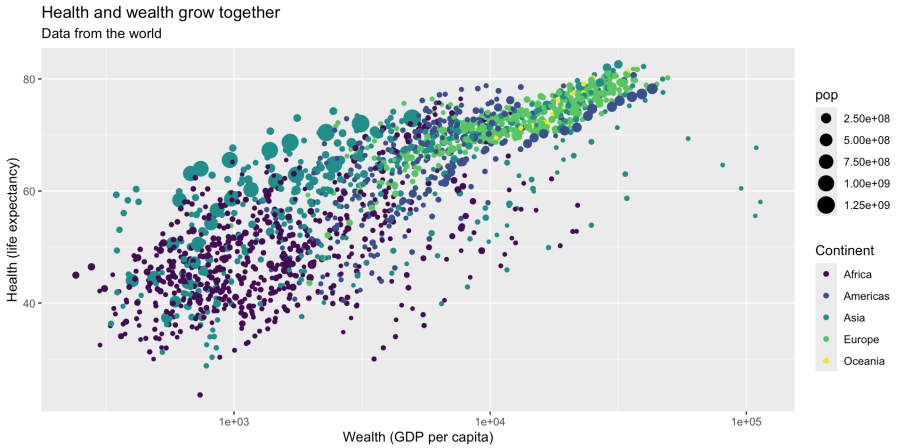
```
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp, c
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  scale_color_viridis_d() +
  labs(title = "Health and wealth grow together",
       subtitle = "Data from the world",
       x = "Wealth (GDP per capita)")
```



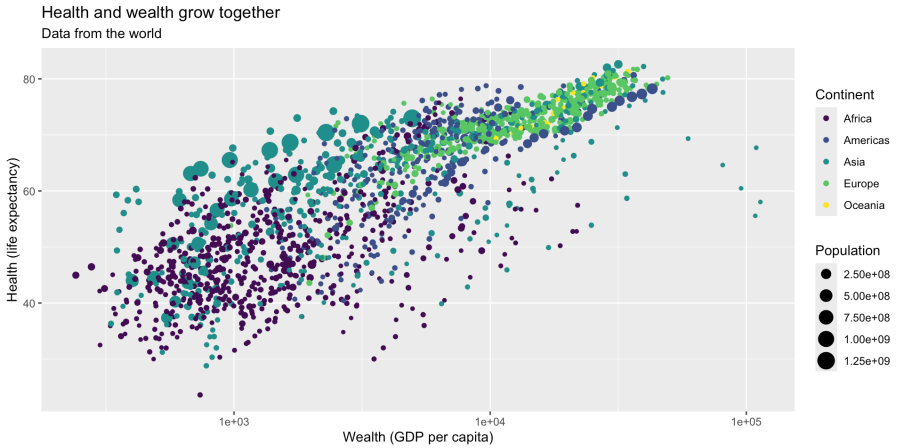
```r
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp, c
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  scale_color_viridis_d() +
  labs(title = "Health and wealth grow together",
       subtitle = "Data from the world",
       x = "Wealth (GDP per capita)",
       y = "Health (life expectancy)")
```



