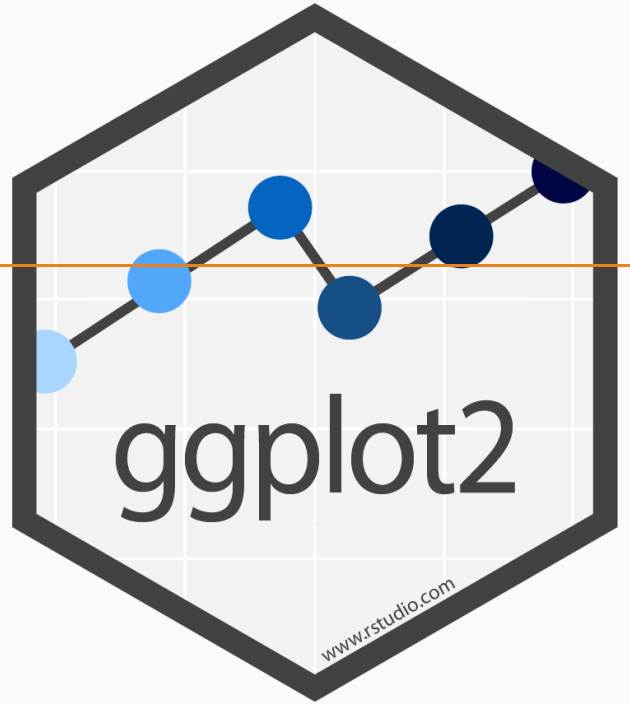
Data Visualisation in R

Introduction to ggplot2 and leaflet

Dr. Laurie Baker
Data Science Campus
2020/04/09 (updated: 2020-04-17)



Introduction to ggplot2

What we'll cover today.

- Brief intro to the theory behind `ggplot2` and the "grammar of graphics".
- Building a plot using `ggplot`.
- Building interactives maps using leaflet.
- Slides and code adapted from Garrick Aden-Buie "Gentle ggplot2 tutorial" on GitHub: http://github.com/gadenbuie/gentle-ggplot2

How would you draw a line graph by hand?

country	year	pop
Chile	1997	14.60
Chile	2002	15.50
Chile	2007	16.30
Rwanda	1997	7.21
Rwanda	2002	7.85
Rwanda	2007	8.86
Syria	1997	15.10
Syria	2002	17.20
Syria	2007	19.30

1. Draw the axes, add tick marks.
2. Draw each line. Colour by country.
3. Add the axes labels.
4. Add a title.
5. Add a legend.

How would you draw a bar graph by hand?

country	year	pop
Chile	1997	14.60
Chile	2002	15.50
Chile	2007	16.30
Rwanda	1997	7.21
Rwanda	2002	7.85
Rwanda	2007	8.86
Syria	1997	15.10
Syria	2002	17.20
Syria	2007	19.30

1. Draw the axes, add tick marks.
2. **Draw each bar.** Colour by country.
3. Add the axes labels.
4. Add a title.
5. Add a legend.

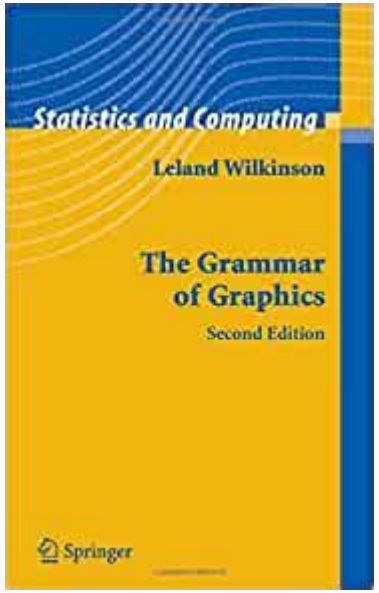
How would you draw a graph?

country	year	pop
Chile	1997	14.60
Chile	2002	15.50
Chile	2007	16.30
Rwanda	1997	7.21
Rwanda	2002	7.85
Rwanda	2007	8.86
Syria	1997	15.10
Syria	2002	17.20
Syria	2007	19.30

1. What decisions did you make?
2. How did the data inform them?
3. Did you look at the values to decide the axes and tick marks?
4. How did you decide the labels?

How would you draw a graph?

The grammar of graphics

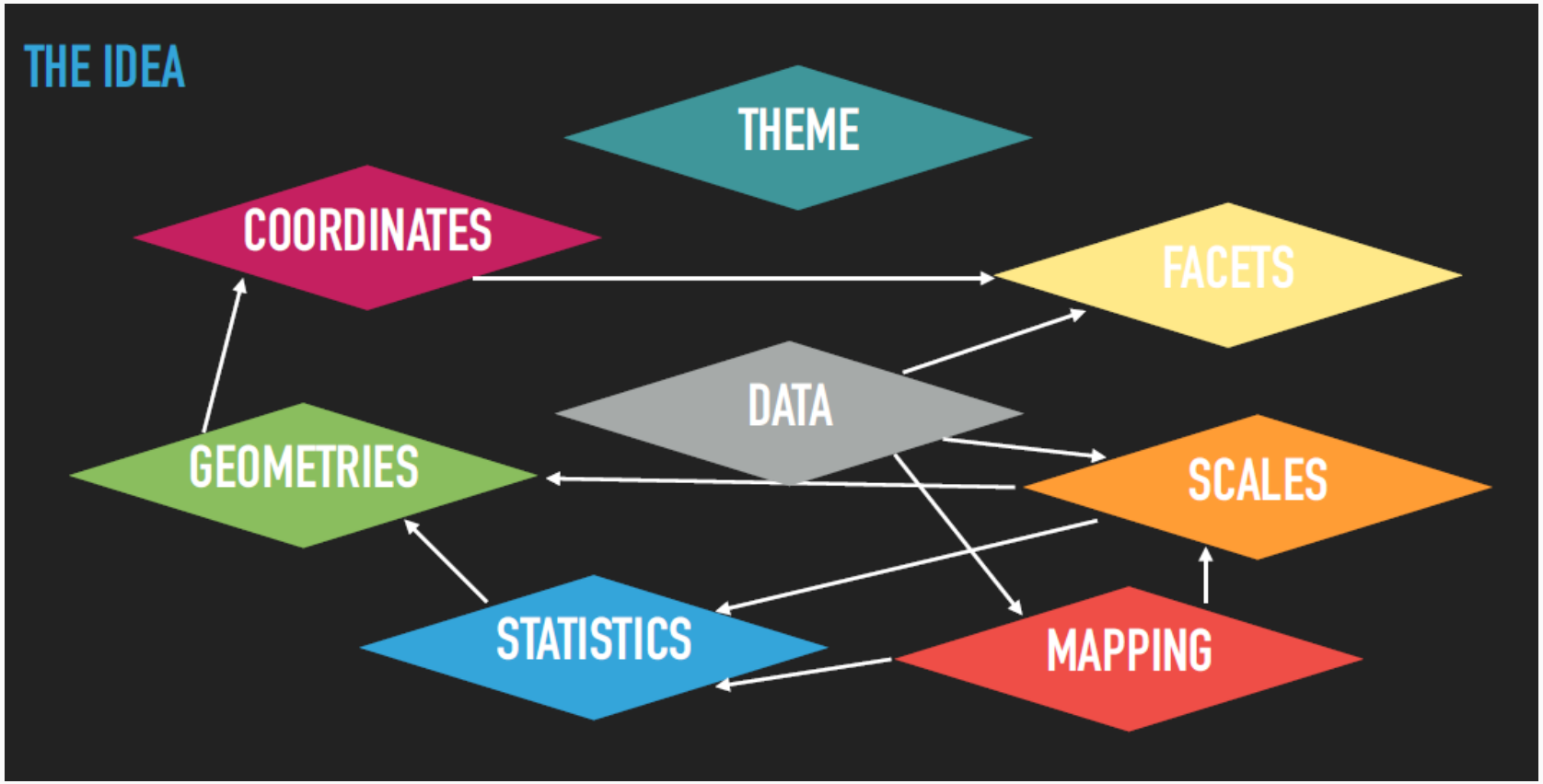


Grammar of Graphics

- Computers also follow steps.
- The **grammar of graphics =** rules/steps for plotting.
- First published in 1999.
- Breaks down graphics into its constituent parts.
- Focus on the relationship between the **variables** and the *visual properties* of the graph (e.g. *colour* = **country**).
- Foundation for **ggplot2**, **tableau**, **vegalite** etc.

The grammar of graphics

Links between variables and visual properties of the graph.



Thomas Lin Pedersen: Plot Anything With GGPLOT2

What is *ggplot2*?

Hadley Wickham

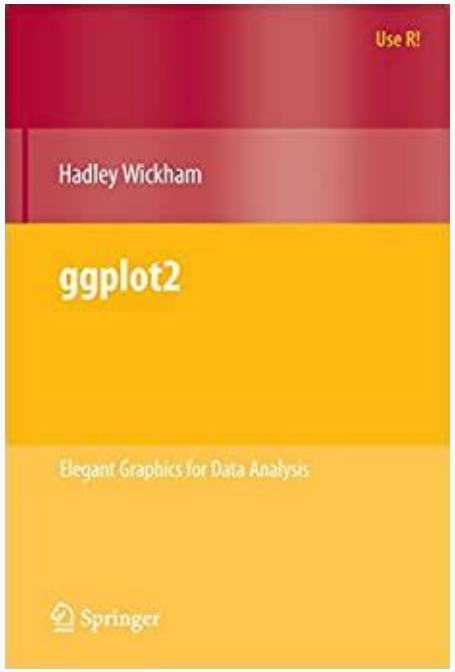
- **gg**plot2 is the implementation of the **g**rammar of **g**raphics in R with some adaptations.
- ..."a powerful way of thinking about visualisation, as a way of **mapping between variables and the visual properties of geometric objects** that you can perceive."

http://disq.us/p/sv640d

Why use *ggplot2*?

- Package for functional data visualization.
    1. Wrangle data
    2. Map data to visual elements
    3. Tweak scales, guides, axis, labels, theme
- Once you know the syntax it is easy to
    - Reason about how data drives visualization
    - Iterate to create multiple visualizations
    - Be consistent in the visualizations you make.

Learning objectives



ggplot2

- `ggplot2` is a huge package with lots of options, but it's well documented and organized.
- We'll cover a lot, but won't have time to go into every specific.
- The aim is to **equip** you with **where** and **what** to look for.

Getting started

Option 1: install the metapackage tidyverse

```
install.packages('tidyverse')
```

Option 2: install just `ggplot2`

```
install.packages('ggplot2')
```

Getting started

Load the tidyverse

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------------------------------------ t
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Getting started

Other packages you'll need for this adventure

- gapminder dataset from the `gapminder` package by Jenny Bryan.

```
## install.packages("gapminder")
library(gapminder)
```

gg is for
Grammar of Graphics

Every plot starts with data

MPG Ratings of Cars

- Manufacturer
- Car Type (Class)
- City MPG
- Highway MPG

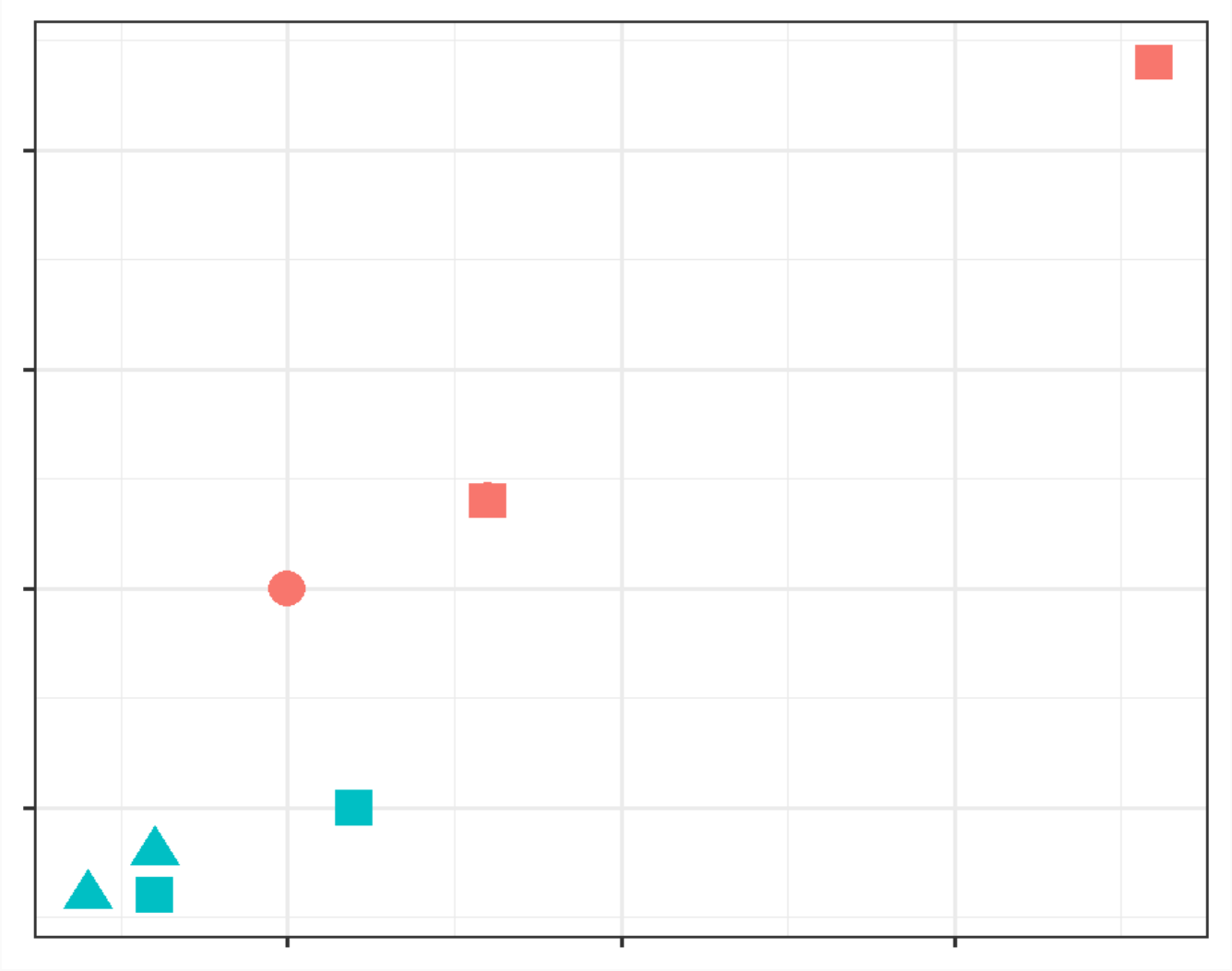
manufacturer	class	cty	hwy	model
audi	compact	18	27	a4
audi	compact	15	25	a4 quattro
ford	suv	12	18	expedition 2wd
ford	suv	13	19	explorer 4wd
toyota	suv	16	20	4runner 4wd
toyota	compact	18	27	camry solara
toyota	compact	28	37	corolla
toyota	suv	13	18	land cruiser wagon 4wd

Guess the data behind this plot?

MPG Ratings of Cars

- Manufacturer
- Car Type (Class)
- City MPG
- Highway MPG

What variable is represented by point shape?

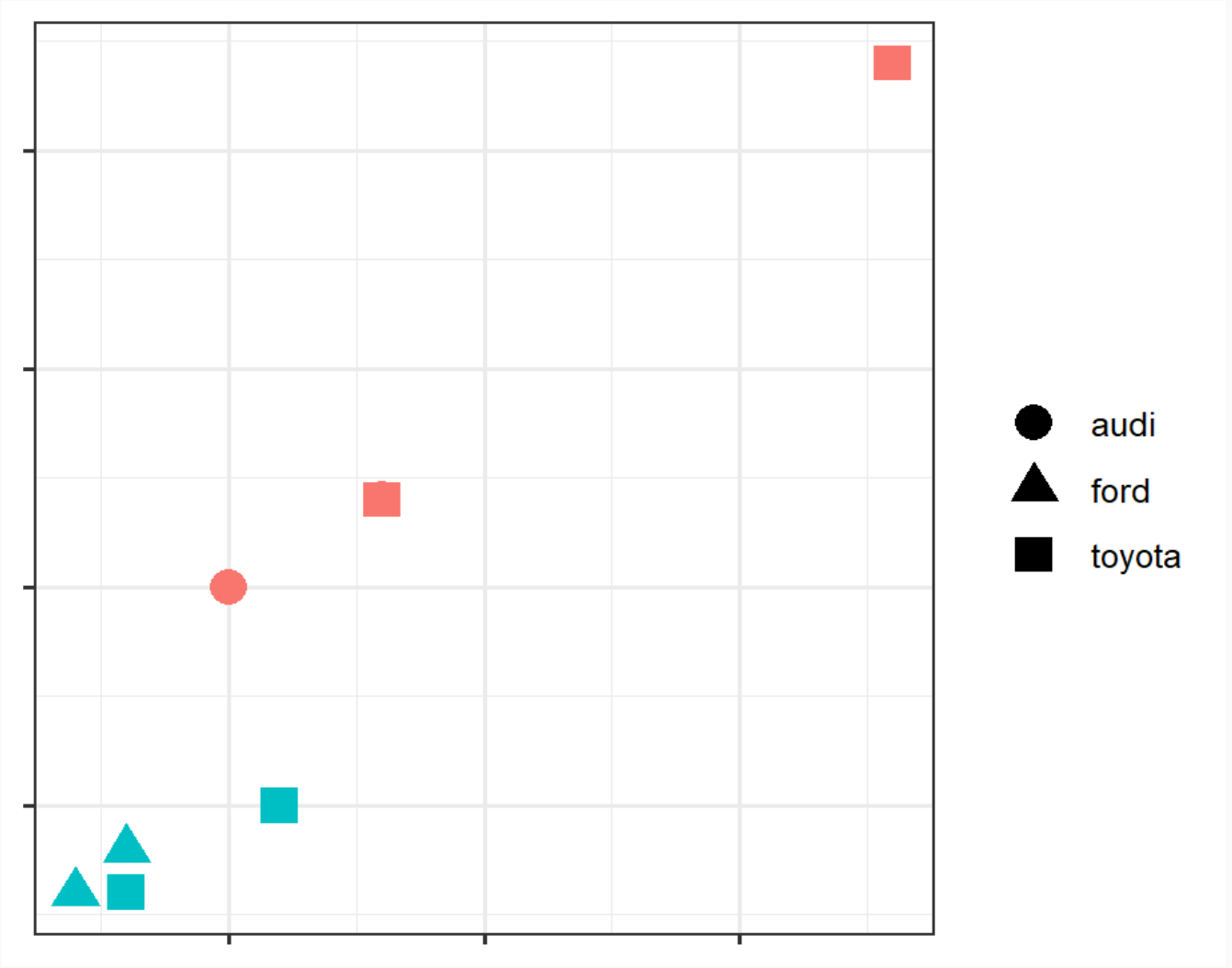


Guess the data behind this plot?

