Graphics in R

using `ggplot`

Understanding Political Numbers

Feb 13, 2019

Get started

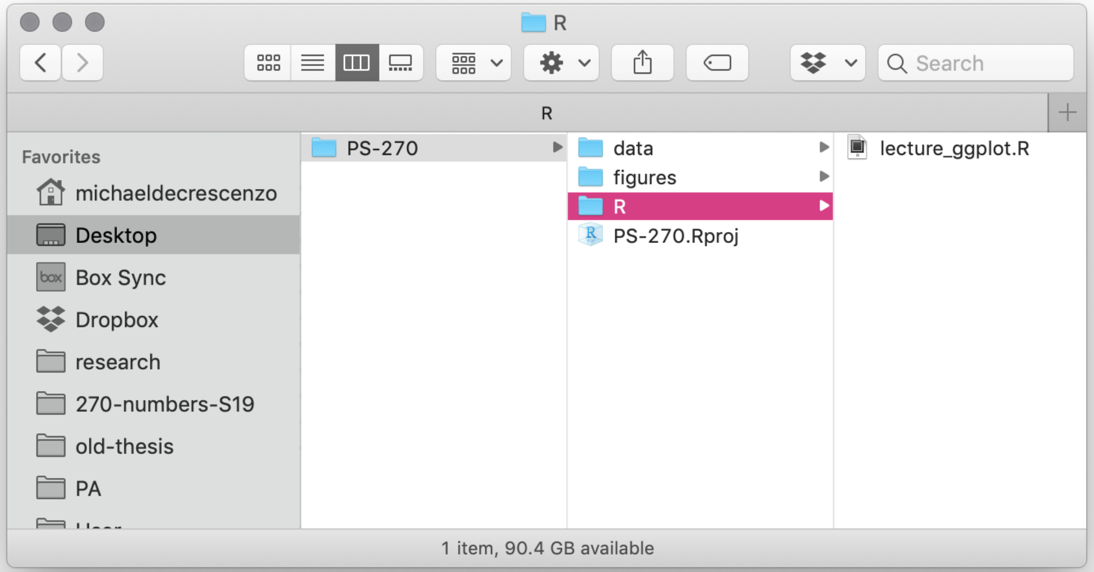
In your `PS-270` folder on your computer, double-click `.Rproj` file to open RStudio.

Create folders within `PS-270`

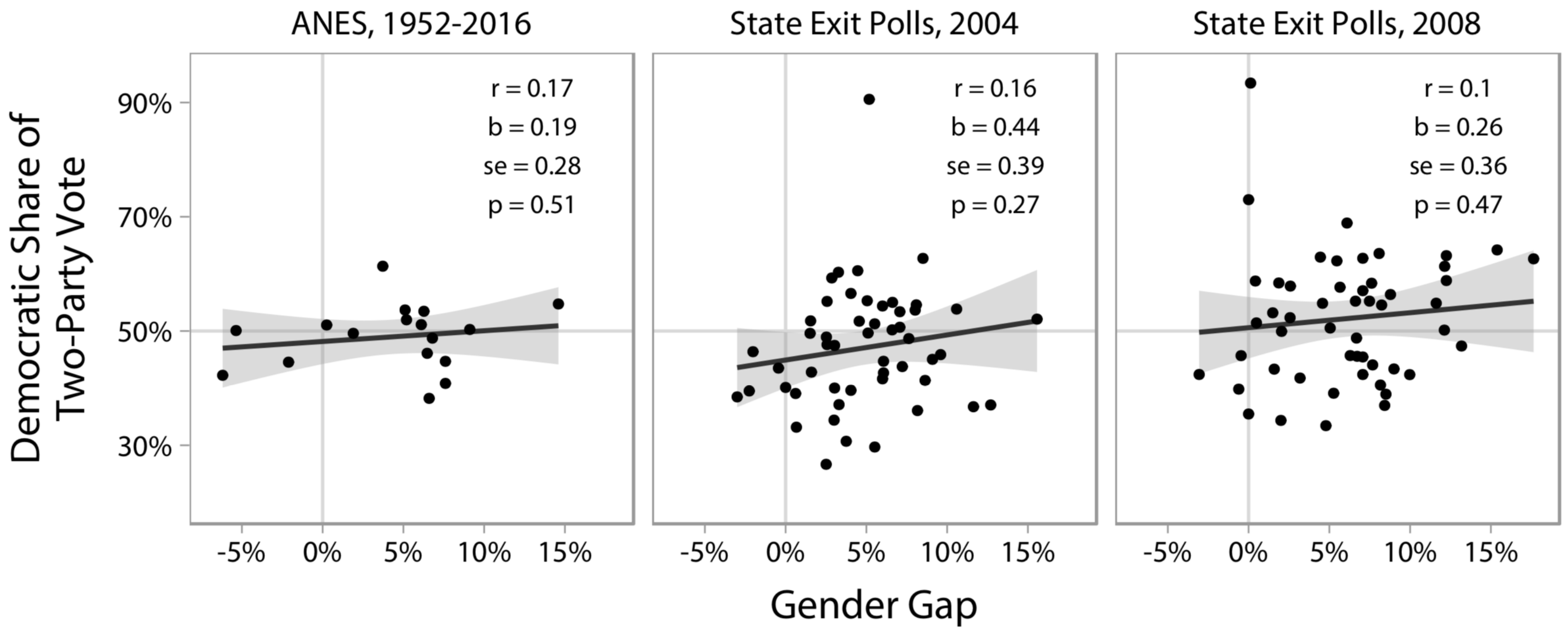
- `R` folder, for `R` script files
- `figures` folder, for saving figures
- `data` folder, for data

On Canvas: Download `lecture_ggplot.R` and save to `R` folder

In Rstudio: open `lecture_ggplot.R`



What is `ggplot`?



A plotting system for R

Originally its own package—bundled into the `{tidyverse}` package

`gg` for "Grammar of Graphics"

What is a "grammar" of graphics?

Data: your data frame

Aesthetic mapping: how data (variables) become plot attributes (axes, color, sizes)

Scales: modifying the *mapping* from data to plot (customizing axes, colors, sizes)

Geoms: geometric representations of data (points, lines, bars)

Facets: sub-panels in the plot

Coordinates: features of the coordinate system (orientation...)

What is a "grammar" of graphics?

Data: your data frame

Aesthetic mapping: how data (variables) become plot attributes (axes, color, sizes)

Scales: modifying the *mapping* from data to plot (customizing axes, colors, sizes)

Geoms: geometric representations of data (points, lines, bars)

Facets: sub-panels in the plot

Coordinates: features of the coordinate system (orientation...)