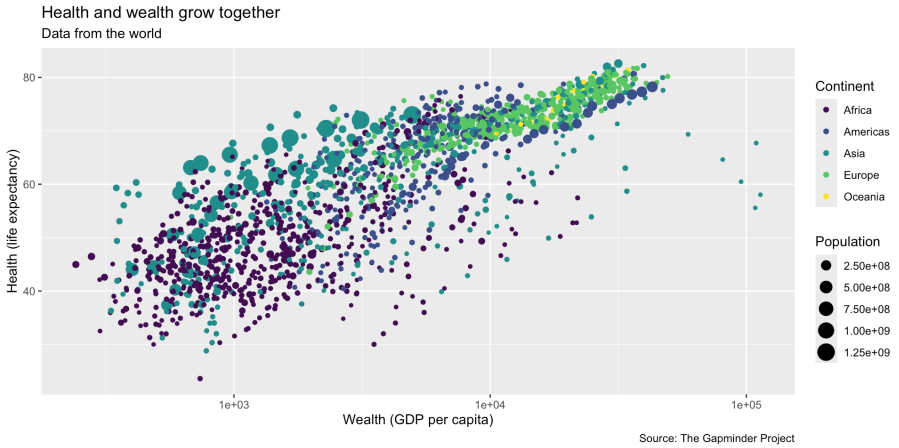
```r
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp, c
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  scale_color_viridis_d() +
  labs(title = "Health and wealth grow together",
       subtitle = "Data from the world",
       x = "Wealth (GDP per capita)",
       y = "Health (life expectancy)",
       color = "Continent")
```



```
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp, c
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  scale_color_viridis_d() +
  labs(title = "Health and wealth grow together",
       subtitle = "Data from the world",
       x = "Wealth (GDP per capita)",
       y = "Health (life expectancy)",
       color = "Continent",
       size = "Population")
```



```r
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp, c
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  scale_color_viridis_d() +
  labs(title = "Health and wealth grow together",
       subtitle = "Data from the world",
       x = "Wealth (GDP per capita)",
       y = "Health (life expectancy)",
       color = "Continent",
       size = "Population",
       caption = "Source: The Gapminder Project")
```