MPG Ratings of Cars

- **Manufacturer**
- Car Type (Class)
- City MPG
- Highway MPG

What variable is represented by colour?

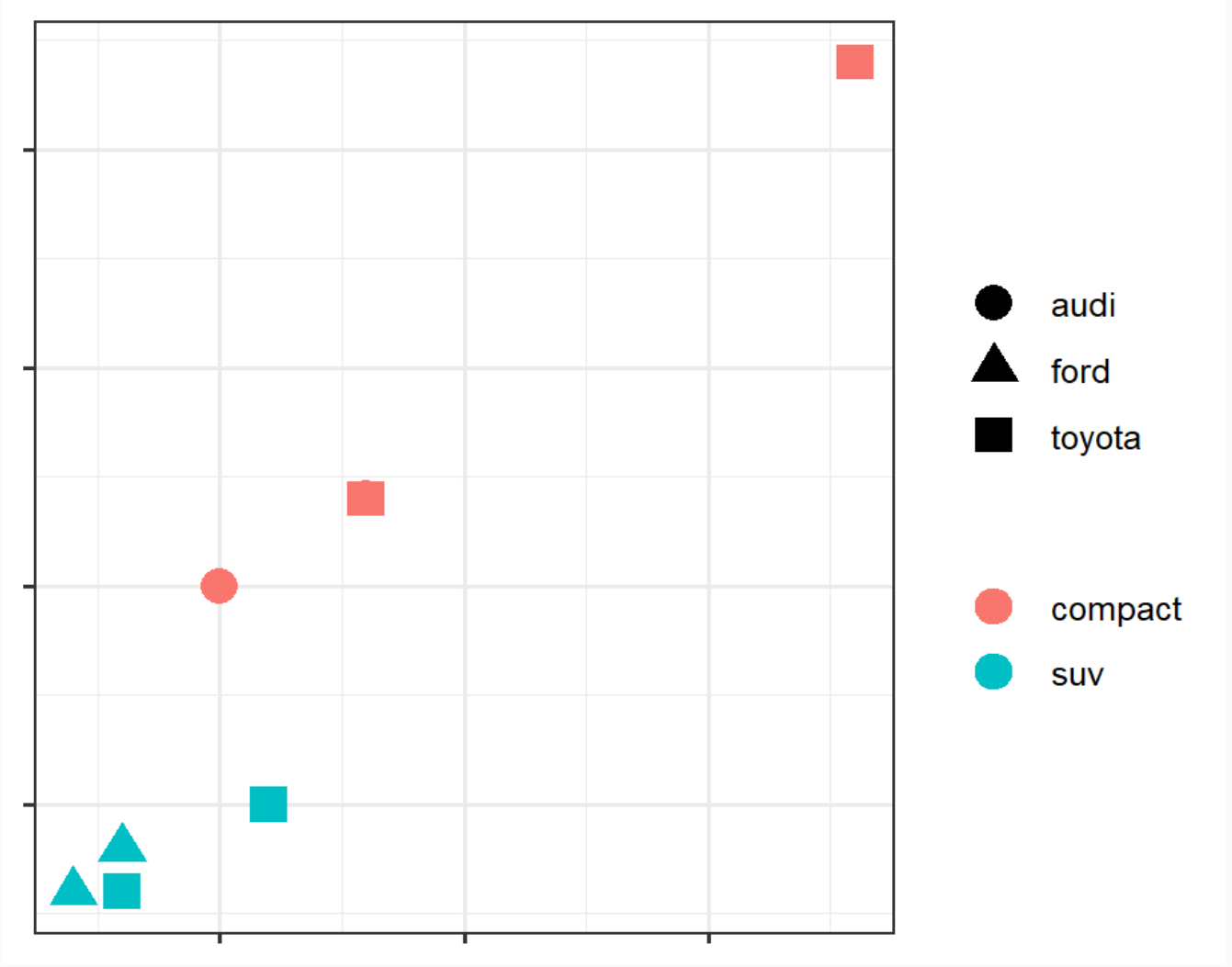


Guess the data behind this plot?

MPG Ratings of Cars

- Manufacturer
- **Car Type (Class)**
- City MPG
- Highway MPG

What variable is represented on the x axis?

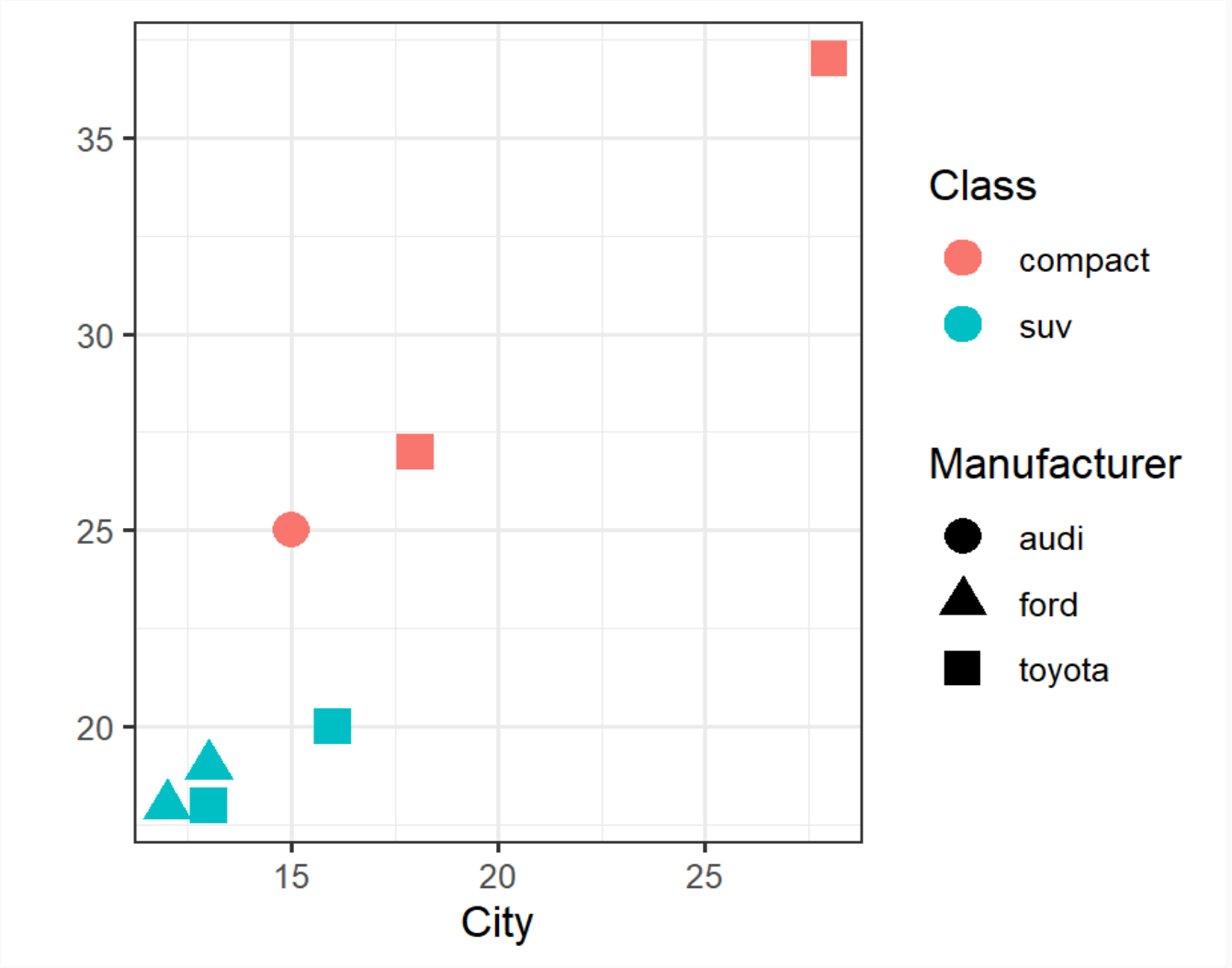


Guess the data behind this plot?

MPG Ratings of Cars

- Manufacturer
- Car Type (Class)
- **City MPG**
- Highway MPG

What variable is on the Y axis?

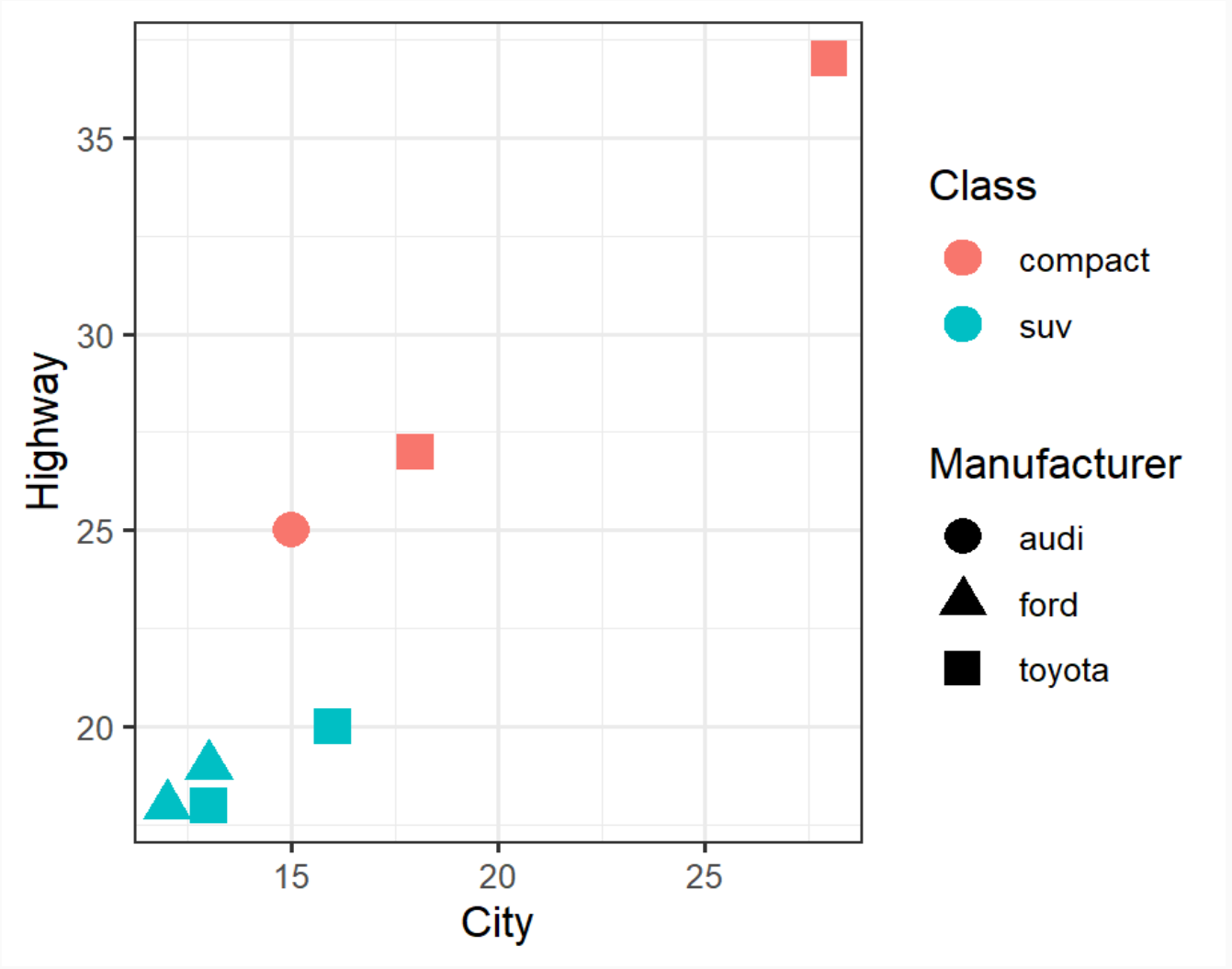


Guess the data behind this plot?

MPG Ratings of Cars

- Manufacturer
- Car Type (Class)
- City MPG
- **Highway MPG**

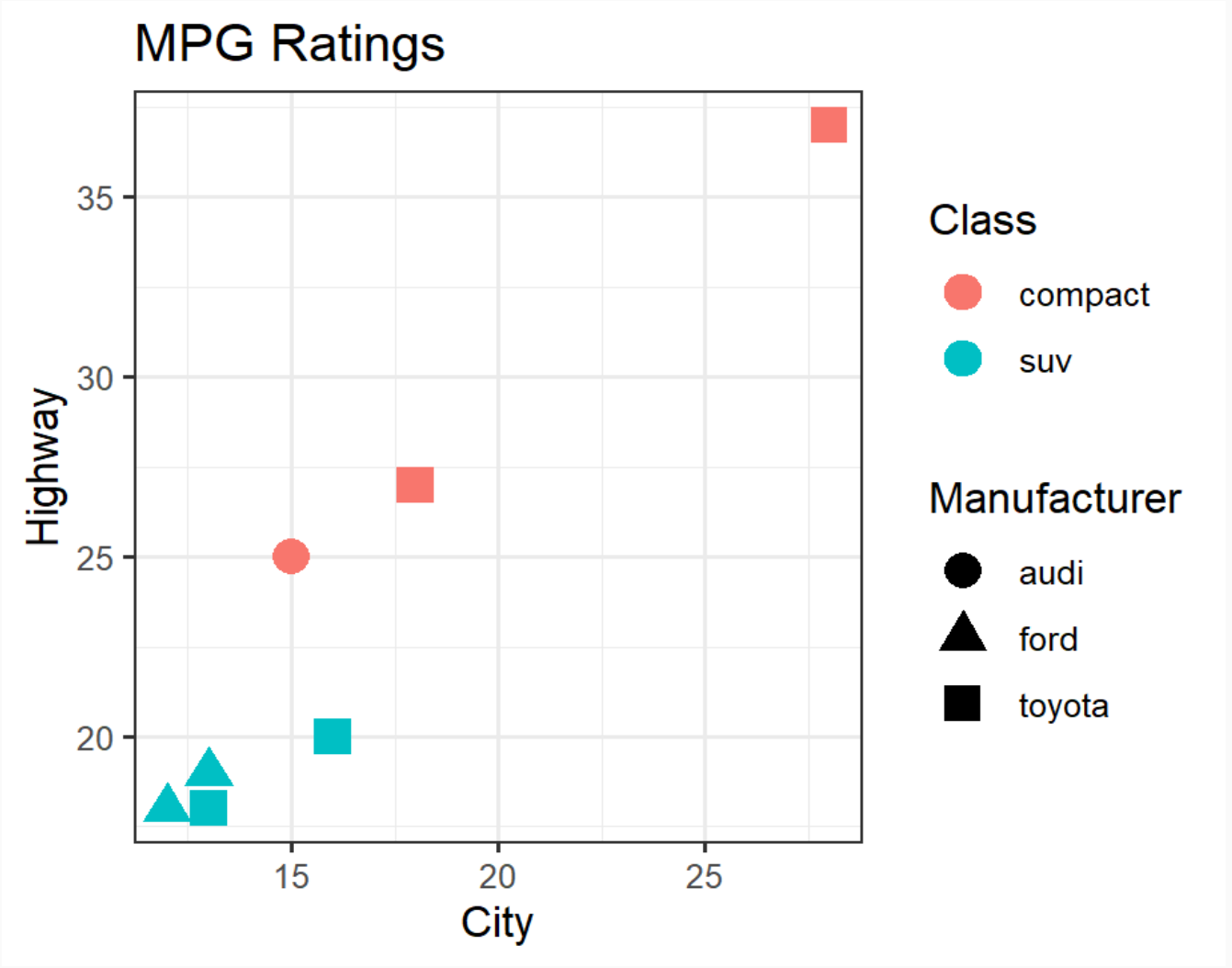
What is the title of the plot?



Guess the data behind this plot?

MPG Ratings of Cars

- Manufacturer
- Car Type (Class)
- City MPG
- Highway MPG



How do we express visuals in words?