Why `ggplot` is great

- Premise: the grammar describes attributes of most (all?) graphics
- Premise: `ggplot` functions manipulate the grammar
- Conclusion: make *lots* of graphics using same basic tools

Get oriented

Load packages:

```r
# {tidyverse} contains ggplot tools
library("tidyverse")

install.packages("gapminder") # contains dataset
install.packages("here")      # easier to save things
library("gapminder")
library("here")
```

The `lecture_ggplot.R` file already contains code that I wrote. You can...

Execute the code as-is.

- Mac: `Command + Enter`
- Windows: `Ctrl + Enter`
- Tip: more RStudio keyboard shortcuts

Even better: re-type commands to familiarize yourself w/ the flow

- Use my code if something isn't working

Meet the data

```r
# print the data
gapminder

# variable names
names(gapminder)
```

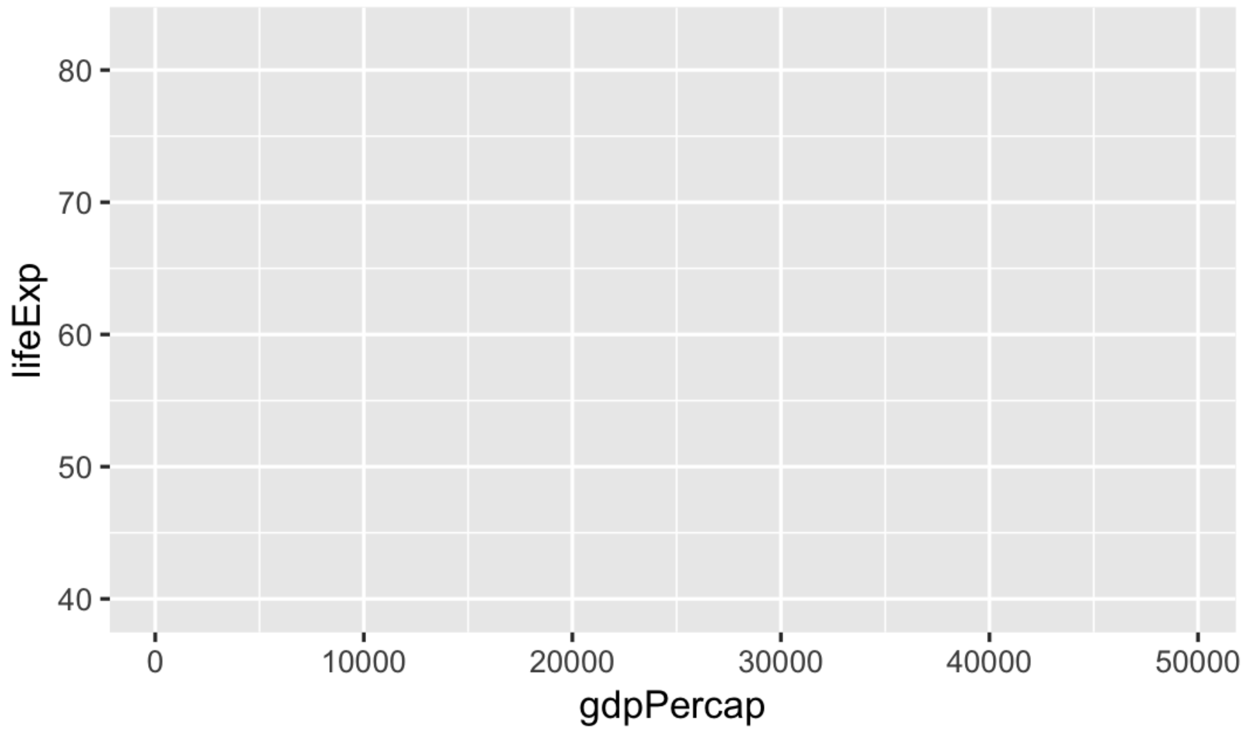
Different subsets of data will let us highlight different graphics capabilities

```r
# create a new object: 'gapminder' for most recent year
gap07 <- filter(gapminder, year == 2007)

# create a new object: the subset of 'gapminder' where continent is "Oceania"
gapOC <- filter(gapminder, continent == "Oceania")

# What's left?
gap07
gapOC
```

Let's make a graph

Start a plot

```r
ggplot(data = gap07, mapping = aes(x = gdpPercap, y = lifeExp))
```



Let's break it down

```r
ggplot(data = gap07,
       mapping = aes(x = gdpPercap, y = lifeExp))
```

Grammar: data. First, Declare the dataset where `ggplot` can find your data.

Let's break it down

```
ggplot(data = gap07,
       mapping = aes(x = gdpPercap, y = lifeExp))
```

Grammar: data. First, Declare the dataset where `ggplot` can find your data.

Grammar: aesthetic mapping. Tell `ggplot` which variables to look at for plot-relevant data

- Grab *axis* information from the `gdpPercap` and `lifeExp` variables.
- If we want aesthetics (axes, point shape, color, line style) to correspond to variables, must use the `aes()` function.

Let's break it down

```
ggplot(data = gap07,
       mapping = aes(x = gdpPercap, y = lifeExp))
```

Grammar: data. First, Declare the dataset where `ggplot` can find your data.

Grammar: aesthetic mapping. Tell `ggplot` which variables to look at for plot-relevant data

- Grab *axis* information from the `gdpPercap` and `lifeExp` variables.
- If we want aesthetics (axes, point shape, color, line style) to correspond to variables, must use the `aes()` function.