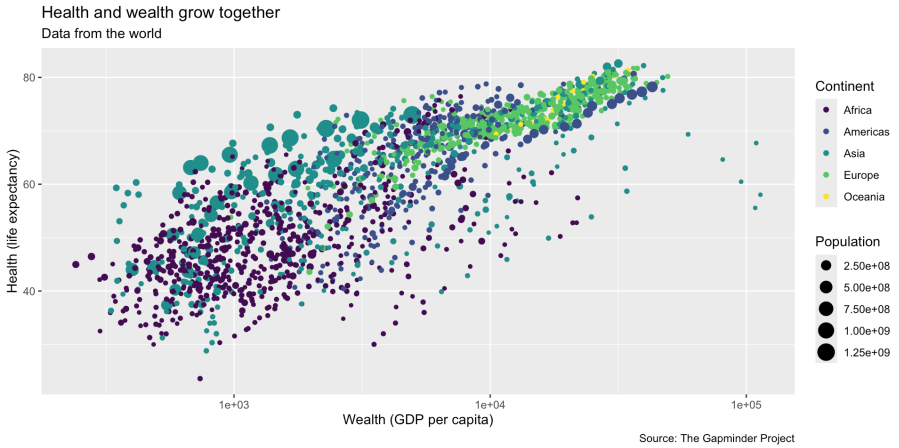
Theme

Change the appearance of anything in the plot

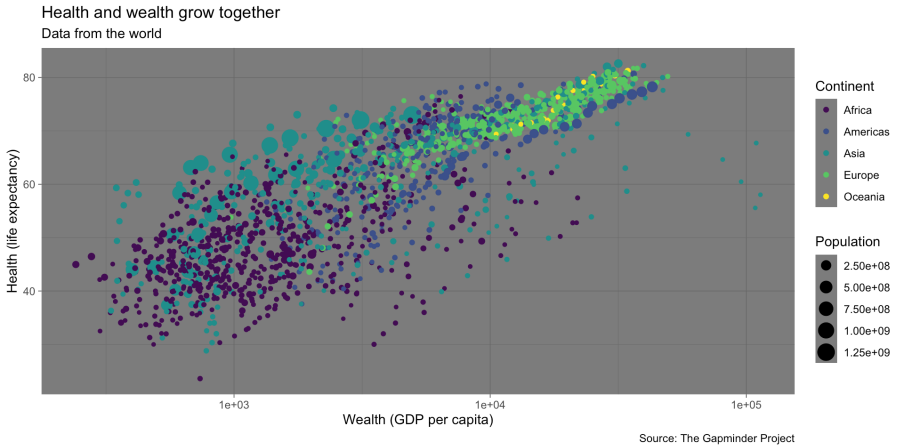
There are many built-in themes

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

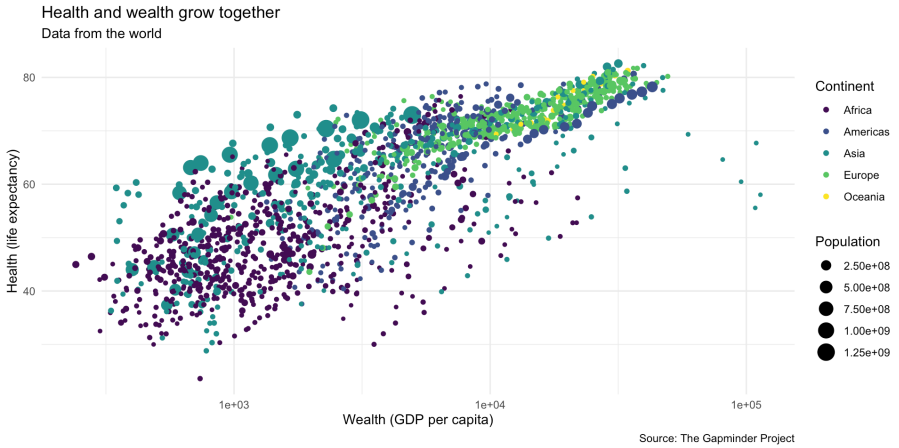
```r
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp, c
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  scale_color_viridis_d() +
  labs(title = "Health and wealth grow together",
       subtitle = "Data from the world",
       x = "Wealth (GDP per capita)",
       y = "Health (life expectancy)",
       color = "Continent",
       size = "Population",
       caption = "Source: The Gapminder Project")
```



```r
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp, c
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  scale_color_viridis_d() +
  labs(title = "Health and wealth grow together",
       subtitle = "Data from the world",
       x = "Wealth (GDP per capita)",
       y = "Health (life expectancy)",
       color = "Continent",
       size = "Population",
       caption = "Source: The Gapminder Project")
  theme_dark()
```



```r
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp, c
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  scale_color_viridis_d() +
  labs(title = "Health and wealth grow together",
       subtitle = "Data from the world",
       x = "Wealth (GDP per capita)",
       y = "Health (life expectancy)",
       color = "Continent",
       size = "Population",
       caption = "Source: The Gapminder Project")
  theme_minimal()
```