- **Data** to be visualized
- **Aesthetic mappings** from data to visual component
- **Geometric objects** that appear on the plot
- **Statistics** transform data on the way to visualization
- **Coordinates** organize location of geometric objects
- **Scales** define the range of values for aesthetics
- **Facets** group into subplots
- **Themes** the visual elements of the plot not linked to the data

gg is for Grammar of Graphics

Data

```
ggplot(data)
```

Tidy Data

1. Each **variable** forms a column
2. Each **observation** forms a row
3. Each **value** is a cell

Start by asking

1. What information do I want to use in my visualization?
2. Is that data contained in one column/row for a given data point?

gg is for Grammar of Graphics

Data

```
ggplot(data)
```

country	1997	2002	2007
Chile	14.599929	15.497046	16.284741
Rwanda	7.212583	7.852401	8.860588
Syria	15.081016	17.155814	19.314747

```
tidy_pop ← gather(messy_pop, 'year', 'pop', -country)
```

country	year	pop
Chile	1997	14.600
Rwanda	1997	7.213
Syria	1997	15.081
Chile	2002	15.497

gg is for Grammar of Graphics

Data

Aesthetics

```
+ aes()
```

Map data to visual elements or parameters

- year
- pop
- country

country	year	pop
Chile	1997	14.600
Rwanda	1997	7.213
Syria	1997	15.081
Chile	2002	15.497

gg is for Grammar of Graphics

Data

Aesthetics

```
+ aes()
```

Map data to visual elements or parameters

- year → **x**
- pop → **y**
- country → *shape*, *color*, etc.

country	year	pop
Chile	1997	14.600
Rwanda	1997	7.213
Syria	1997	15.081
Chile	2002	15.497

gg is for Grammar of Graphics

Data

Aesthetics

```
+ aes()
```

Map data to visual elements or parameters

```
aes(
  x = year,
  y = pop,
  colour = country
)
```

gg is for Grammar of Graphics

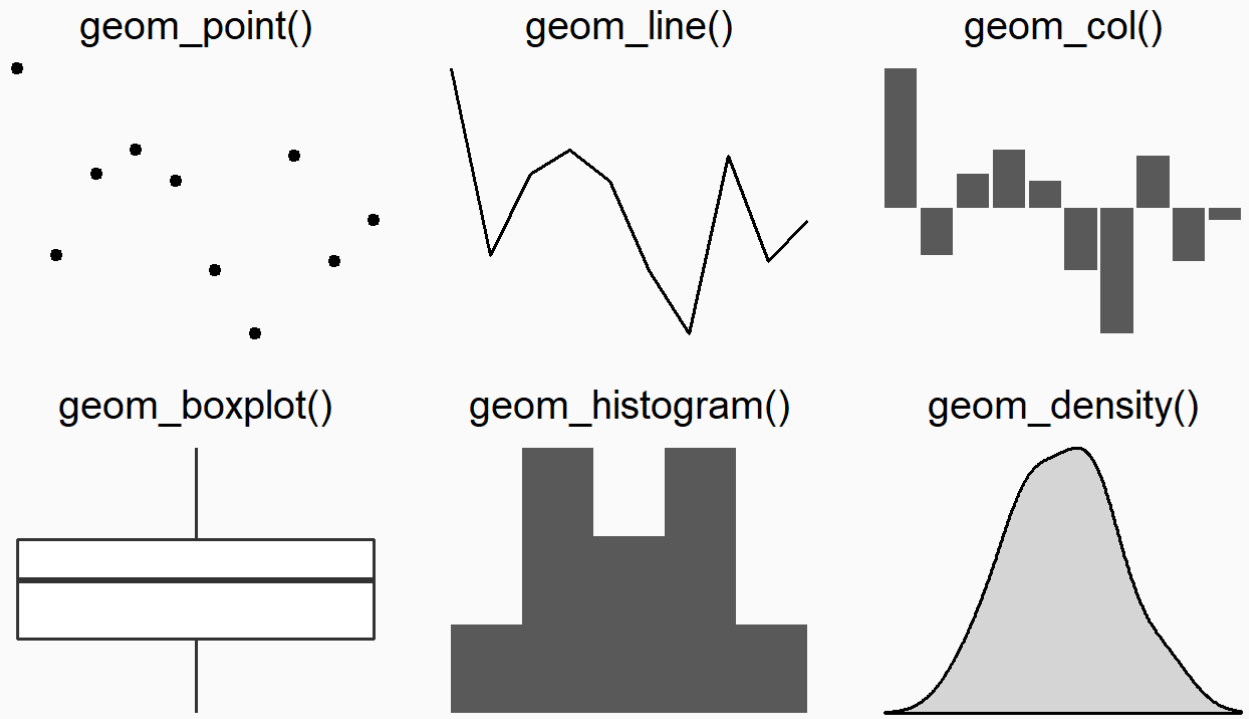
Data

Aesthetics

Geoms

```
+ geom_*()
```

Geometric objects displayed on the plot



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

```
+ geom_*()
```

Here are the some of the most widely used geoms

Type	Function
Point	`geom_point()`
Line	`geom_line()`
Bar	`geom_bar()`, `geom_col()`
Histogram	`geom_histogram()`
Regression	`geom_smooth()`
Boxplot	`geom_boxplot()`
Text	`geom_text()`
Vert./Horiz. Line	`geom_{vh}line()`
Count	`geom_count()`
Density	`geom_density()`

https://eric.netlify.com/2017/08/10/most-popular-ggplot2-geoms/

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

```
+ geom_*()
```

See http://ggplot2.tidyverse.org/reference/ for many more options

```
##  [1] "geom_abline"     "geom_area"       "geom_bar"        "geom_bin2d"
##  [5] "geom_blank"      "geom_boxplot"    "geom_col"        "geom_contour"
##  [9] "geom_count"      "geom_crossbar"   "geom_curve"      "geom_density"
## [13] "geom_density_2d" "geom_density2d"  "geom_dotplot"    "geom_errorbar"
## [17] "geom_errorbarh"  "geom_freqpoly"   "geom_hex"        "geom_histogram"
## [21] "geom_hline"      "geom_jitter"     "geom_label"      "geom_line"
## [25] "geom_linerange"  "geom_map"        "geom_path"       "geom_point"
## [29] "geom_pointrange" "geom_polygon"    "geom_qq"         "geom_qq_line"
## [33] "geom_quantile"   "geom_raster"     "geom_rect"       "geom_ribbon"
## [37] "geom_rug"        "geom_segment"    "geom_sf"         "geom_sf_label"
## [41] "geom_sf_text"    "geom_smooth"     "geom_spoke"      "geom_step"
## [45] "geom_text"       "geom_tile"       "geom_violin"     "geom_vline"
```

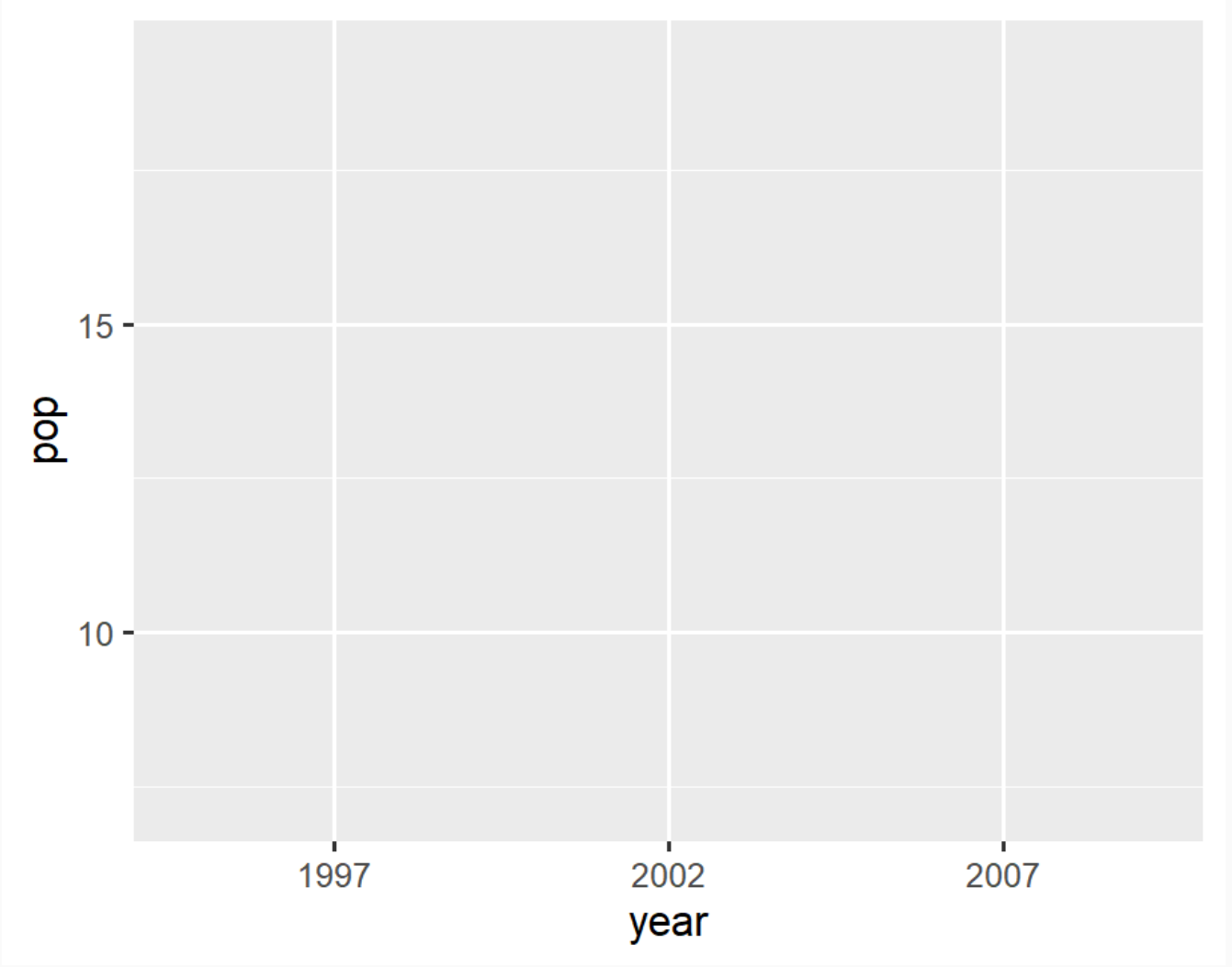
Or just start typing `geom_` in RStudio

Our first plot!

```
ggplot(tidy_pop)
```

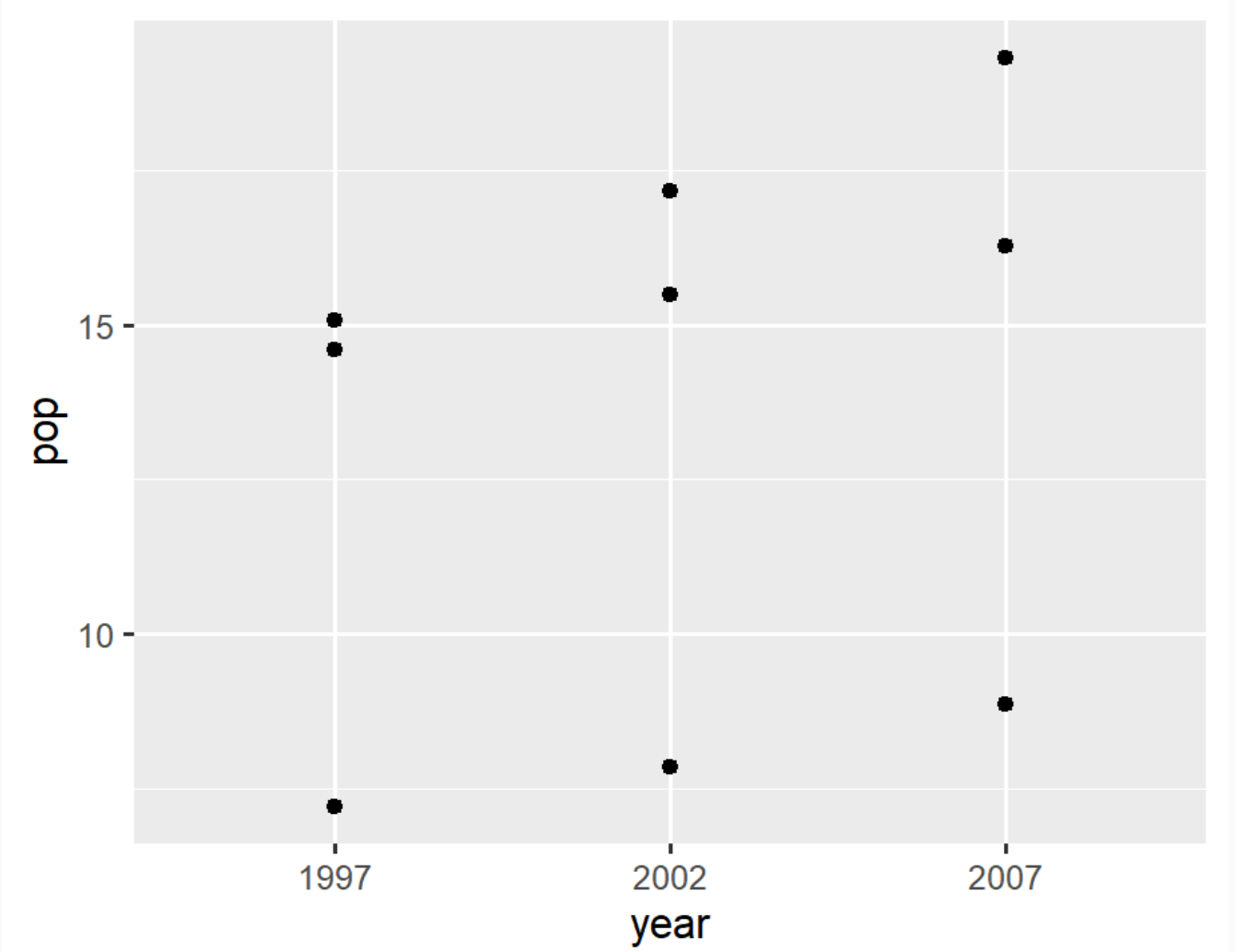
Our first plot!

```
ggplot(tidy_pop,
  aes(x = year,
      y = pop)
  )
```



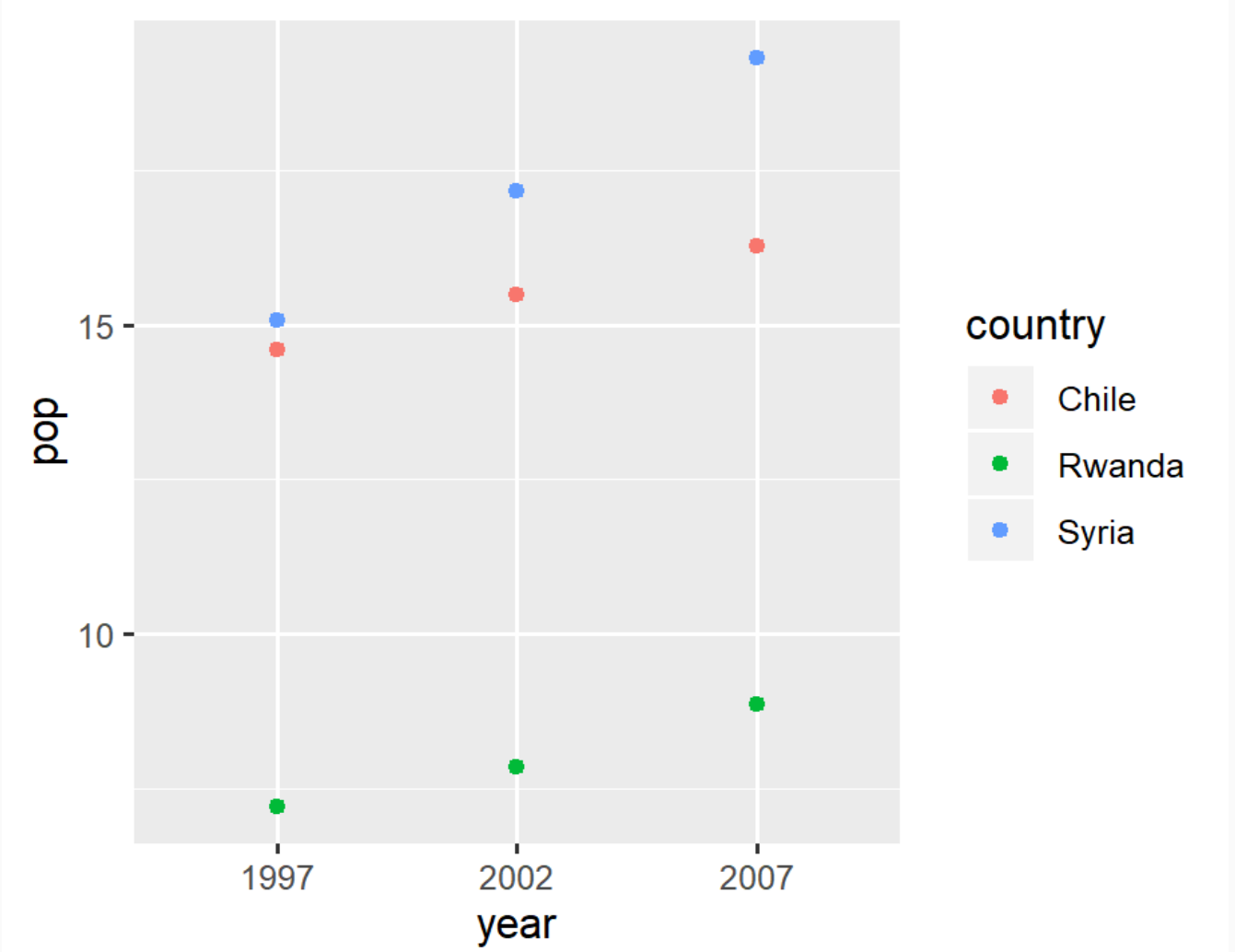
Our first plot!

```
ggplot(tidy_pop,
  aes(x = year,
      y = pop)
  ) +
  geom_point()
```



Our first plot!