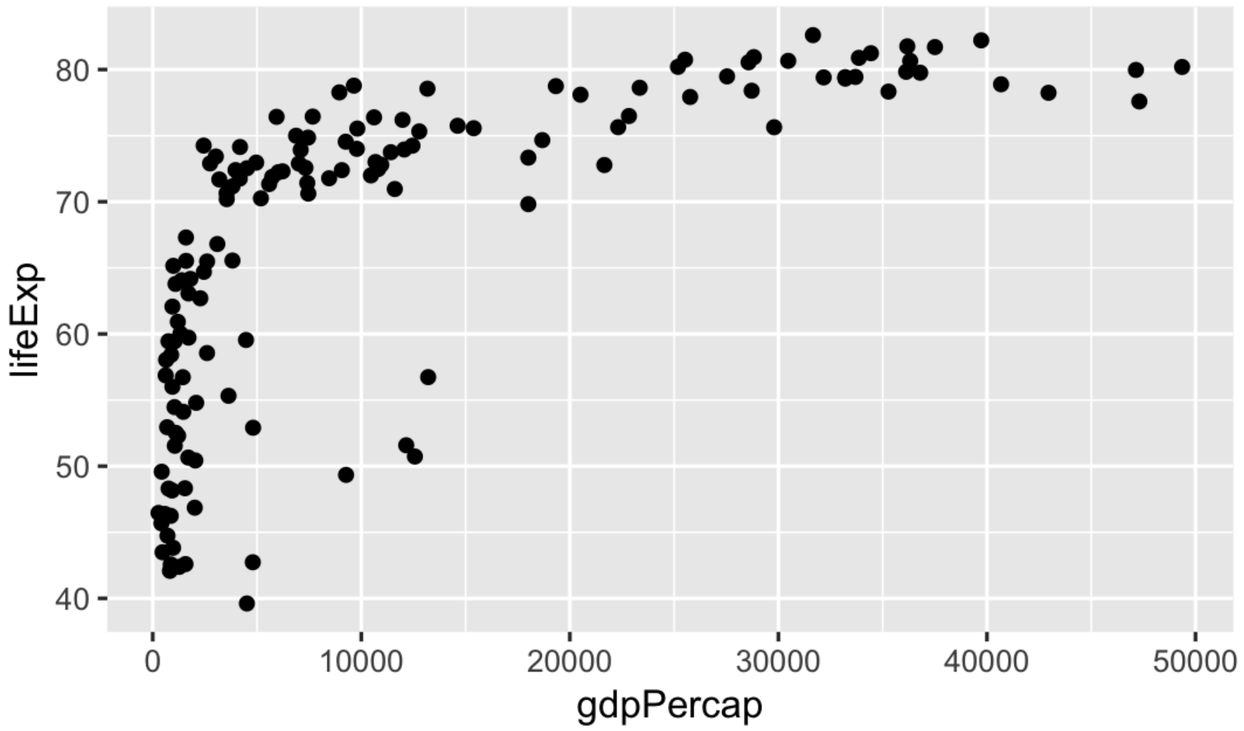
(Browse the Tidyverse Style Guide)

Geoms

```
ggplot(data = gap07, mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point()
```

Add components with `+`

Grammar: geoms. Functions take the form `geom_*()`.



More aesthetics

```
ggplot(data = gap07, mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point(aes(color = continent))
```

Grammar: aesthetics. How does data become a plot feature?

Translation: "I want different colors for each continent"

Which aesthetics can be modified? Check the help file: `?geom_point`

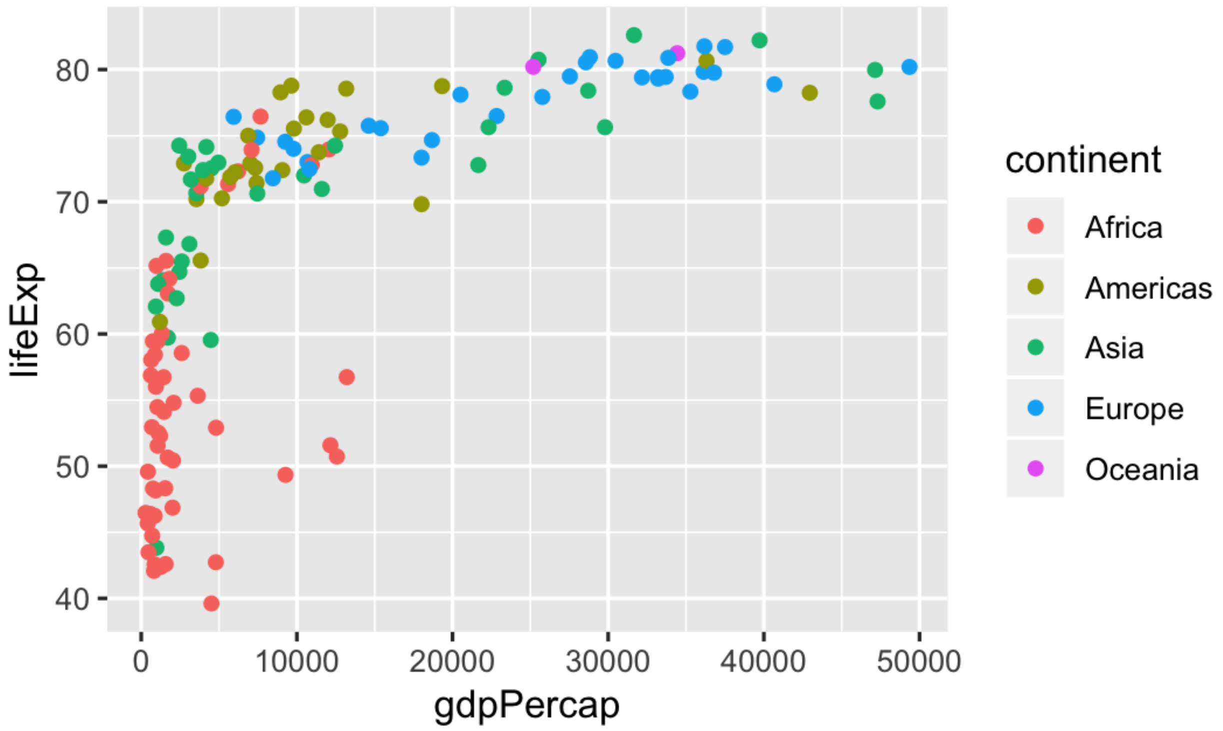
More aesthetics

```r
ggplot(data = gap07, mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point(aes(color = continent))
```

Grammar: aesthetics. How does data become a plot feature?

Translation: "I want different colors for each continent"