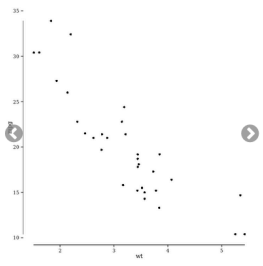
Theme

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

ggthemes

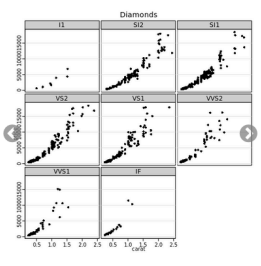
theme_wsj

Wall Street Journal theme



theme_tufte

Tufte Maximal Data, Minimal Ink Theme

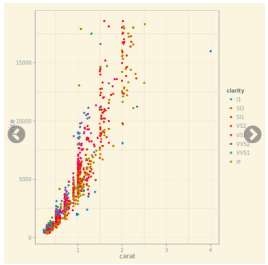


theme_stata

Themes based on Stata graph schemes

theme_solid

Theme with nothing other than a background color

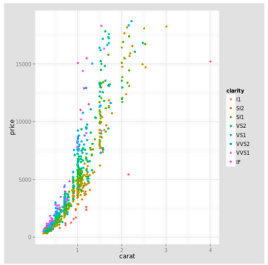


theme_solarized

ggplot color themes based on the Solarized palette

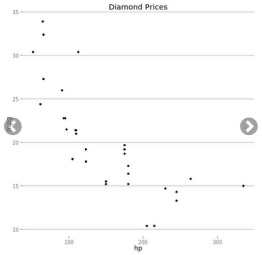
theme_map

Clean theme for maps



theme_igray

Inverse gray theme

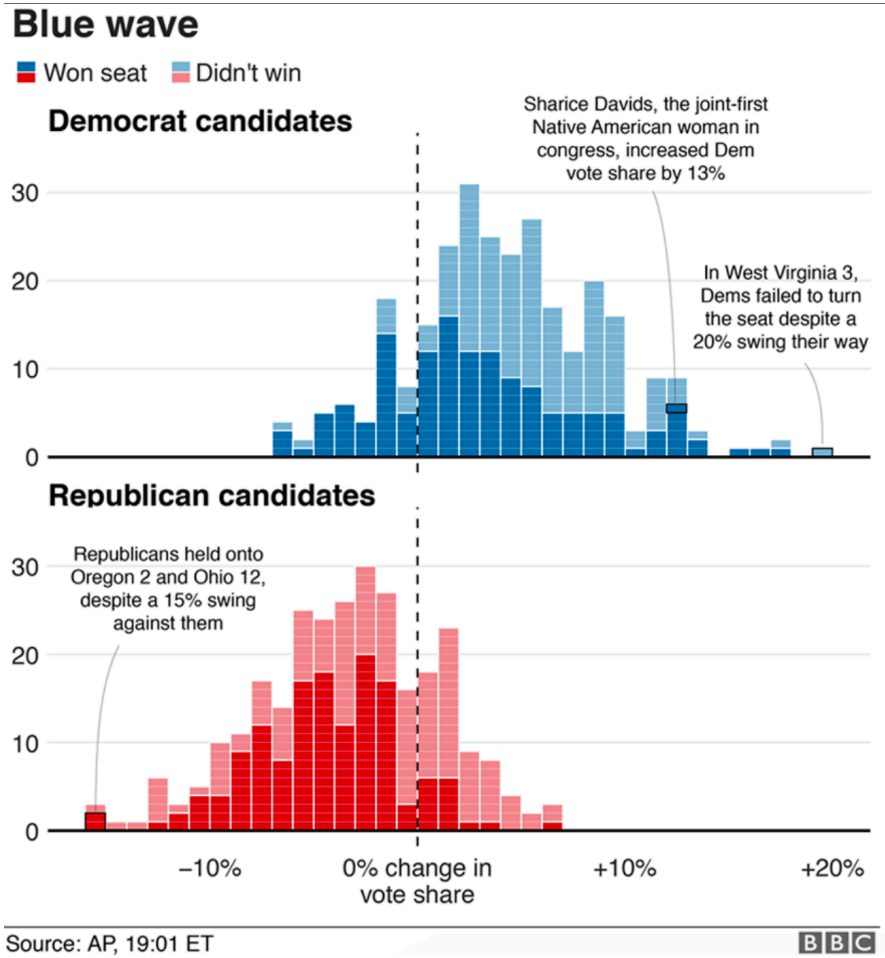


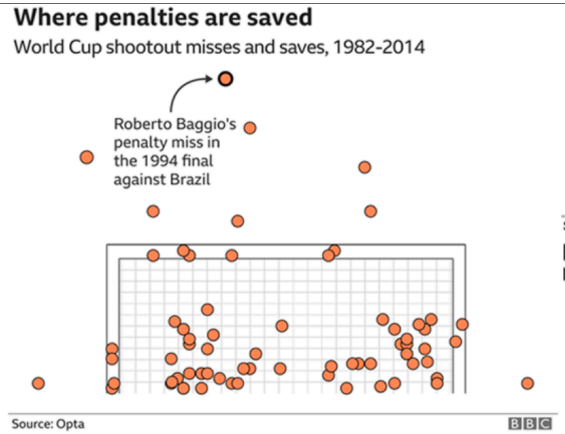
theme_hc

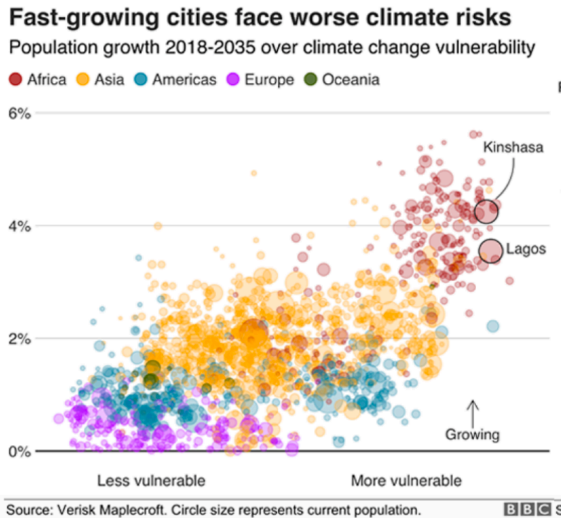