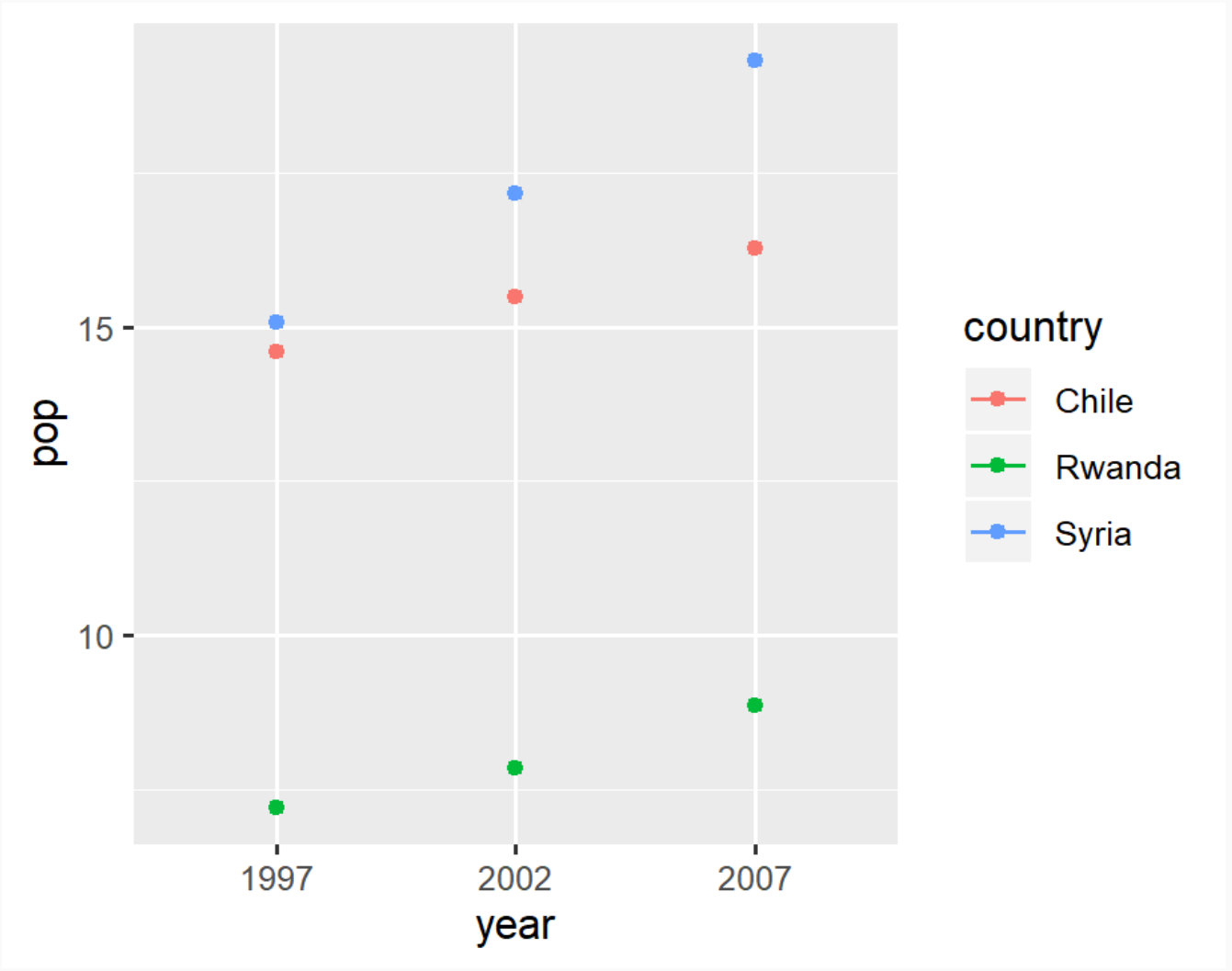
```
ggplot(tidy_pop,
  aes(x = year,
      y = pop,
      color = country)
  ) +
  geom_point()
```



Our first plot!

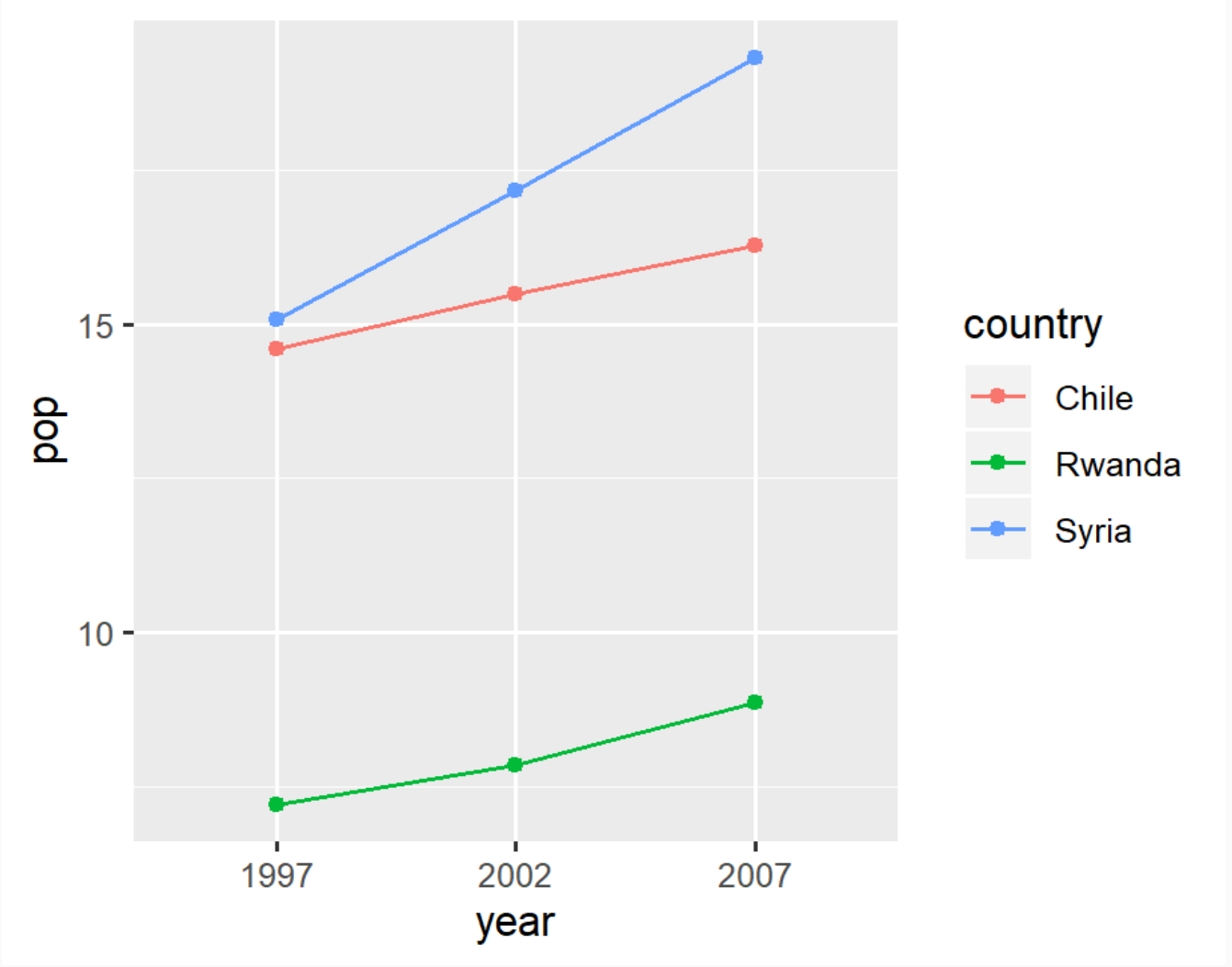
```
ggplot(tidy_pop,
  aes(x = year,
      y = pop,
      color = country)
  ) +
  geom_point() +
  geom_line()
```

geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?



Our first plot!

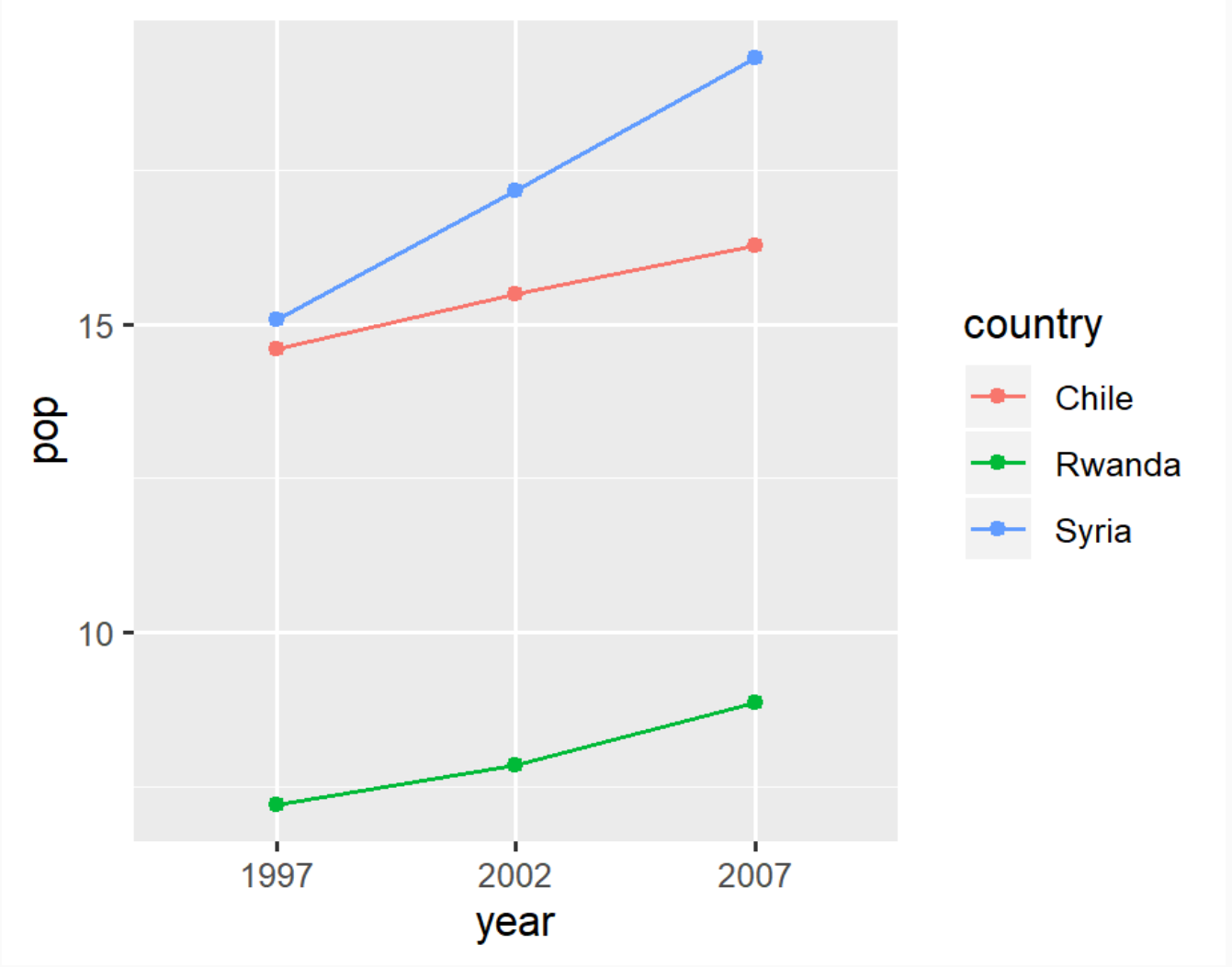
```
ggplot(tidy_pop,
  aes(x = year,
      y = pop,
      color = country)
  ) +
  geom_point() +
  geom_line(
    aes(group = country))
```



Our first plot!

```
g ← ggplot(tidy_pop,
  aes(x = year,
      y = pop,
      color = country)
  ) +
  geom_point() +
  geom_line(
    aes(group = country))

g
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

```
+ geom_*()
```

```
geom_*(mapping = aes(), data, stat, position)
```

- `data` Geoms can have their own data
  - Has to map onto global coordinates
- `aes` Geoms can have their own aesthetics
  - Inherits global aesthetics
  - Have geom-specific aesthetics
    - `geom_point` needs `x` and `y`, optional `shape`, `color`, `size`, etc.
    - `geom_ribbon` requires `x`, `ymin` and `ymax`, optional `fill`
  - Use `?` to find out the aesthetics required and the ones you can change: `?geom_ribbon`

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

```
+ geom_*()
```

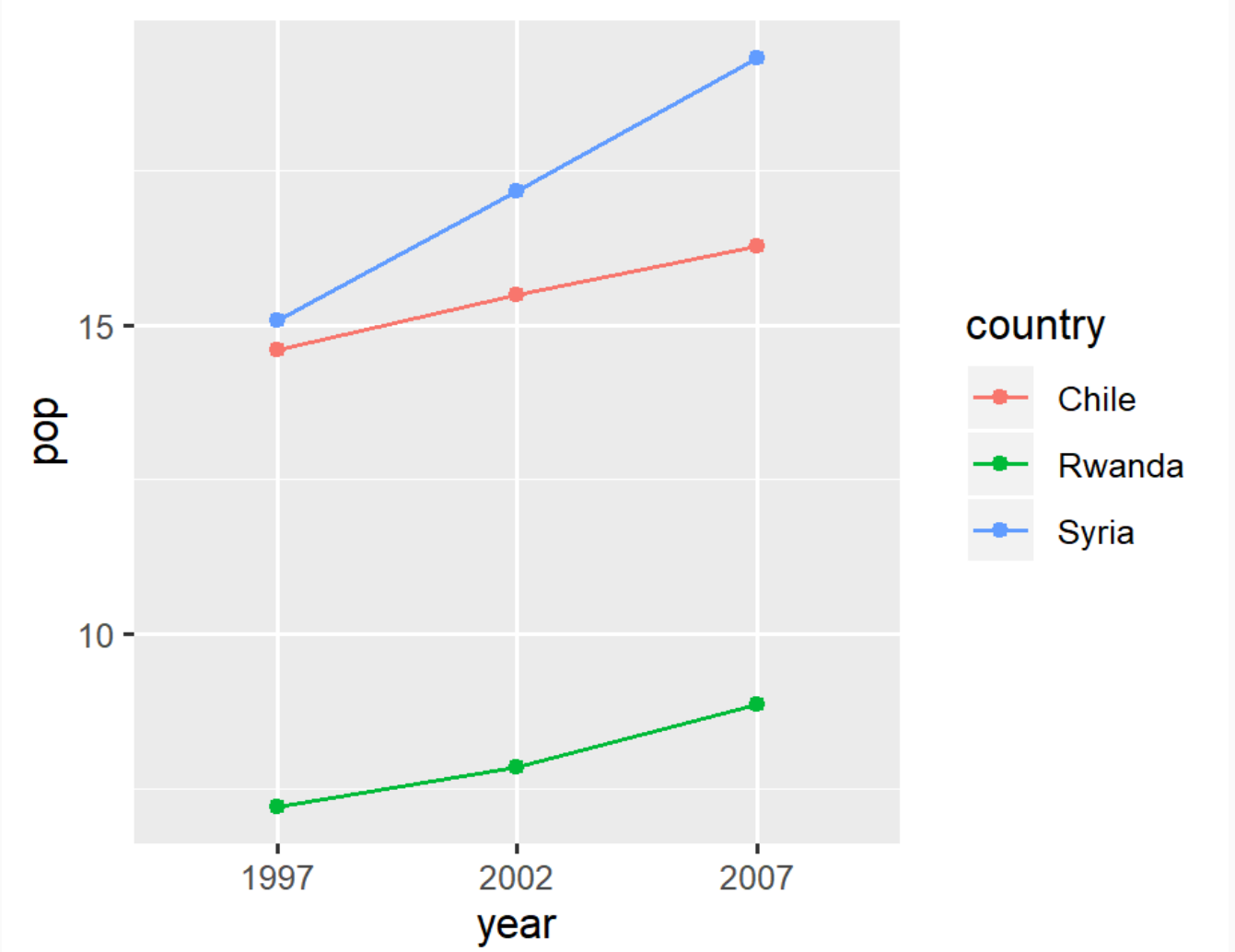
```
geom_*(mapping, data, stat, position)
```

- `stat` Some geoms apply further transformations to the data
  - All respect `stat = 'identity'`
  - Ex: `geom_histogram` uses `stat_bin()` to group observations
- `position` Some adjust location of objects
  - `'dodge'`, `'stack'`, `'jitter'`

Our first plot!