Which aesthetics can be modified? Check the help file: `?geom_point`

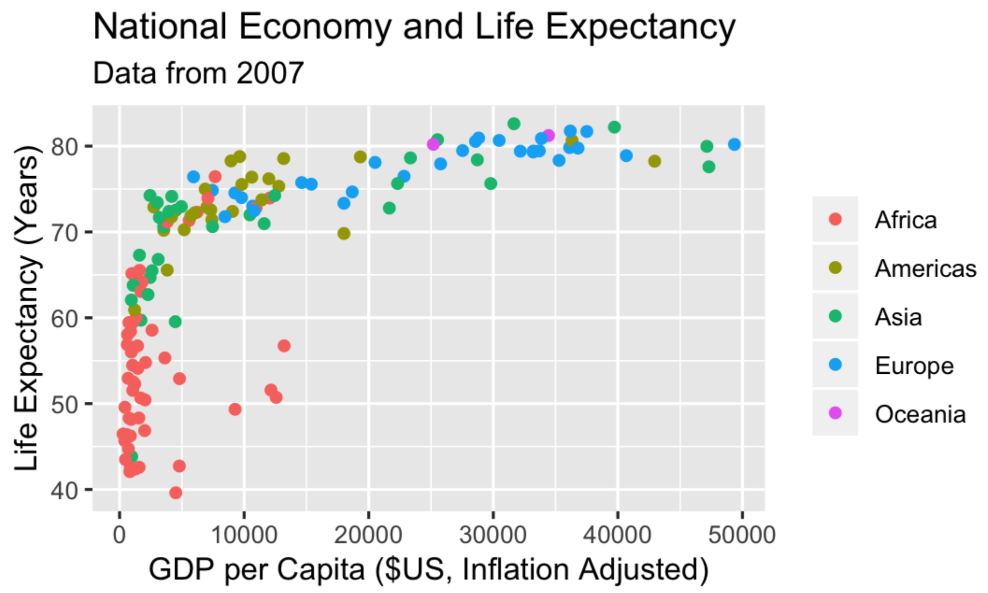


Labels (and saving)

```r
ggplot(data = gap07, mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point(aes(color = continent)) +
  labs(x = "GDP per Capita ($US, Inflation Adjusted)",
       y = "Life Expectancy (Years)",
       color = NULL,
       title = "National Economy and Life Expectancy",
       subtitle = "Data from 2007")
```

save the plot! (inside the `figures` folder)

```r
ggsave(here("figures", "my-plot.pdf"),
       height = 3, width = 5)
```



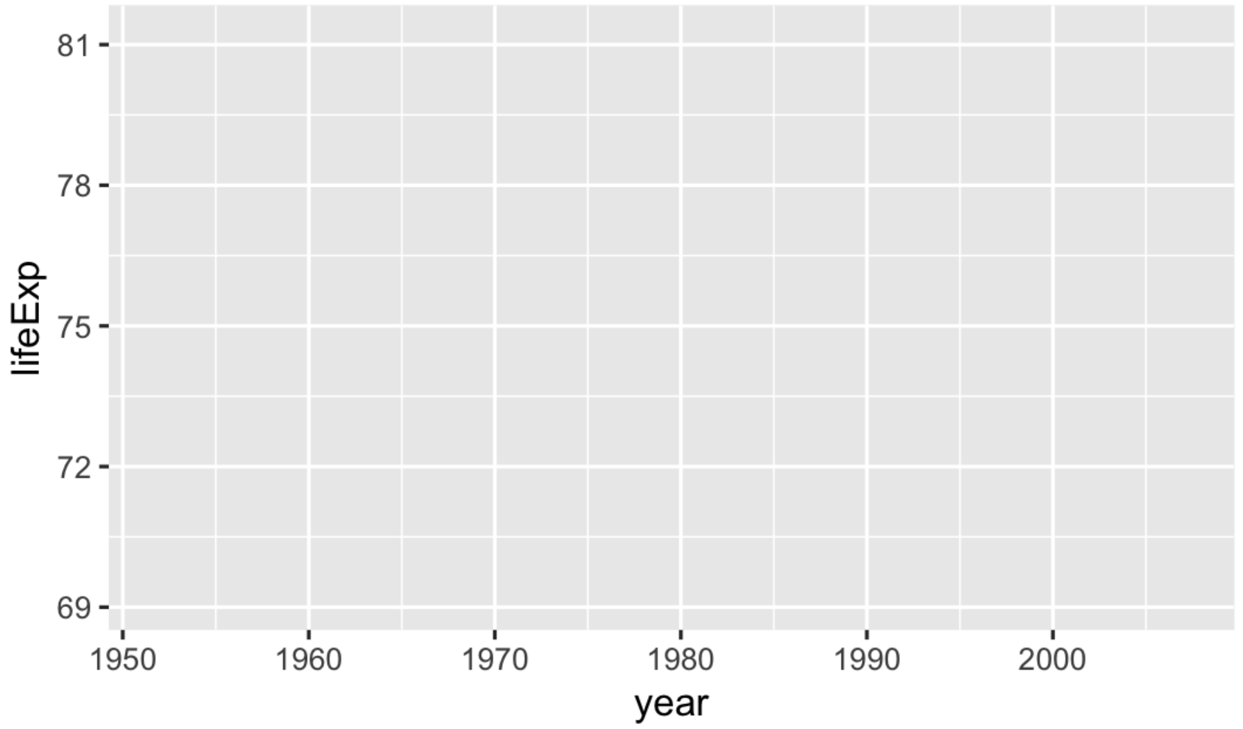
New Plot

New plot, using Oceania data

```
ggplot(gapOC, aes(x = year, y = lifeExp))
```

`data =` and `mapping =` are implied

We want a line for each country...what do we do?



Lines

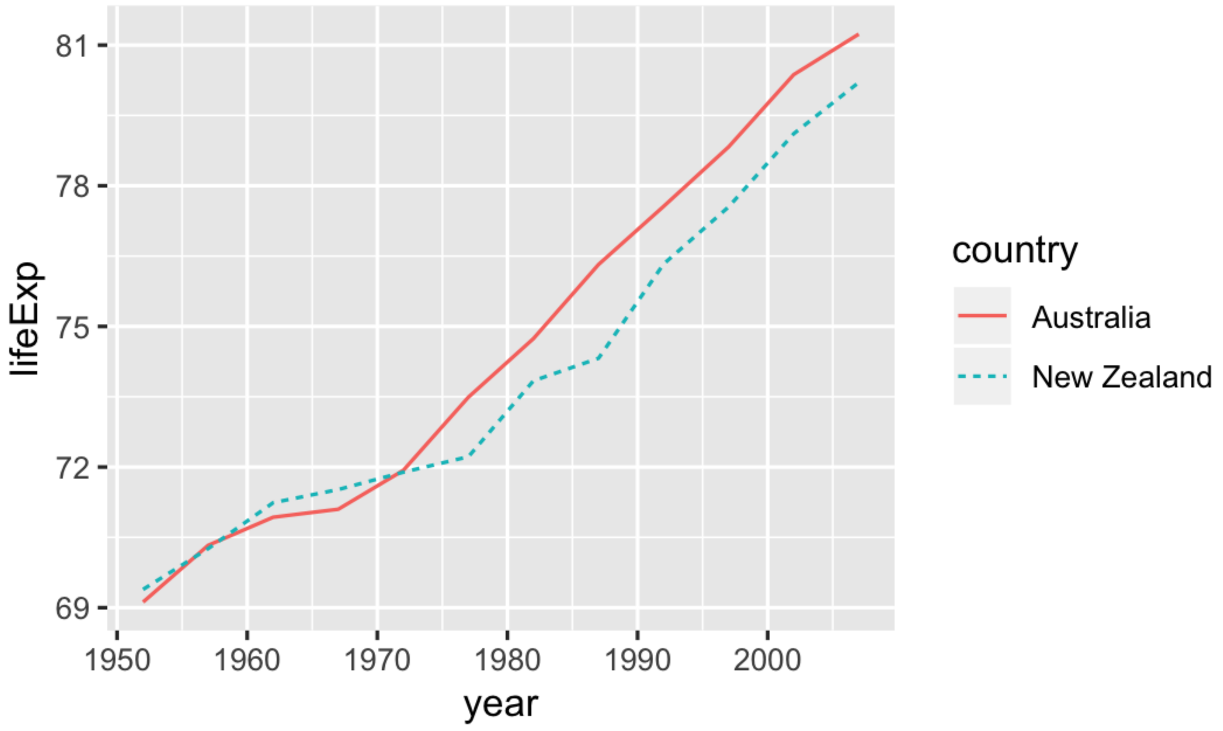
```
ggplot(gapOC, aes(x = year, y = lifeExp)) +
  geom_line(aes(linetype = country,
                color = country))
```