Highcharts JS theme

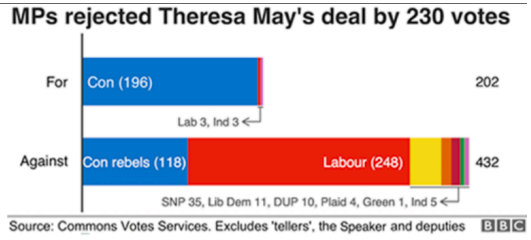
Theme

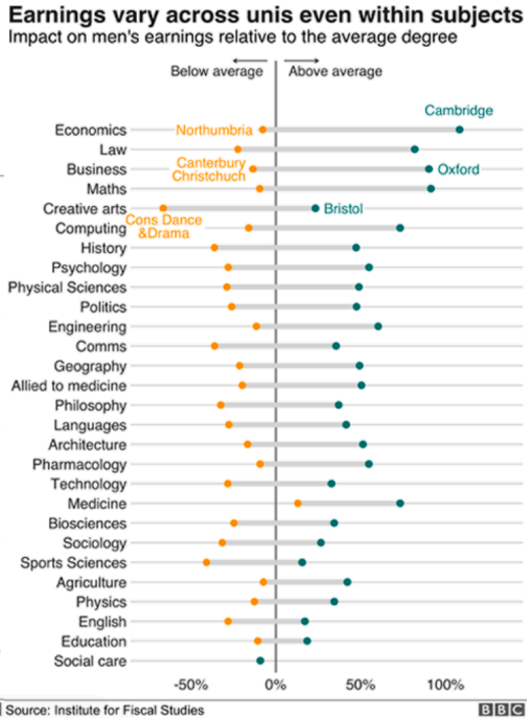
Organizations often make their own custom themes, like the BBC









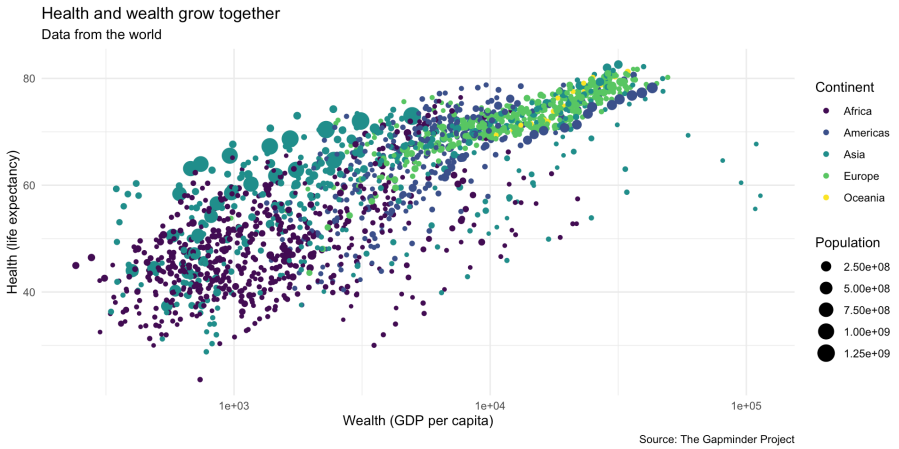


Theme options

Make theme adjustments with `theme()`

There are a billion options here!

```
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp, c
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  scale_color_viridis_d() +
  labs(title = "Health and wealth grow together",
       subtitle = "Data from the world",
       x = "Wealth (GDP per capita)",
       y = "Health (life expectancy)",
       color = "Continent",
       size = "Population",
       caption = "Source: The Gapminder Project")
  theme_minimal()
```