```
g ← ggplot() +
  geom_point(
    data = tidy_pop,
    aes(x = year,
      y = pop,
      color = country)
  ) +
  geom_line(
    data = tidy_pop,
    aes(x = year,
      y = pop,
      color = country,
      group = country)
    )

g
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

```
+ geom_*()
```

```
geom_*(mapping = aes(), data, stat, position)
```

- `data` Geoms can have their own data
  - Has to map onto global coordinates

What would the advantage be for a geom to have their own data?

gg is for Grammar of Graphics

Data

Aesthetics

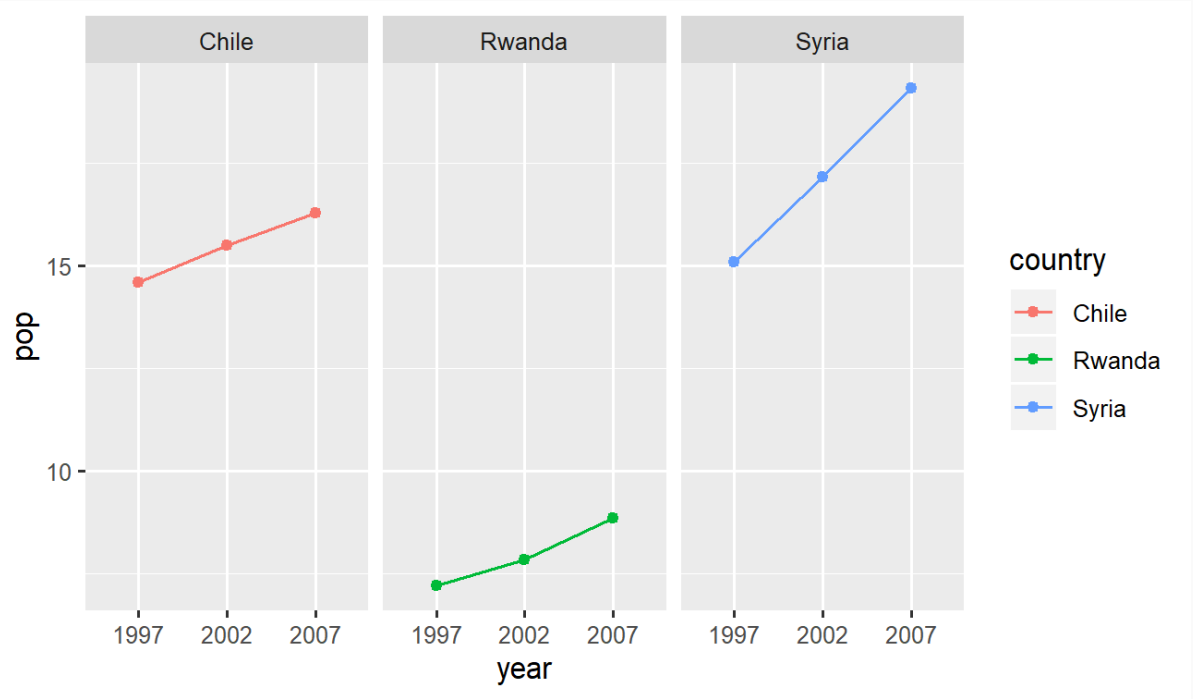
Geoms

Facet

```
+facet_wrap()

+facet_grid()
```

```
g + facet_wrap(~ country)
```



gg is for Grammar of Graphics

Data

Aesthetics

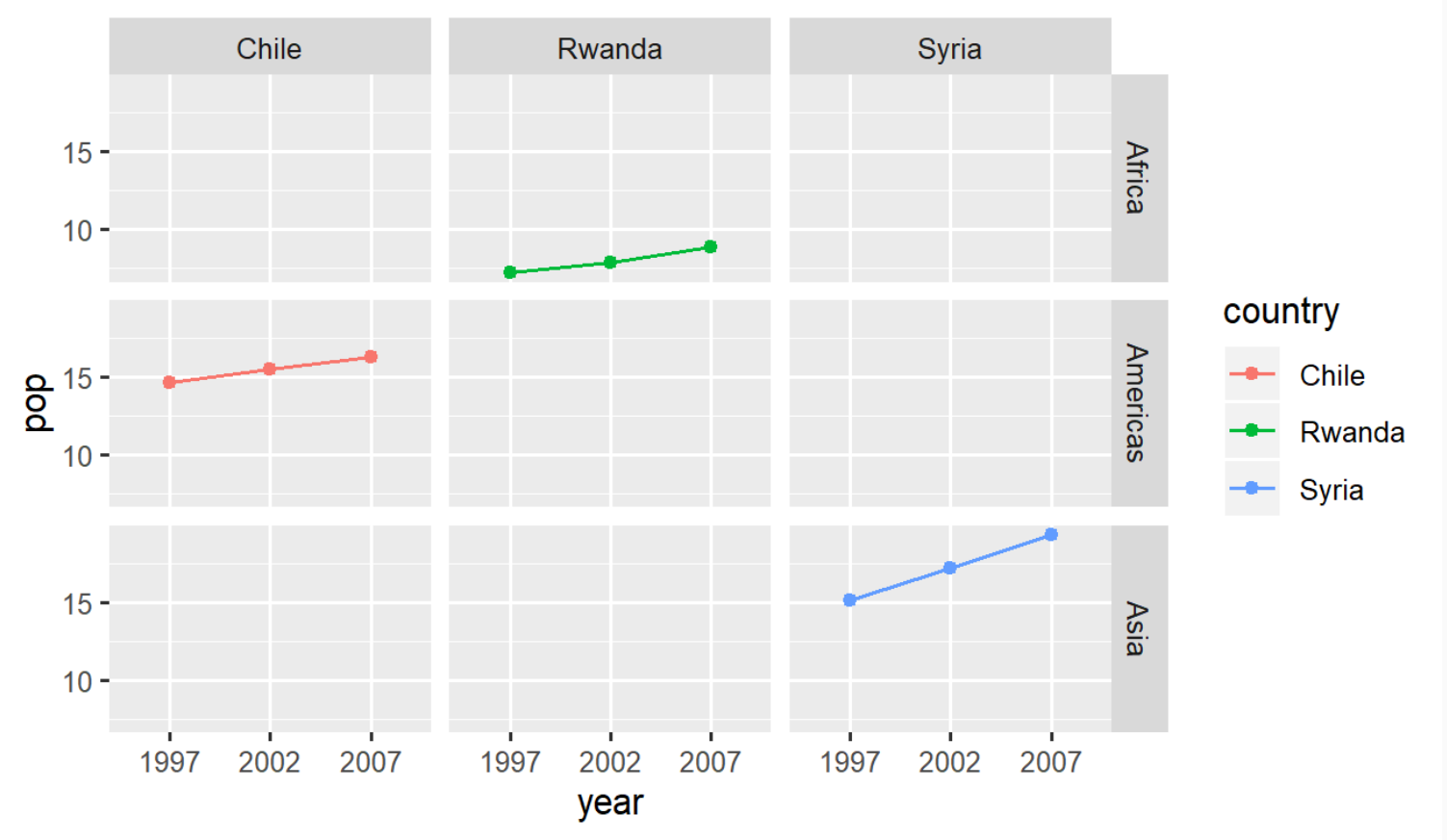
Geoms

Facet

```
+facet_wrap()

+facet_grid()
```

```
g + facet_grid(continent ~ country)
```



gg is for Grammar of Graphics

Data

Aesthetics

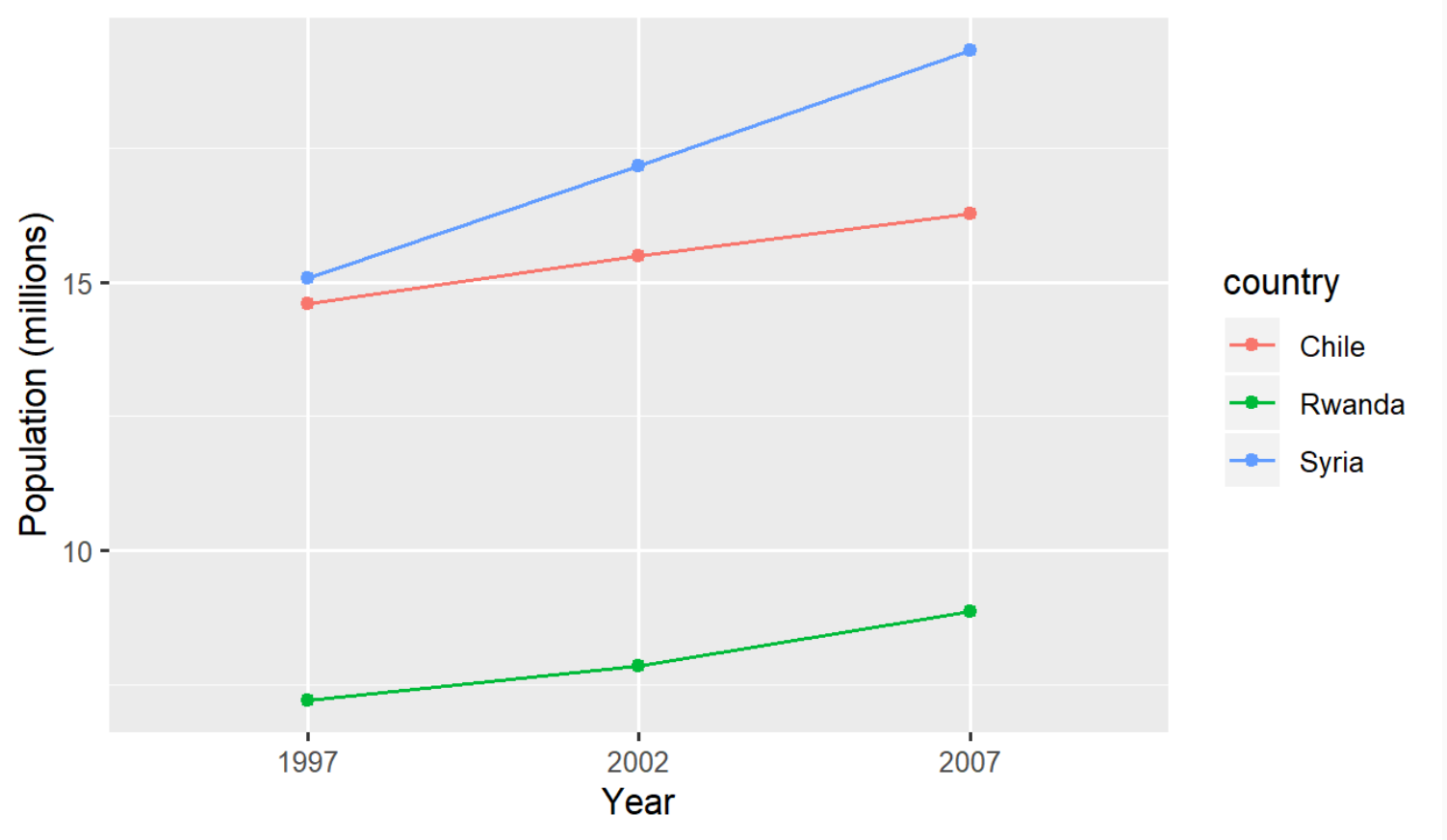
Geoms

Facet

Labels

```
+ labs()
```

```
(g ← g + labs(x = "Year", y = "Population (millions)"))
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

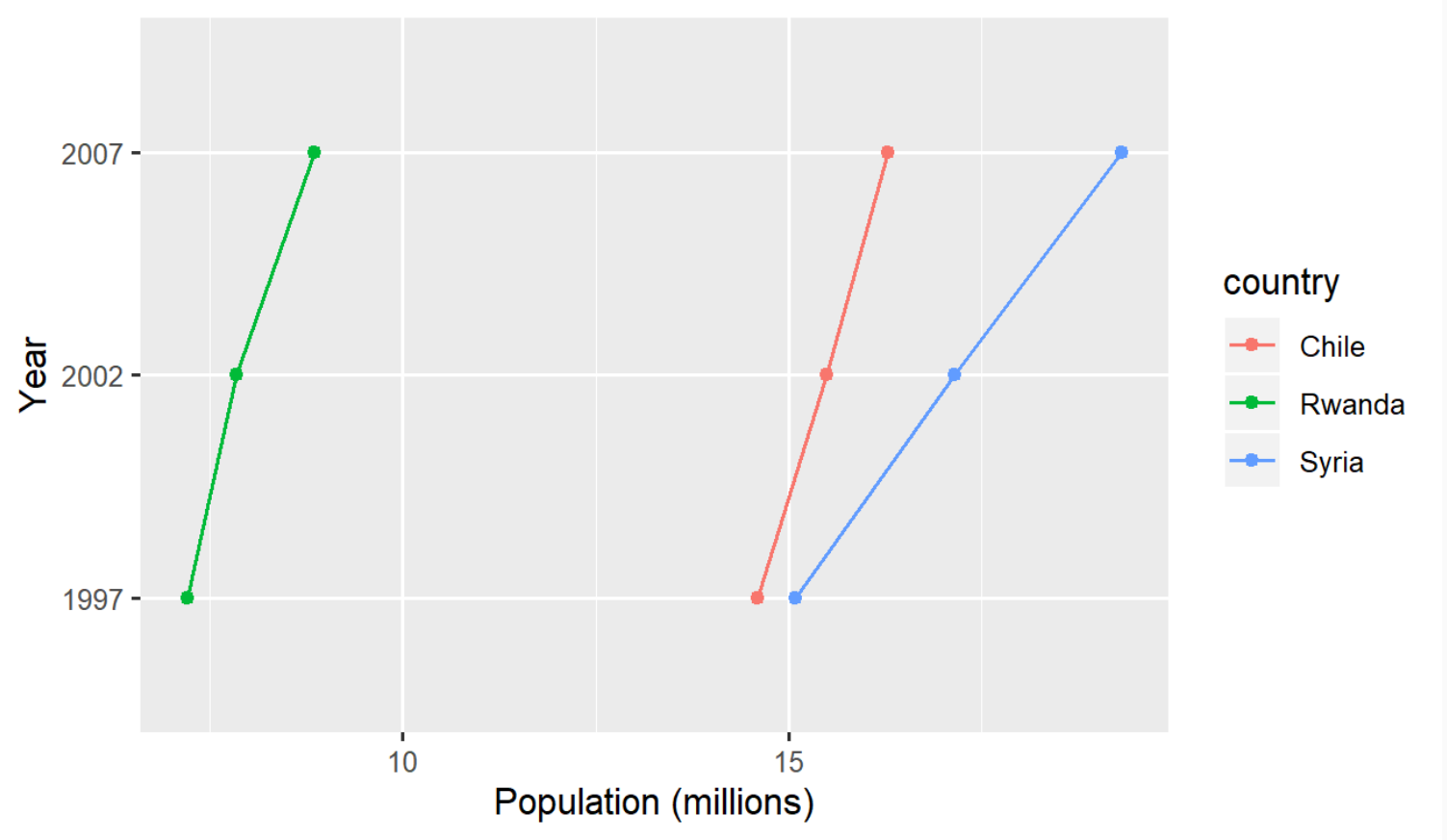
Facet

Labels

Coords

```
+ coord_*()
```

```
g + coord_flip()
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Coords

Scales

```
+ scale_*_*()
```

`scale` + `_` + `<aes>` + `_` + `<type>` + `()`

`<aes>` = parameter to adjust; `<type>` = Parameter Type

- I want to use a different color palette
  `scale_fill_discrete()`
  `scale_color_continuous()`
- I want to rescale y-axis as log
  `scale_y_log10()`
- I want to change my discrete x-axis
  `scale_x_discrete()`