This dataset contains two countries.

```
# count() tabulates a variable
count(gapOC, country)
```

```
## # A tibble: 2 x 2
##   country         n
##   <fct>       <int>
## 1 Australia      12
## 2 New Zealand    12
```

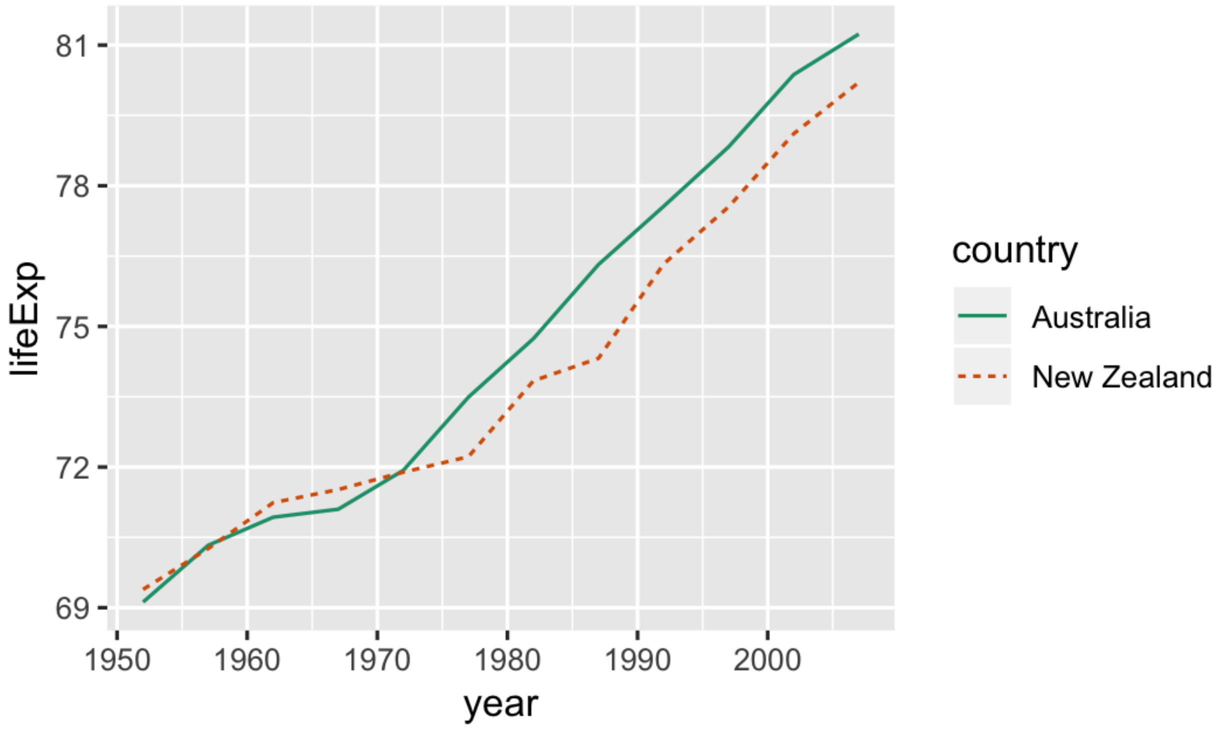
`geom_line()` needs `aes(something = country)` or else we get only one line



Scales (they modify default aesthetics)

```r
ggplot(gapOC, aes(x = year, y = lifeExp)) +
  geom_line(aes(linetype = country, color = country)) +
  scale_color_brewer(palette = "Dark2")
```

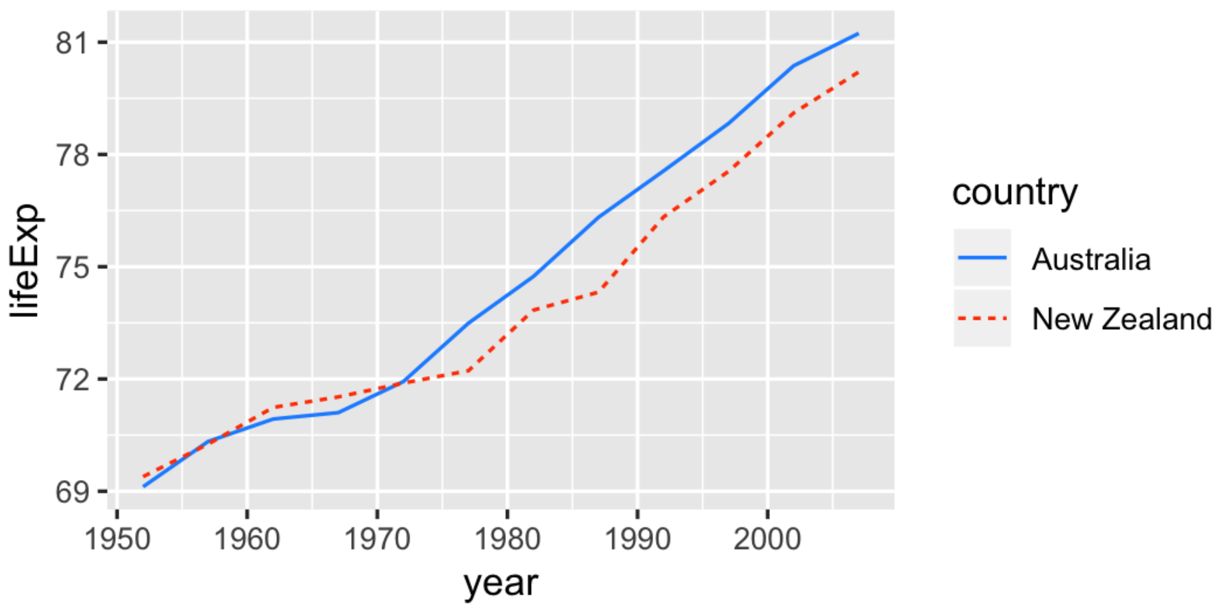
Variants include `scale_color_brewer()`, `scale_color_manual()`, `scale_color_continuous()`