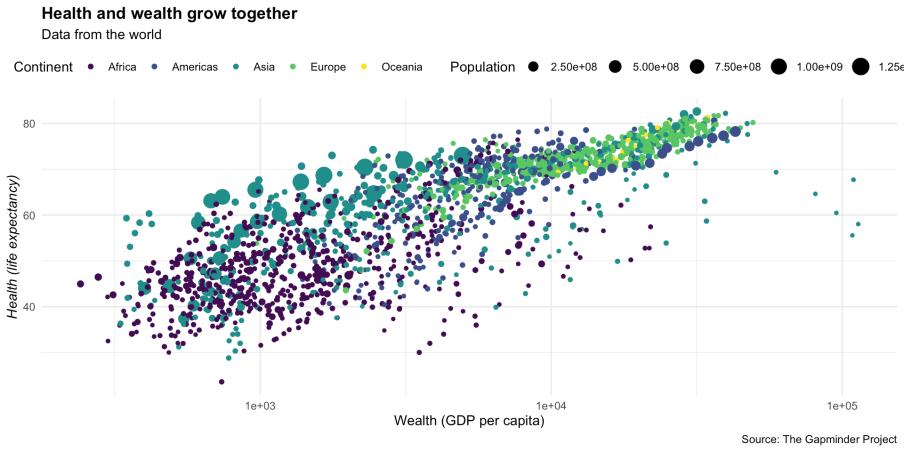


Theme options

Make theme adjustments with `theme()`

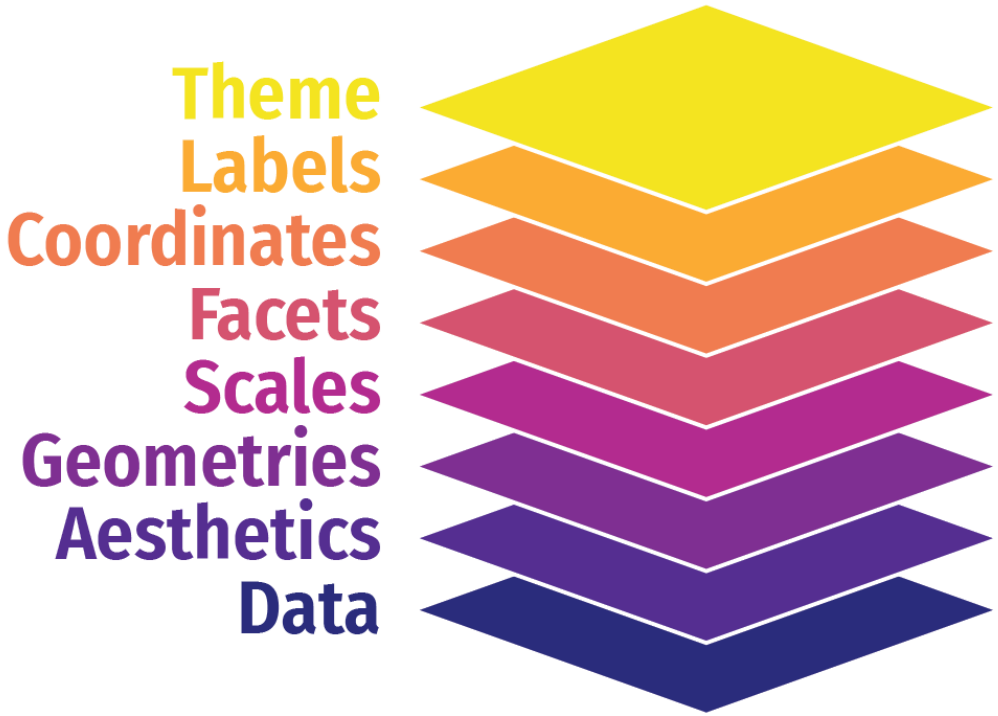
There are a billion options here!

```
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp, c
                     size = pop)) +
  geom_point() +
  scale_x_log10() +
  scale_color_viridis_d() +
  labs(title = "Health and wealth grow together",
       subtitle = "Data from the world",
       x = "Wealth (GDP per capita)",
       y = "Health (life expectancy)",
       color = "Continent",
       size = "Population",
       caption = "Source: The Gapminder Project")
  theme_minimal() +
  theme(legend.position = "top",
        plot.title = element_text(face = "bold"),
        axis.title.y = element_text(face = "italic
```



There are many, many more options

See the {ggplot2} documentation for complete examples of everything you can do



Your turn #1: untidy temperatures

Take this tibble (very similar to a `data.frame`) of temperature recordings at three stations on three dates:

```
temp_data_untidy <- tribble(
  ~date, ~station1, ~station2,  ~station3,
  "2023-10-01", 30.1, 29.8,  31.2,
  "2023-11-01", 28.6, 29.1,  33.4,
  "2023-12-01", 29.9, 28.5,  32.3
)
```

Imagine our goal is to track temperature across time.