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

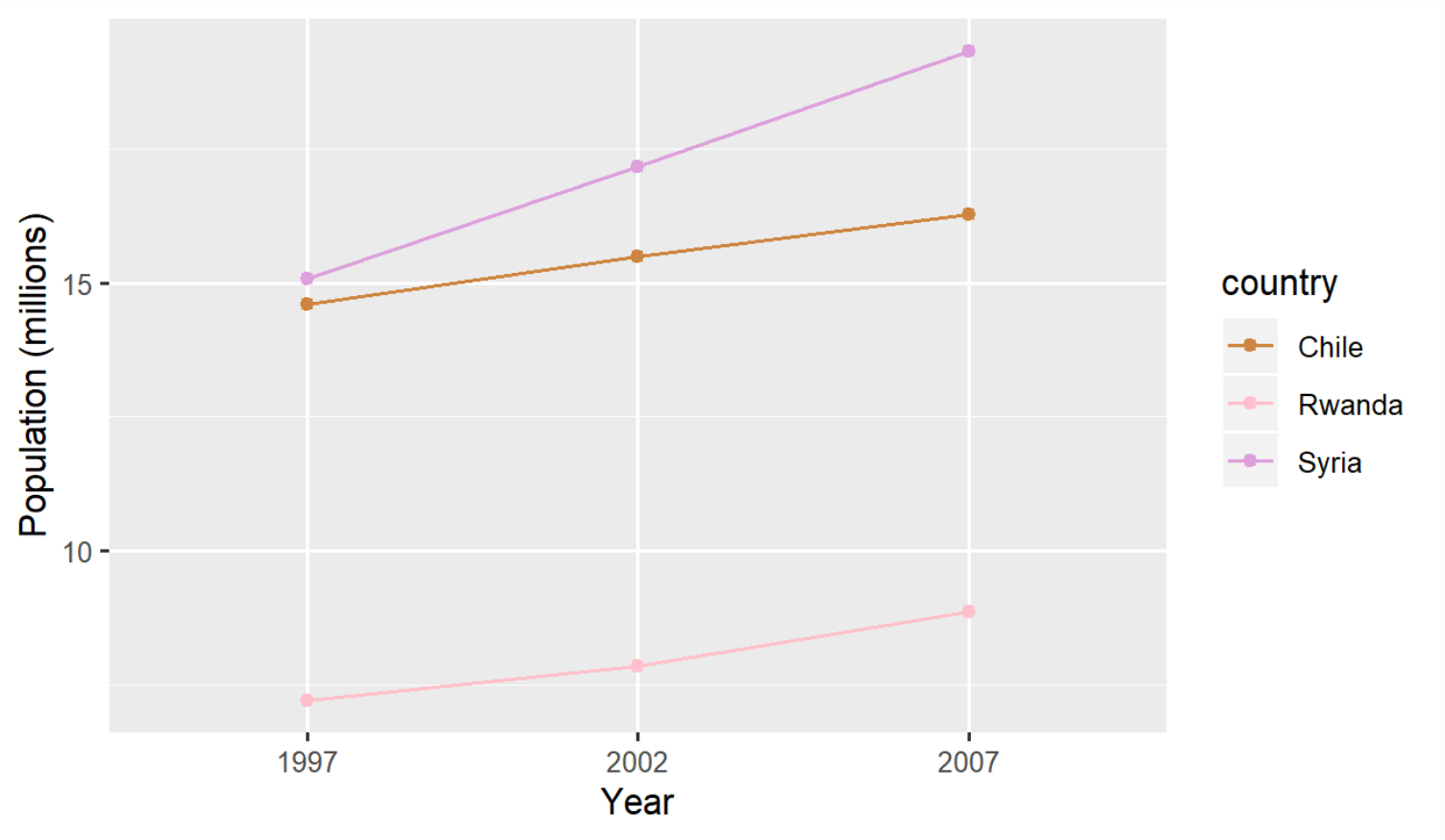
Labels

Coords

Scales

```
+ scale_*_*()
```

```
g + scale_color_manual(values = c("peru", "pink", "plum"))
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

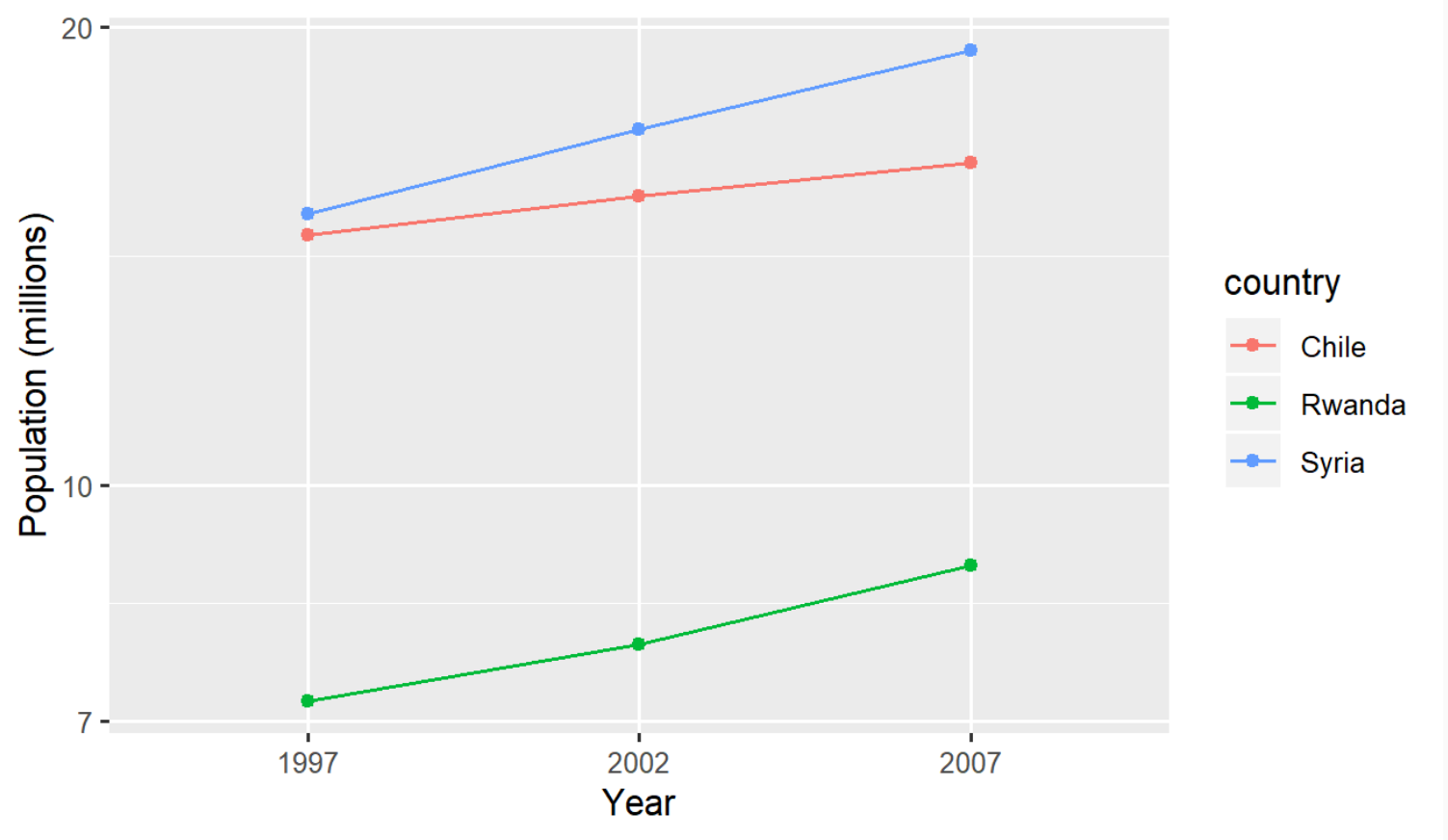
Labels

Coords

Scales

```
+ scale_*_*()
```

```
g + scale_y_log10()
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Coords

Scales

```
+ scale_*_*()
```

```
g + scale_x_discrete(labels = c("MCMXCVII", "MMII", "MMVII"))
```

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Coords

Scales

Theme

```
+ theme()
```

Change the appearance of plot decorations
i.e. things that aren't mapped to data

A few "starter" themes ship with the package

- `g + theme_bw()`
- `g + theme_dark()`
- `g + theme_gray()`
- `g + theme_light()`
- `g + theme_minimal()`

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

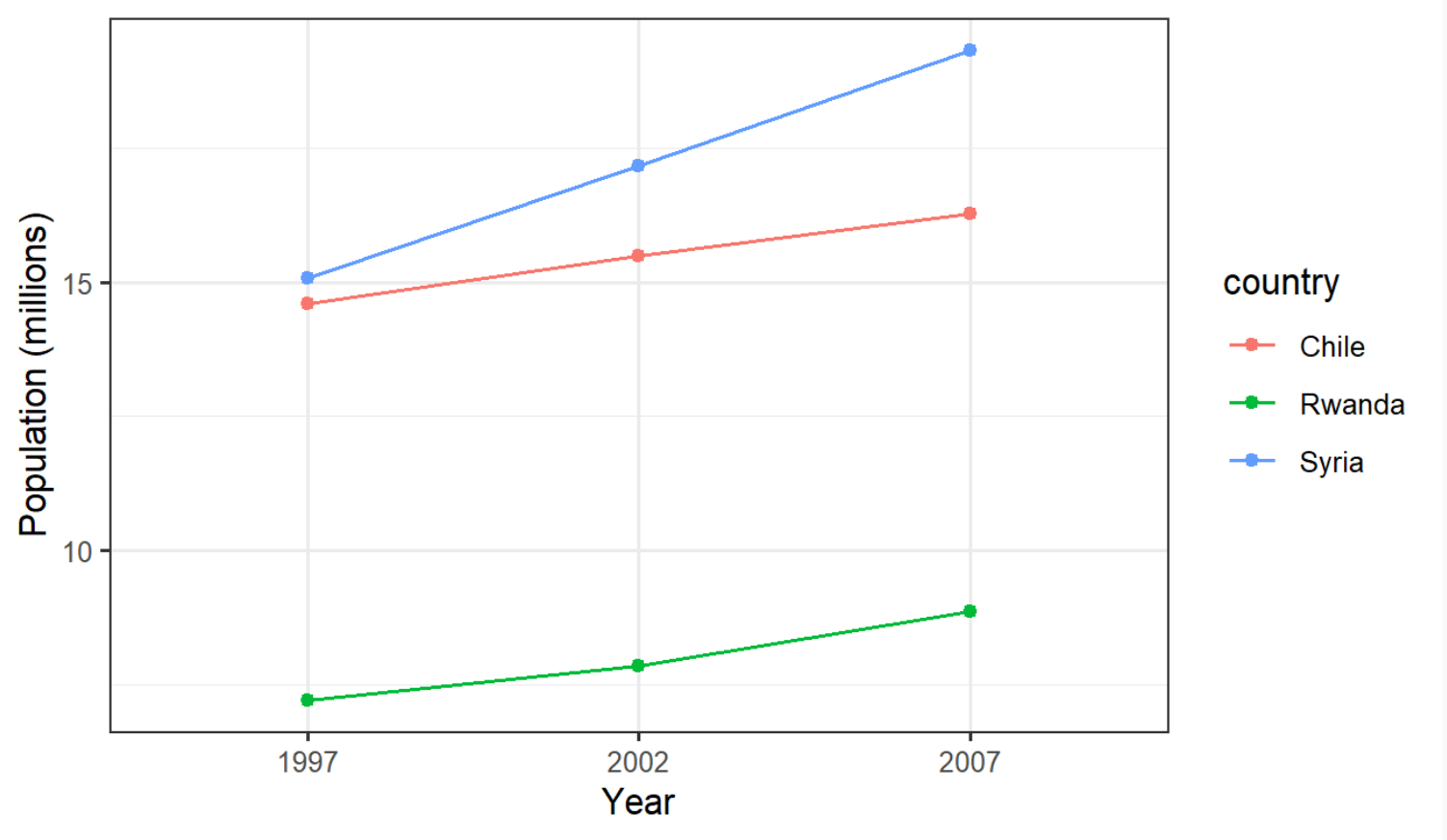
Coords

Scales

Theme

```
+ theme()
```

```
g + theme_bw()
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Coords

Scales

Theme

```
+ theme()
```

Huge number of parameters, grouped by plot area:

- Global options: `line`, `rect`, `text`, `title`
- `axis`: x-, y- or other axis title, ticks, lines
- `legend`: Plot legends
- `panel`: Actual plot area
- `plot`: Whole image
- `strip`: Facet labels

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Coords

Scales

Theme

```
+ theme()
```

Theme options are supported by helper functions:

- `element_blank()` removes the element
- `element_line()`
- `element_rect()`
- `element_text()`

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

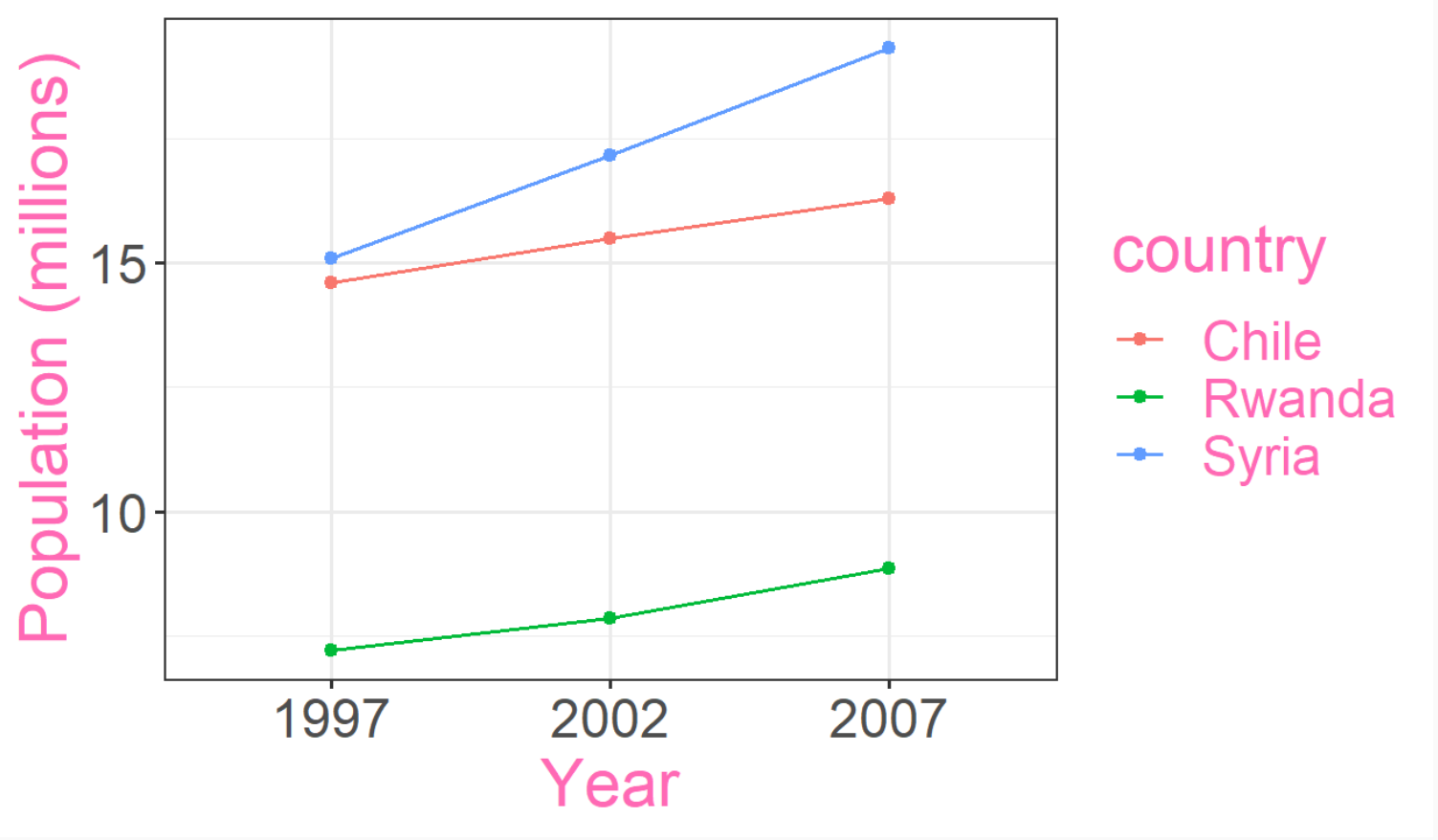
Coords

Scales

Theme

```
+ theme()
```

```
g + theme_bw() + theme(text = element_text(colour = "hotpink", size = 20))
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Coords

Scales

Theme

```
+ theme()
```

You can also set the theme globally with `theme_set()`

```
my_theme ← theme_bw() +
  theme(
    text = element_text(family = "Palatino", size = 12),
    panel.border = element_rect(colour = 'grey80'),
    panel.grid.minor = element_blank()
  )

theme_set(my_theme)
```

All plots will now use this theme!