Scales (they modify default aesthetics)

```
ggplot(gapOC, aes(x = year, y = lifeExp)) +
  geom_line(aes(linetype = country, color = country)) +
  scale_color_manual(values = c("Australia" = "dodgerblue",
                                "New Zealand" = "orangered"))
```

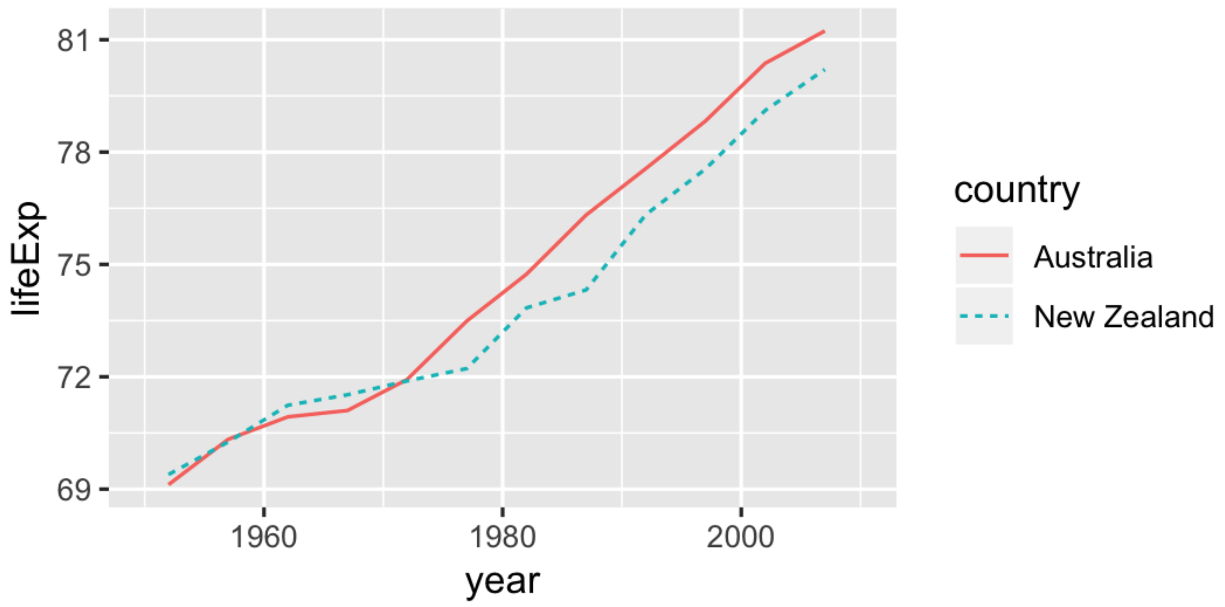
All scale functions:
`scale_aesName_modifier()`



Coordinates

```
ggplot(gapOC, aes(x = year, y = lifeExp)) +
  geom_line(aes(linetype = country, color = country)) +
  coord_cartesian(xlim = c(1950, 2010))
```

This changed the x axis. How could I customize that?



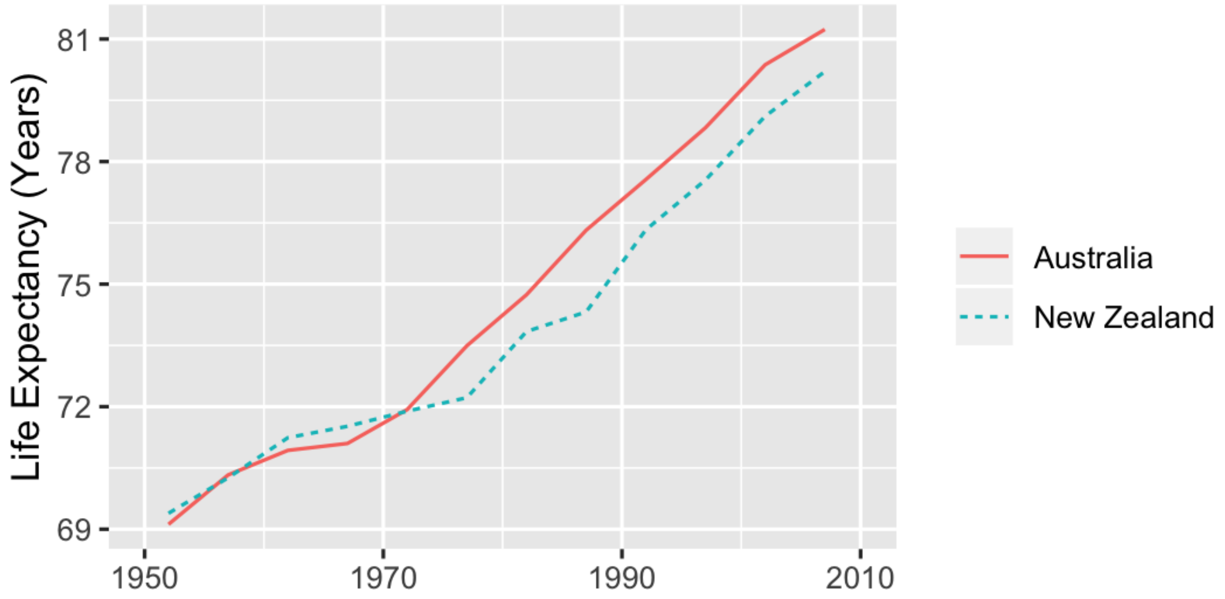
Coordinates

```
ggplot(gapOC, aes(x = year, y = lifeExp)) +
  geom_line(aes(linetype = country, color = country)) +
  coord_cartesian(xlim = c(1950, 2010)) +
  scale_x_continuous(breaks = seq(1950, 2010, 20)) +
  labs(x = NULL, y = "Life Expectancy (Years)",
       color = NULL, linetype = NULL)
```

`x` is an aesthetic, and you mapped it from data.

So… you modify the default use a scale function.