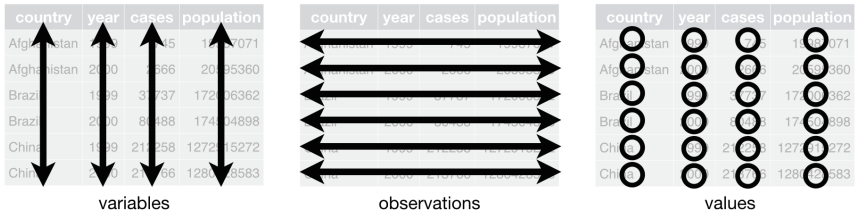
Your turn #1: untidy temperatures

06:00

1. What makes this data untidy? Describe.
2. Make a new data frame called `temp_data_tidy`. Use `pivot_longer()` to tidy the data and create a new `temperature` and `station` variable.
3. Make a plot that tracks the temperature changes over time for `station1` only. Use `filter()` to select the station and use `mutate()` in combination with the `as_date()` function to convert the date variable from character to a date format. into a date. Use `geom_line` for the plot.
4. Now use the the non-filtered data frame with all stations. Add another aesthetic layer to your previous plot, so that your new plot allows to differentiate temperature changes between the different stations. Tip: Use `color`

Your turn #1: untidy temperatures

1. What makes this data untidy? Describe.



1. Variables are columns
2. Observations are rows
3. Values are cells

date	station1	station2	station3
2023-10-01	30.1	29.8	31.2
2023-11-01	28.6	29.1	33.4
2023-12-01	29.9	28.5	32.3

Multiple observations (temperature recordings) per row

Your turn #1: untidy temperatures

2. Make a new data frame called `temp_data_tidy`. Use `pivot_longer()` to tidy the data and create a new `temperature` and `station` variable.

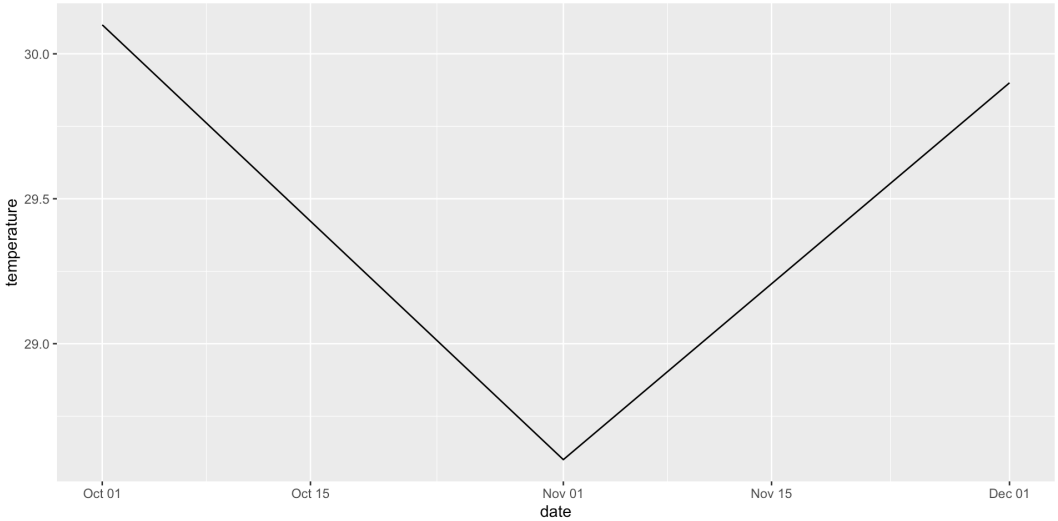
```
temp_data_tidy <- temp_data_untidy |>
  pivot_longer(cols = starts_with("station"),
               names_to = "station",
               values_to = "temperature")
```

date	station	temperature
2023-10-01	station1	30.1
2023-10-01	station2	29.8
2023-10-01	station3	31.2
2023-11-01	station1	28.6
2023-11-01	station2	29.1
2023-11-01	station3	33.4
2023-12-01	station1	29.9
2023-12-01	station2	28.5