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

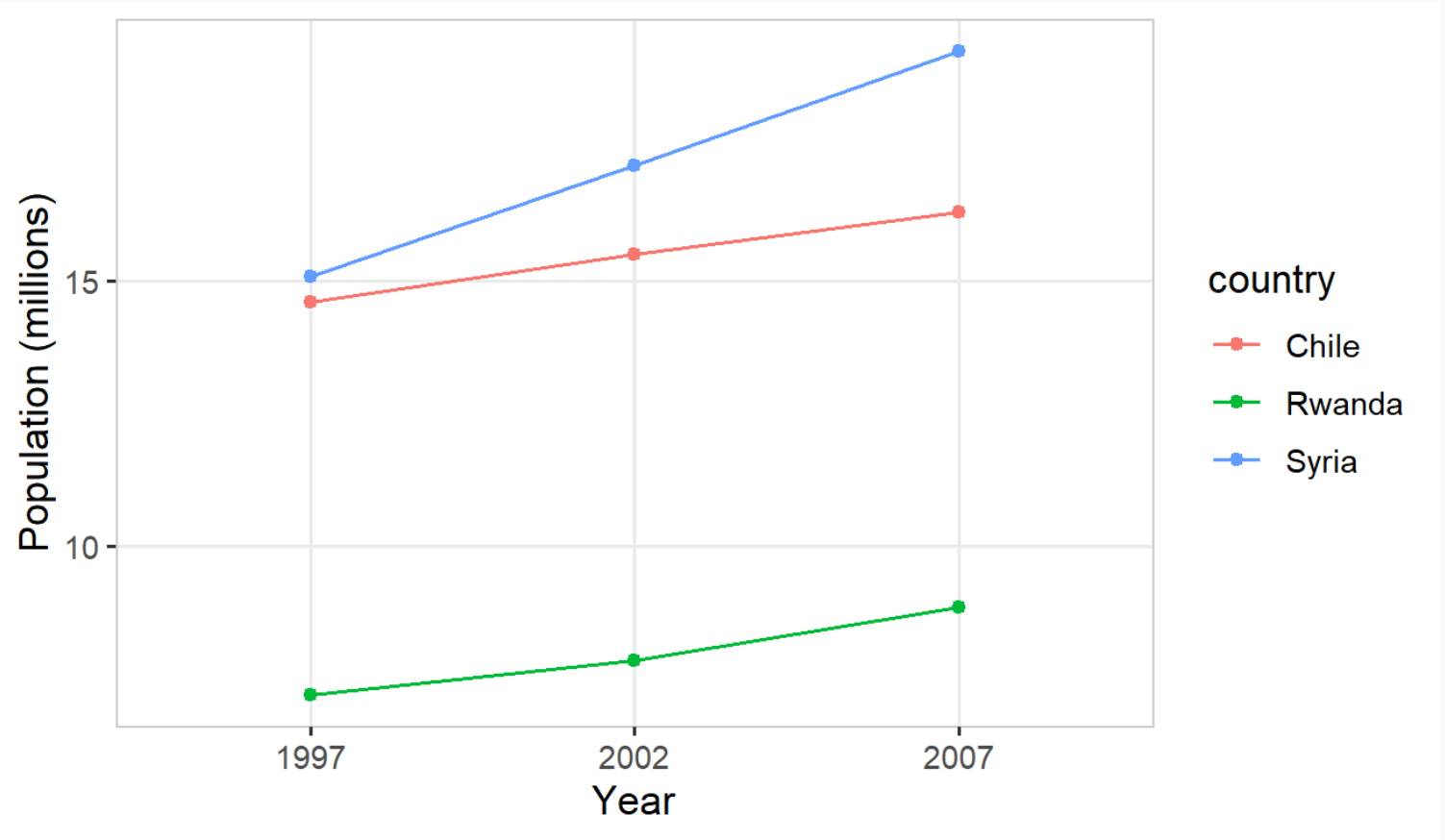
Coords

Scales

Theme

```
+ theme()
```

```
gg
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

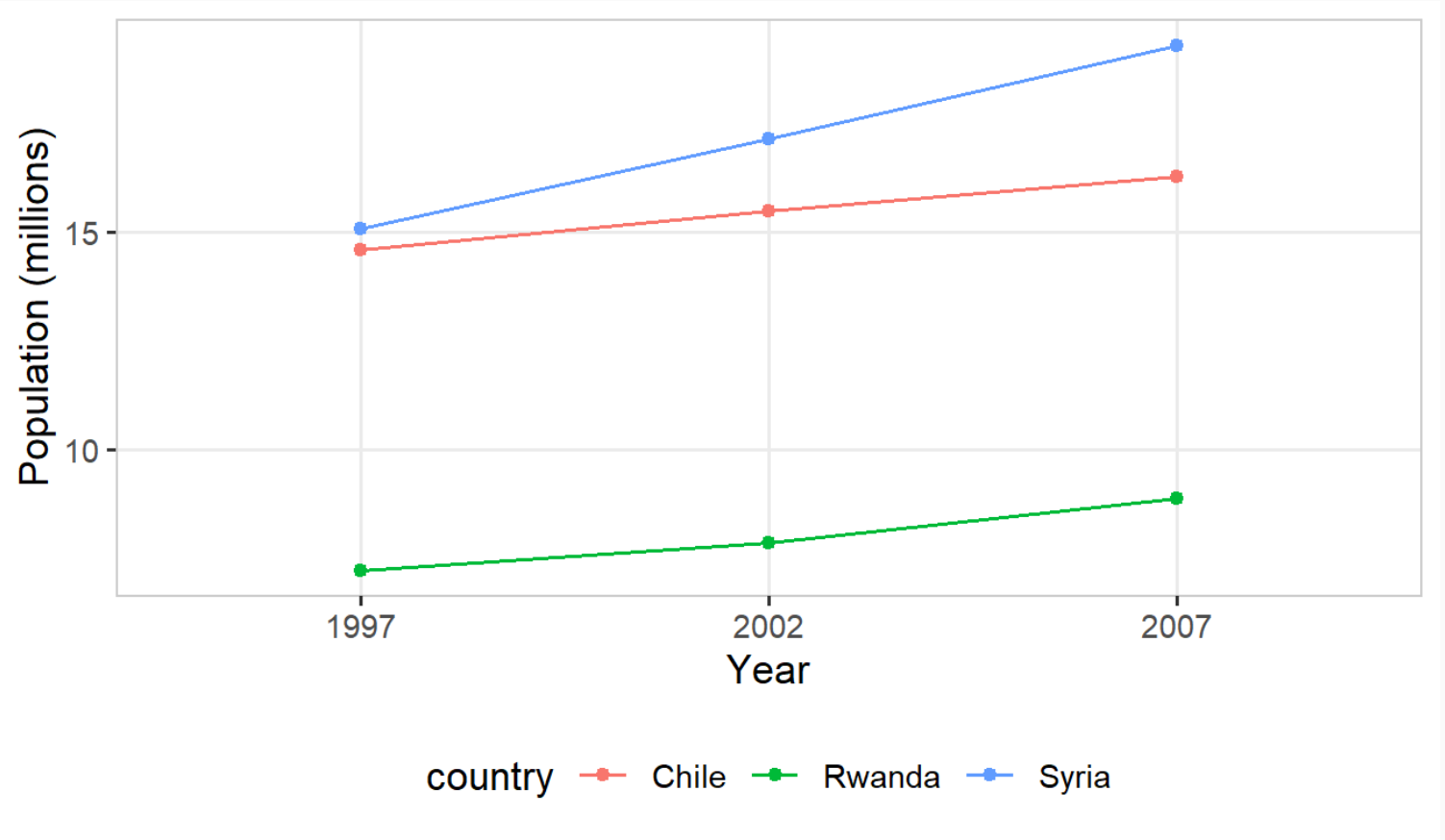
Coords

Scales

Theme

```
+ theme()
```

```
g + theme(legend.position = 'bottom')
```



Save Your Work

To save your plot, use **ggsave**.

```
ggsave(
  filename = "my_plot.png",
  plot = my_plot,
  width = 10,
  height = 8,
  dpi = 100,
  device = "png"
)
```

You have the power!

"Live" Coding

```
library(gapminder)
```

head(gapminder)

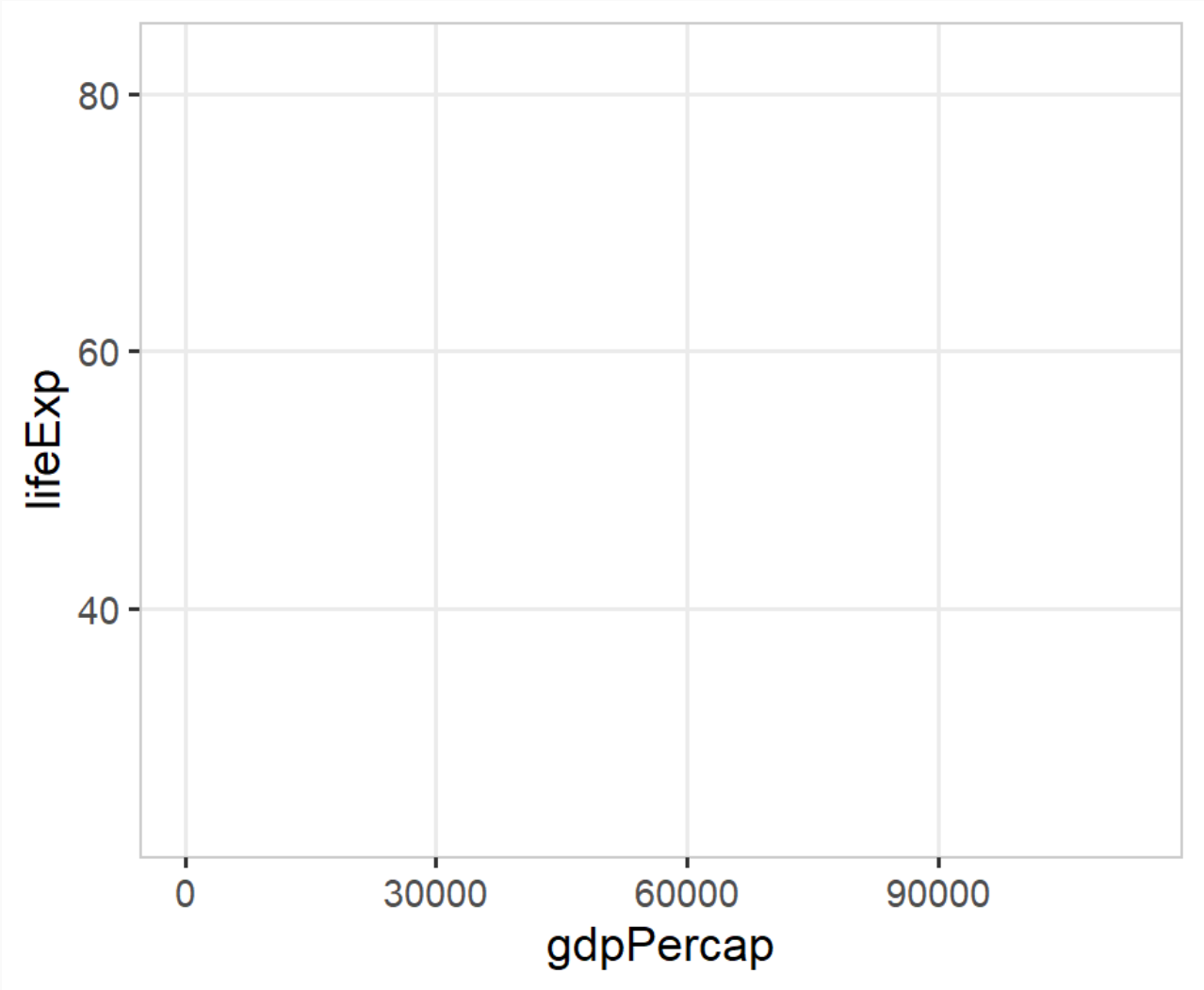
country	continent	year	lifeExp	pop	gdpPercap
Afghanistan	Asia	1952	28.801	8425333	779.4453
Afghanistan	Asia	1957	30.332	9240934	820.8530
Afghanistan	Asia	1962	31.997	10267083	853.1007
Afghanistan	Asia	1967	34.020	11537966	836.1971
Afghanistan	Asia	1972	36.088	13079460	739.9811
Afghanistan	Asia	1977	38.438	14880372	786.1134

glimpse(gapminder)

```
Observations: 1,704
Variables: 6
$ country   <fct> Afghanistan, Afghanistan, Afghanistan, Afghanistan, Afghanistan, Af…
$ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, A…
$ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, 2002, 2…
$ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.822, 41.…
$ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 12881816, …
$ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, 978.011…
```

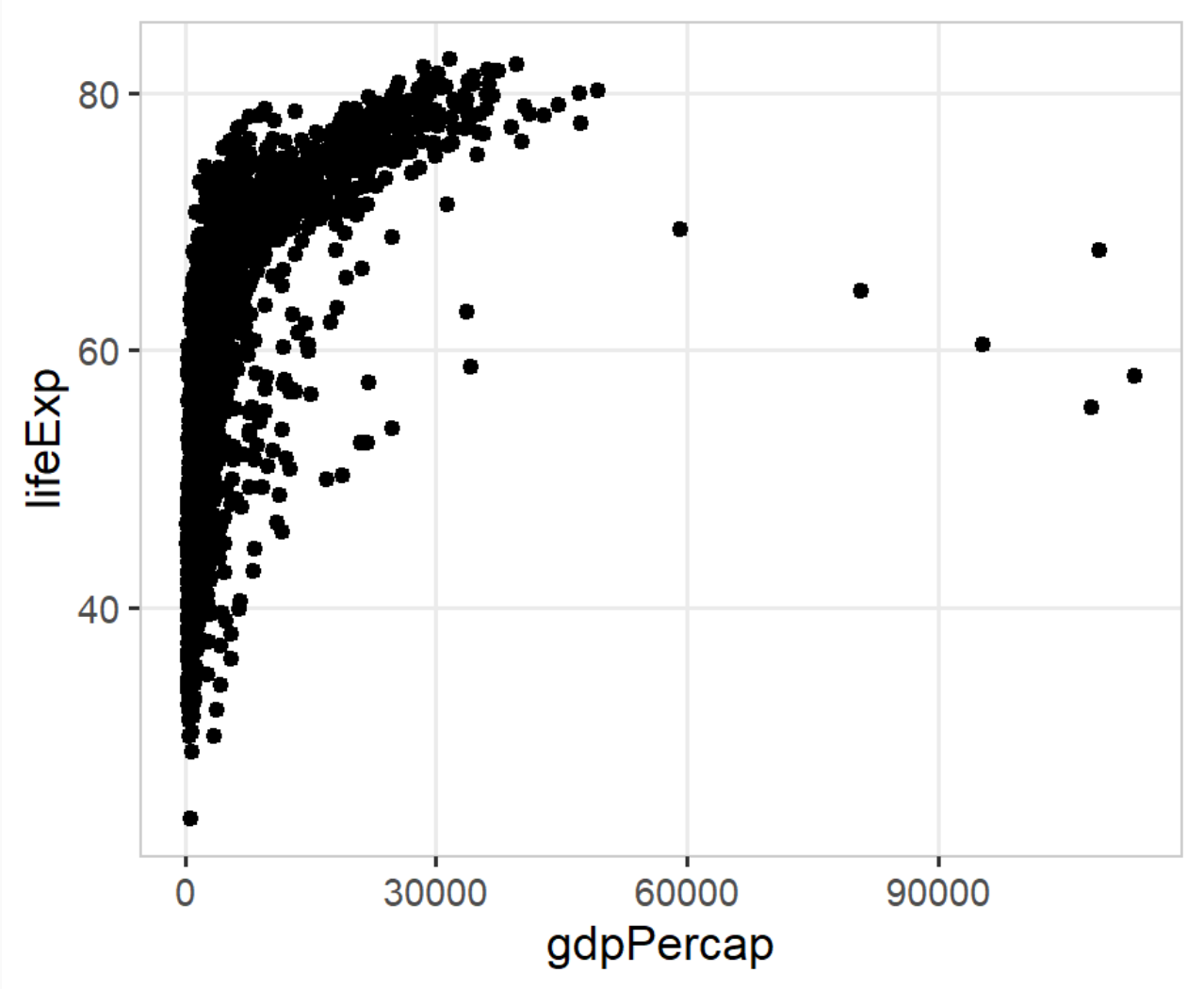
Let's start with `lifeExp` vs `gdpPercap`

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



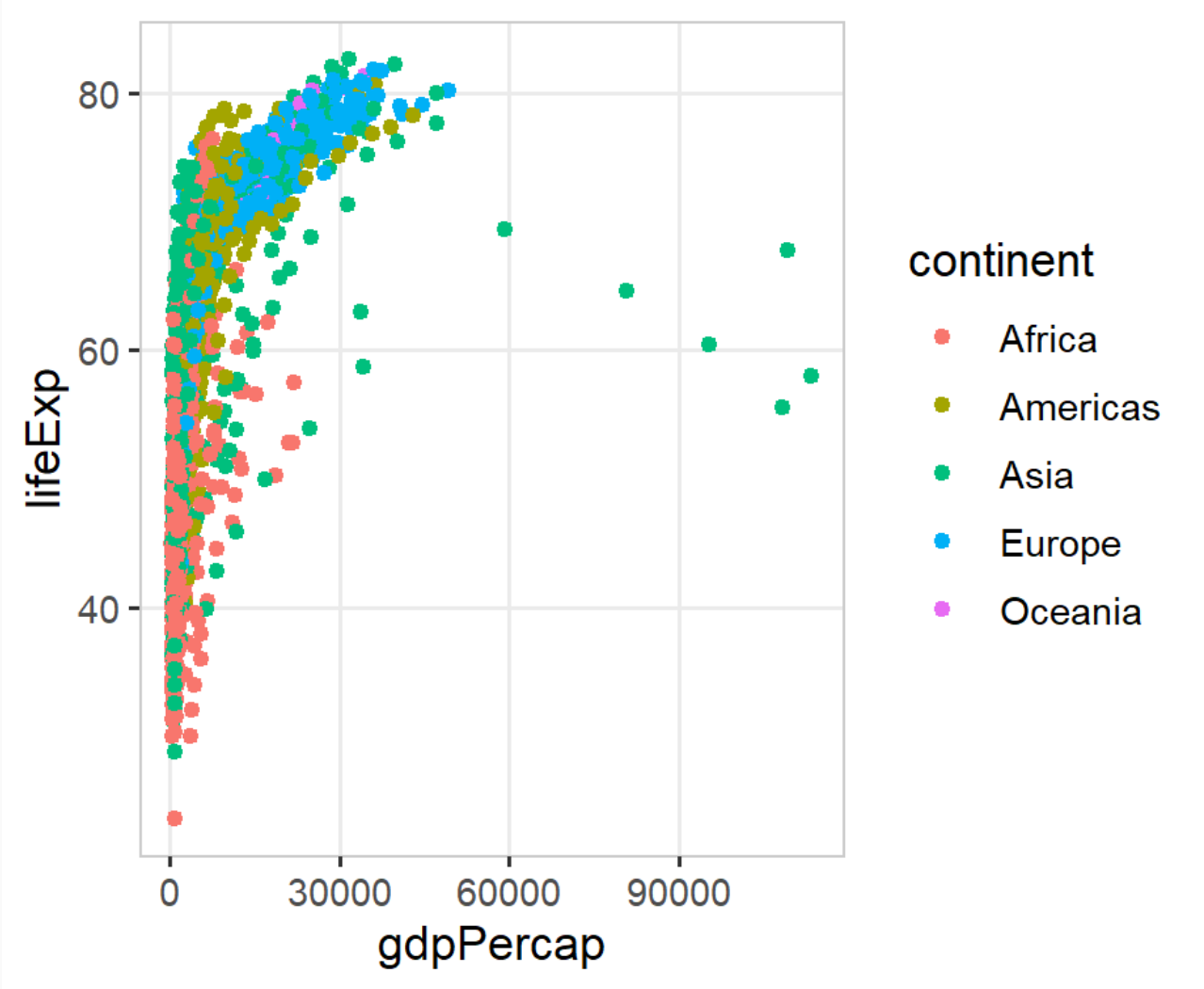
Add points...

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



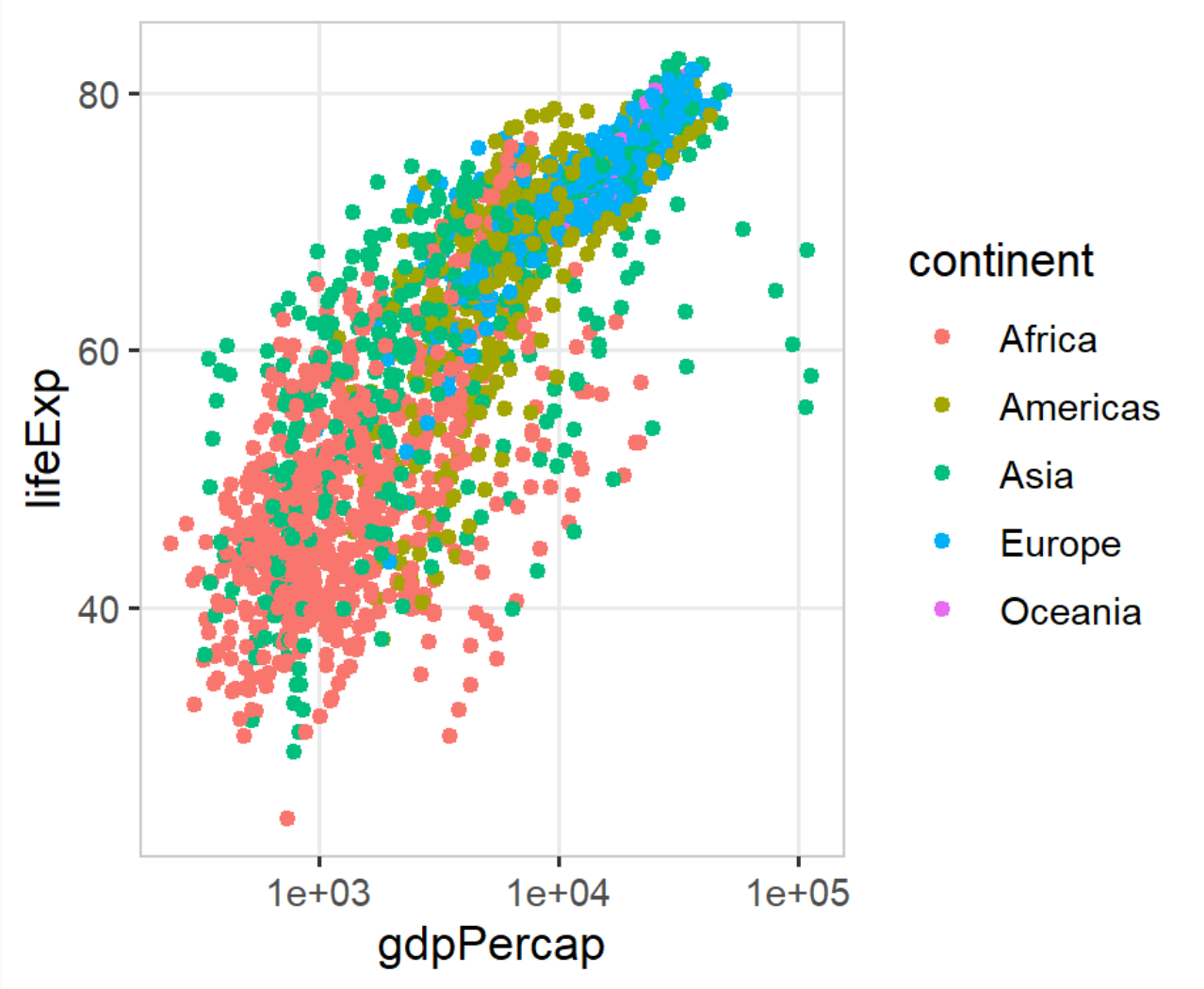
How can I tell countries apart?

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



GDP is squished together on the left

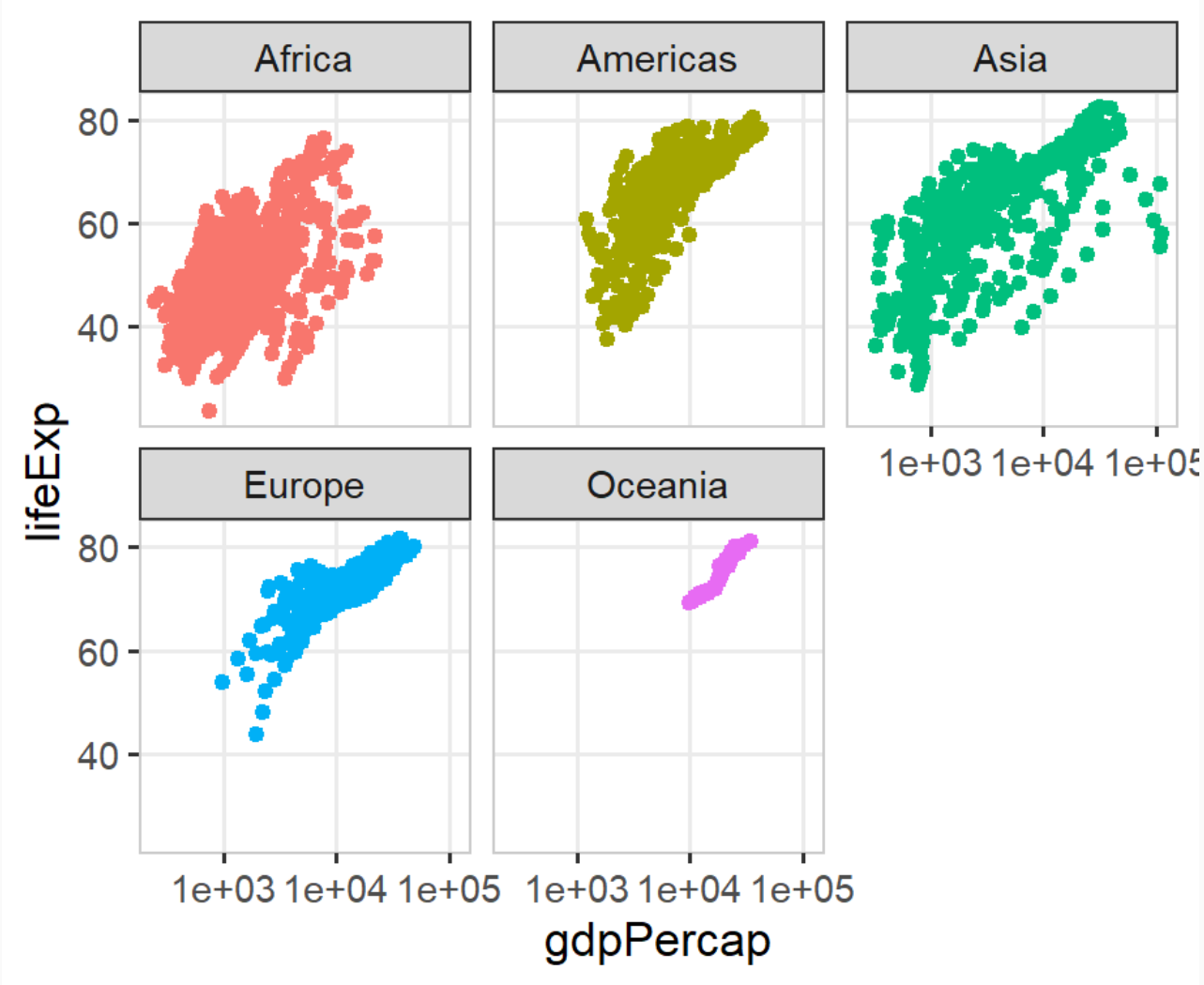
```
ggplot(gapminder,
  aes(x = gdpPercap,
      y = lifeExp,
      color = continent)) +
  geom_point() +
  scale_x_log10()
```



Still lots of overlap in the countries...

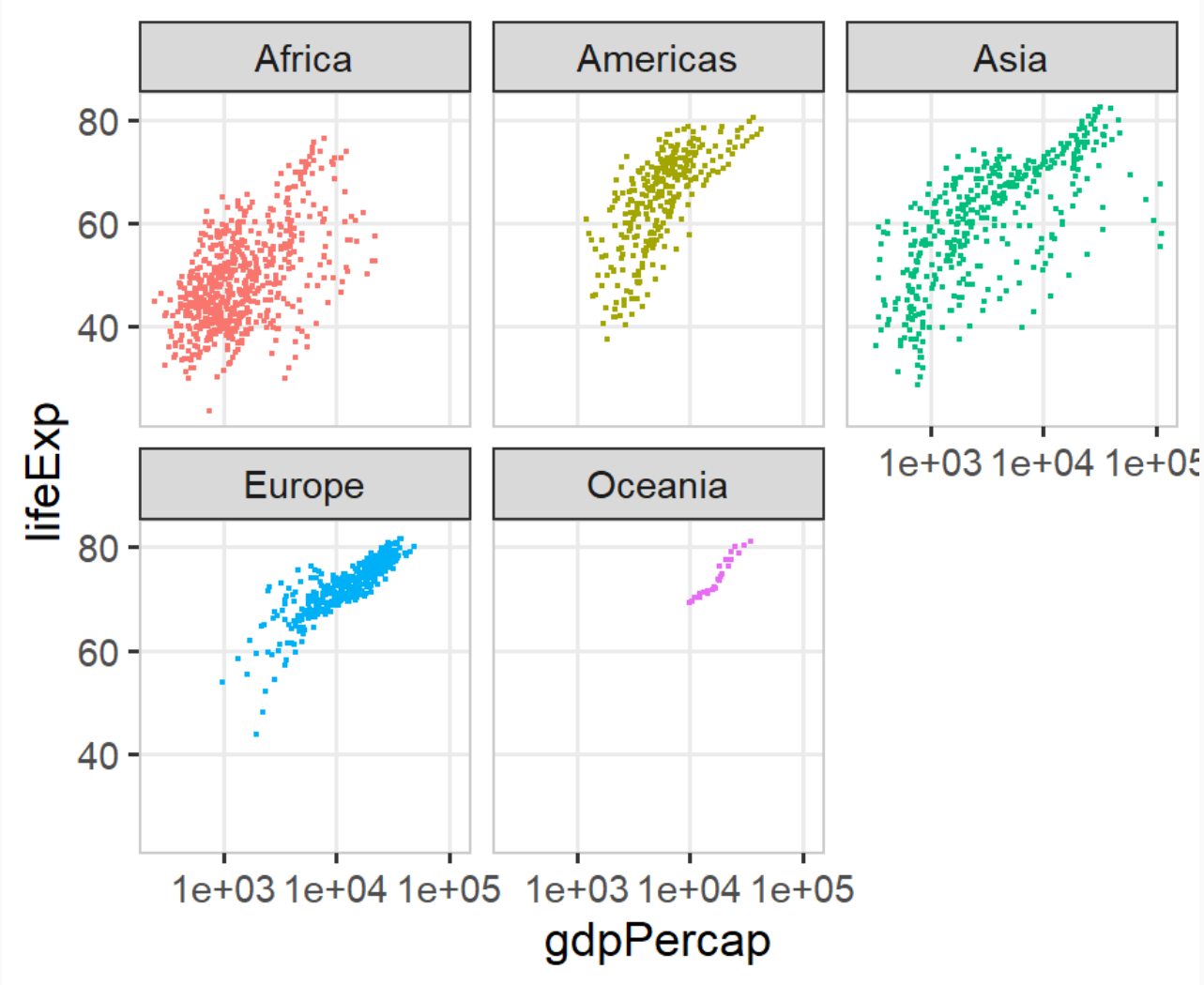
```
ggplot(gapminder,
  aes(x = gdpPercap,
      y = lifeExp,
      color = continent)) +
  geom_point() +
  scale_x_log10() +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```