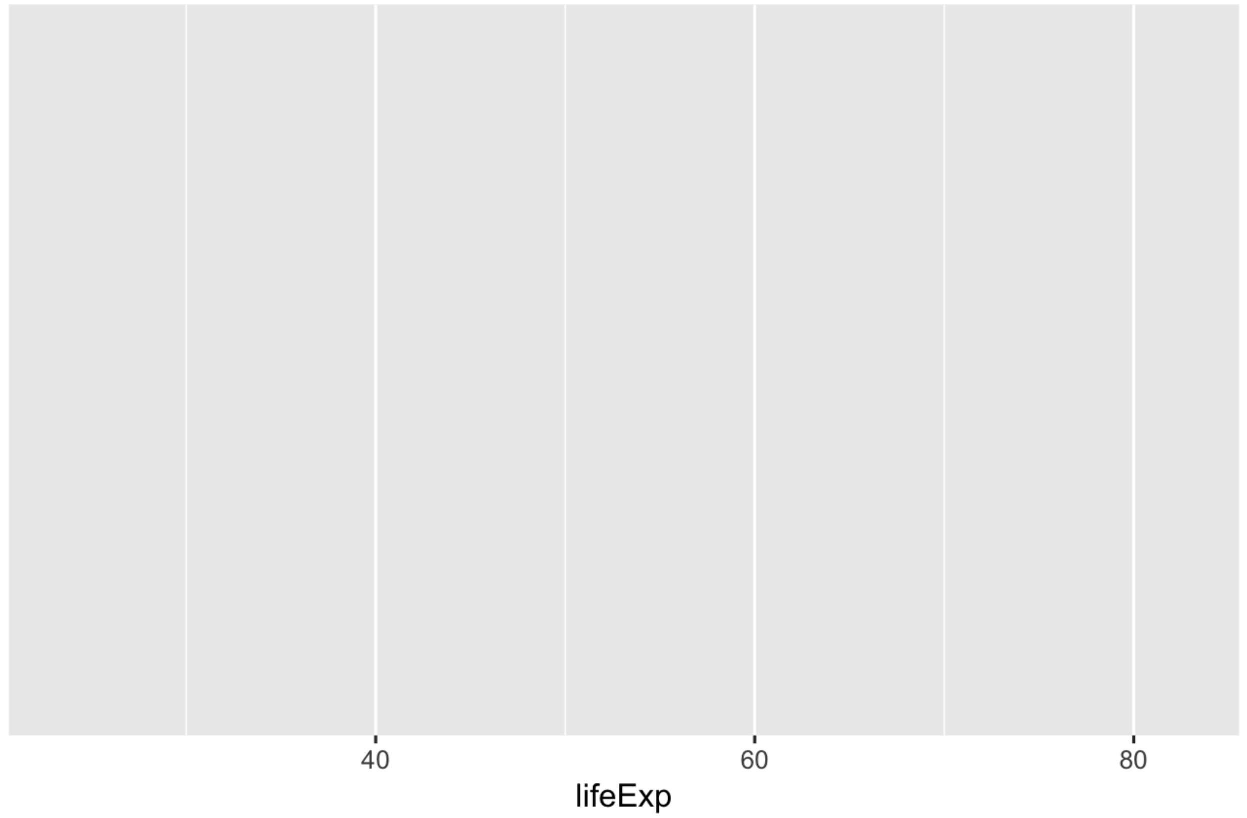
`coord_flip()` flips the horizontal and vertical axes



One Last Plot

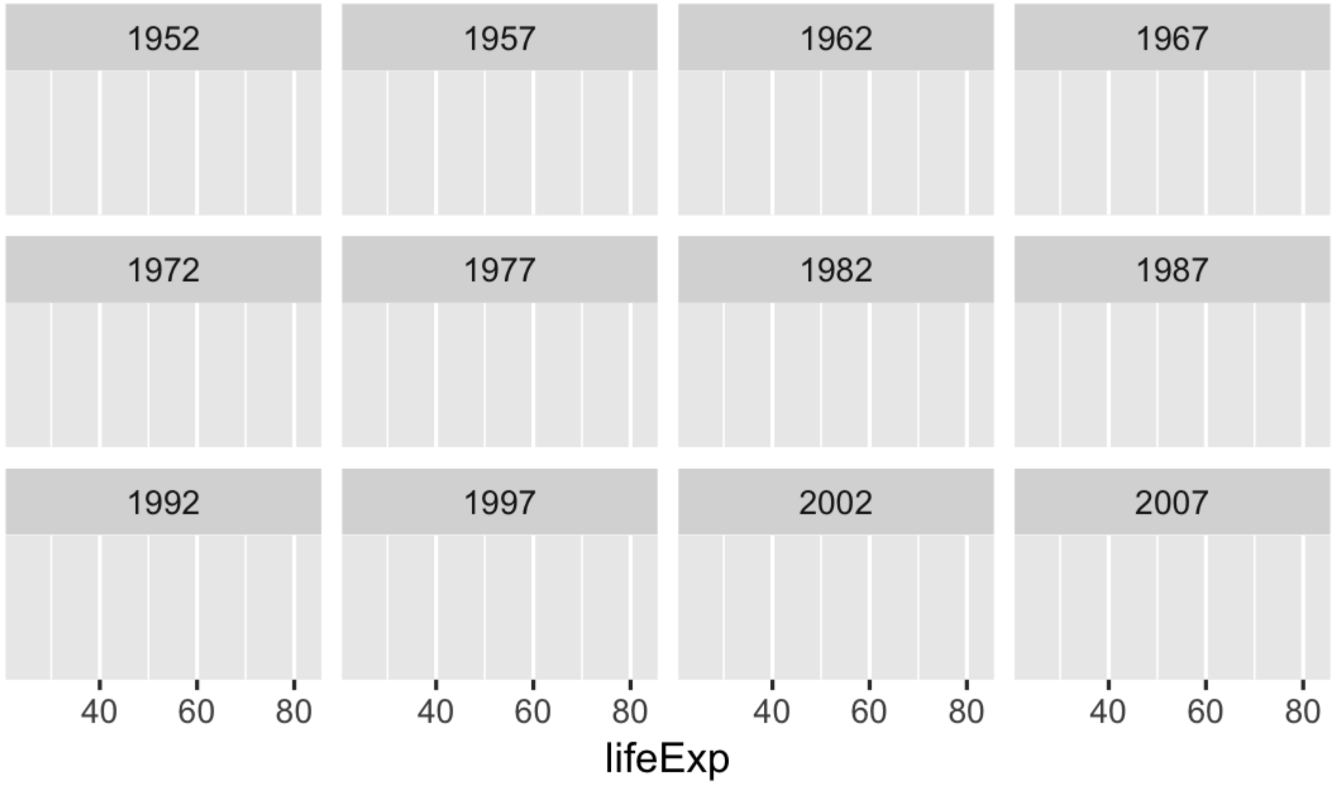
Using the full gapminder data

```r
ggplot(gapminder, aes(x = lifeExp))
```