Your turn #1: untidy temperatures

3. Make a plot that tracks the temperature changes over time for `station1` only. Use `filter()` to select the station and use `mutate()` in combination with the `as_date()` function to convert the date variable from character to a date format. into a date. Use `geom_line` for the plot.

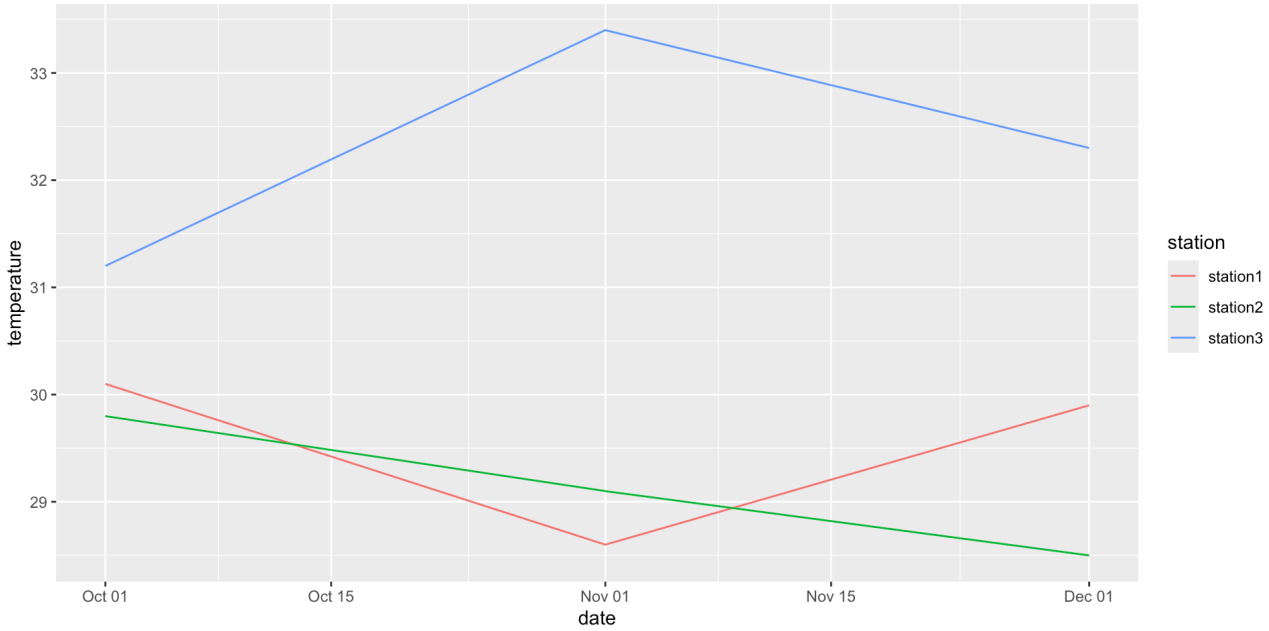
```
temp_data_tidy |>
  filter(station == "station1") |>
  mutate(date = as_date(date)) |>
  ggplot(aes(x = date, y = temperature)) +
  geom_line()
```



Your turn #1: untidy temperatures

4. Now use the the non-filtered data frame with all stations. Add another aesthetic layer to your previous plot, so that your new plot allows to differentiate temperature changes between the different stations. Tip: Use `color`

```
temp_data_tidy |>
  mutate(date = as_date(date)) |>
  ggplot(aes(x = date, y = temperature, color = station)) +
  geom_line()
```



That’s it for today :)