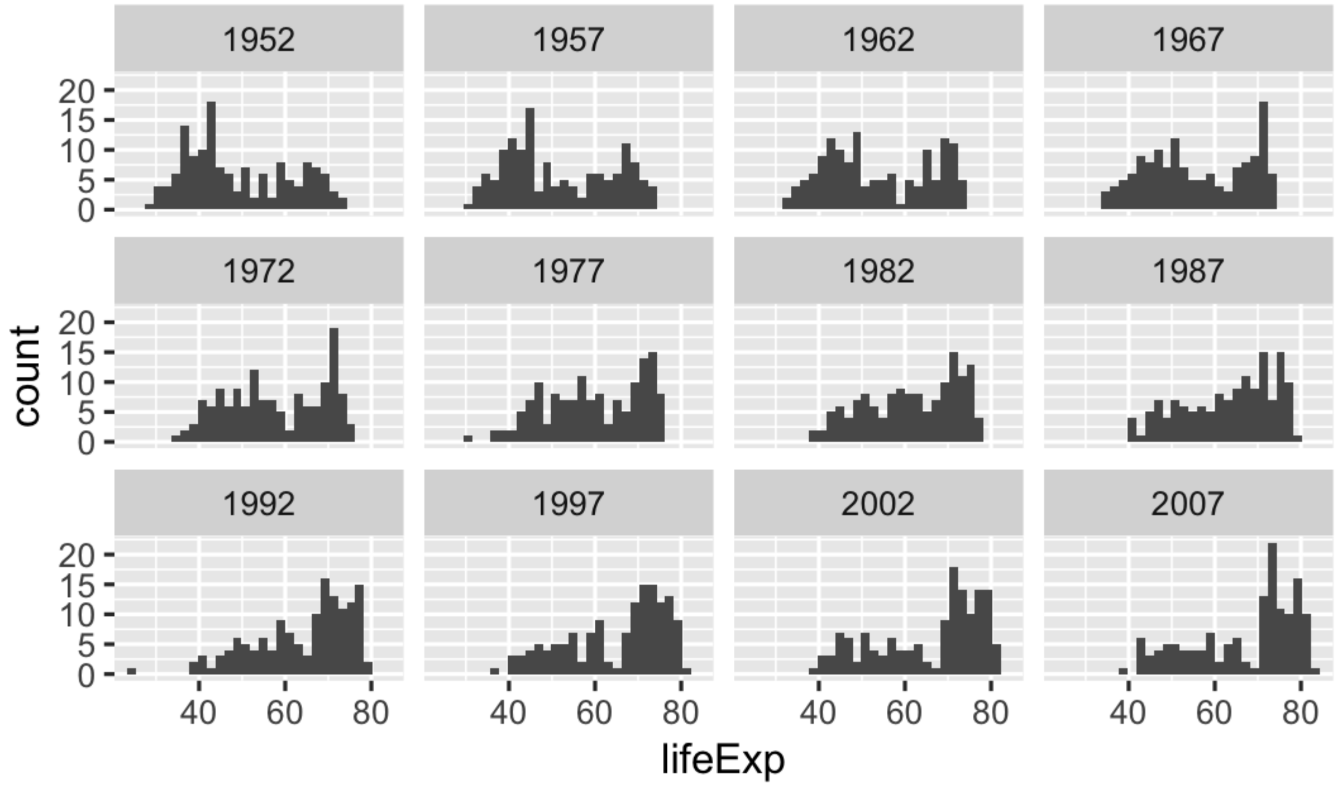
Facets: wrapping

```r
ggplot(gapminder, aes(x = lifeExp)) +
  facet_wrap(~ year) # tilde is necessary
```



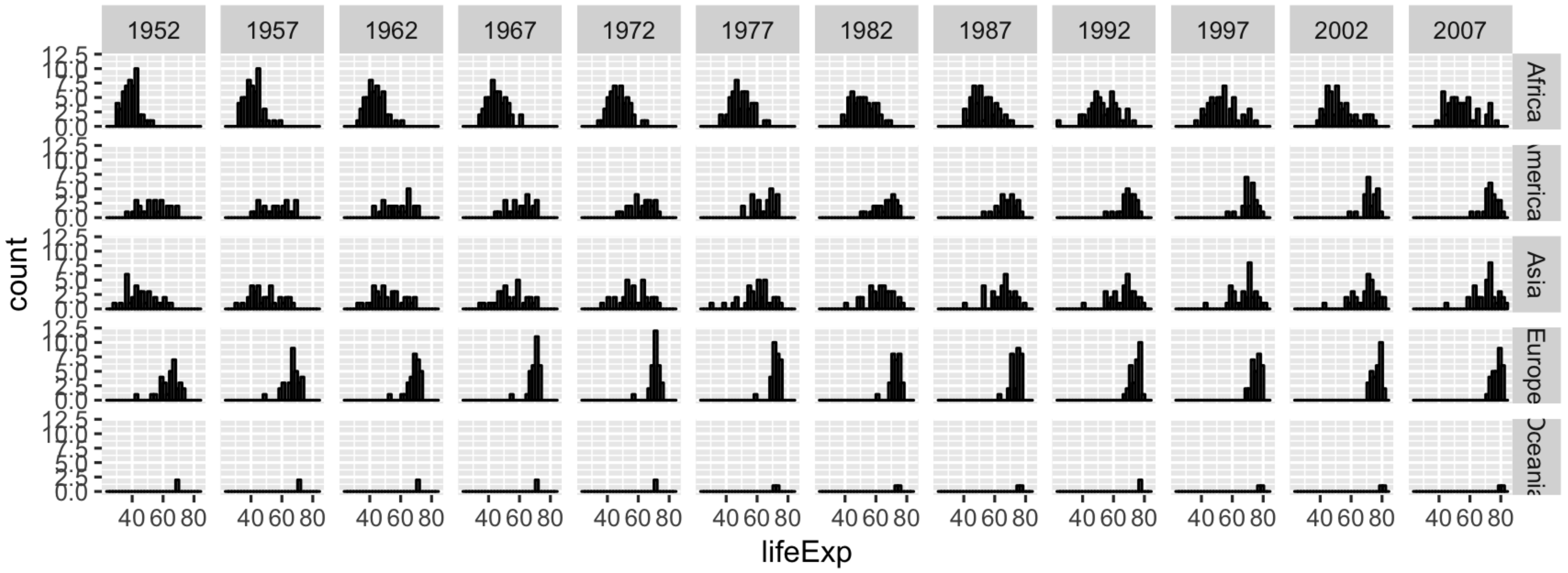
Facets: wrapping

```
ggplot(gapminder, aes(x = lifeExp)) +
  facet_wrap(~ year) +
  geom_histogram()
```



Facets: grid

```
ggplot(gapminder, aes(x = lifeExp)) +
  facet_grid(continent ~ year) +
  geom_histogram(fill = "white", color = "black")
```



Exercise 1

What's next?

In section: more plots

- including histograms (necessary for homework)

Next week: signal and noise

- mean and variance
- more tools for R