No need for color legend thanks to facet titles



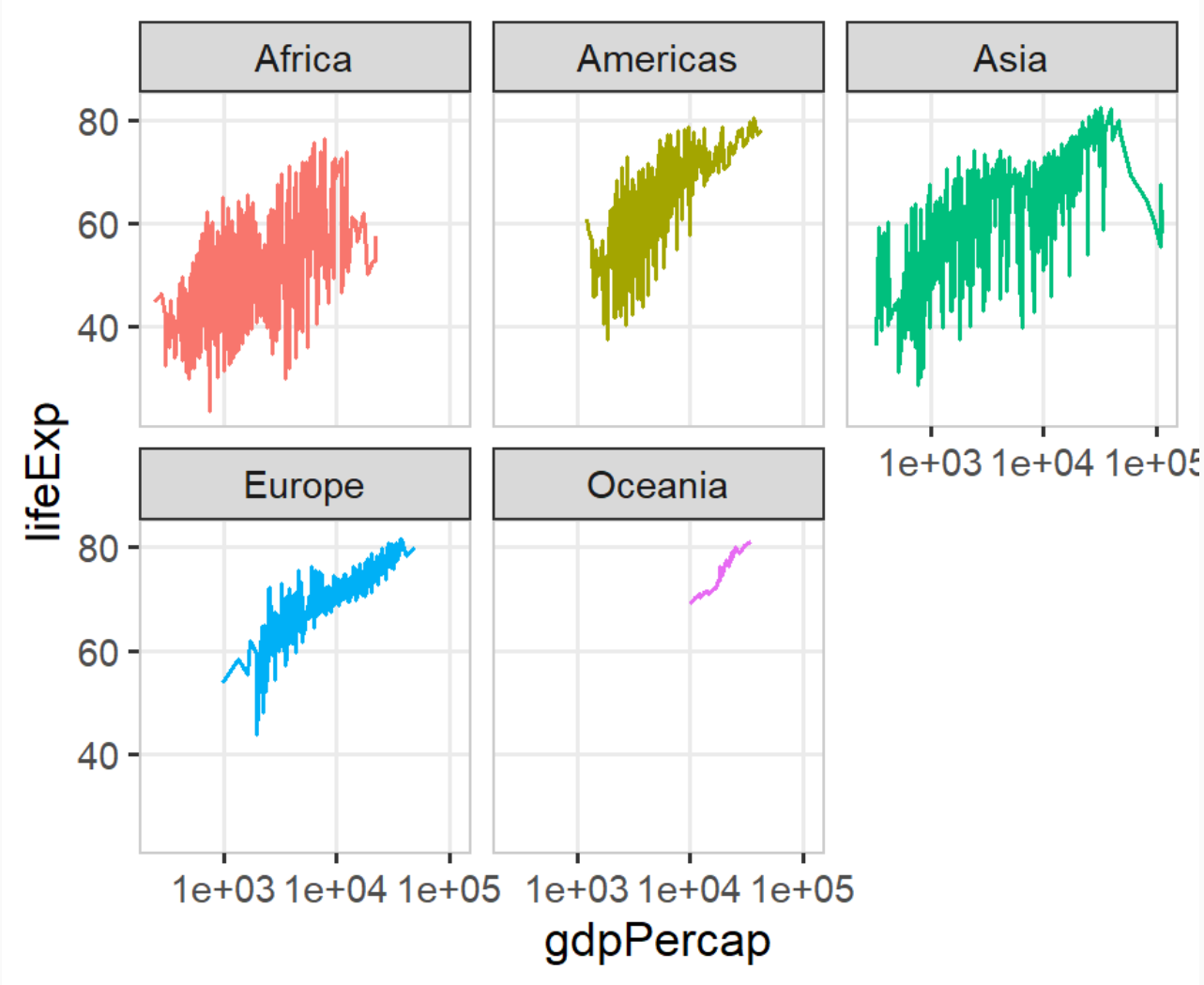
Lots of overplotting due to point size

```
ggplot(gapminder,
  aes(x = gdpPercap,
      y = lifeExp,
      color = continent)) +
  geom_point(size = 0.25) +
  scale_x_log10() +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```



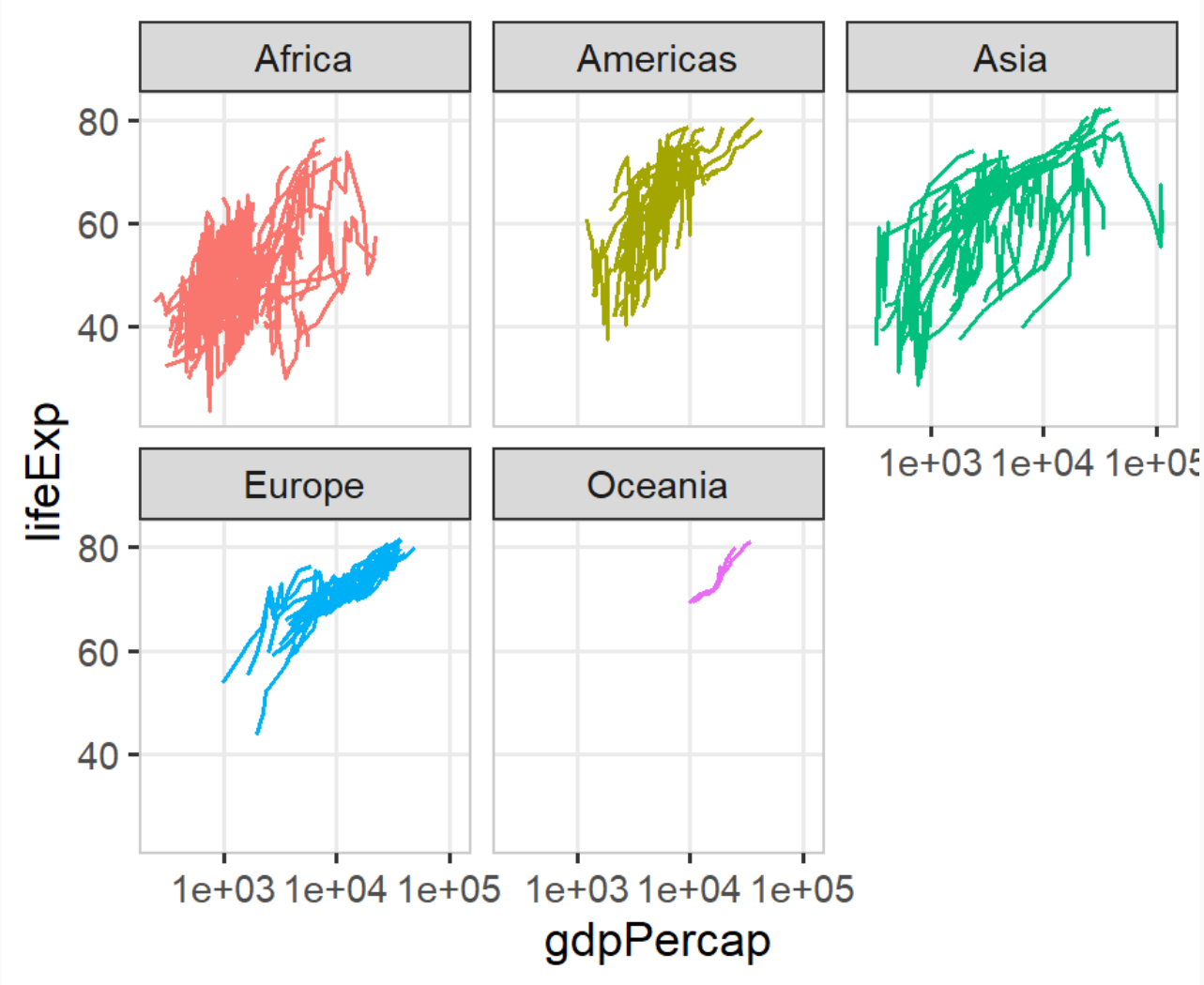
Is there a trend?

```
ggplot(gapminder,
  aes(x = gdpPercap,
      y = lifeExp,
      color = continent)) +
  geom_line() +
  #geom_point(size = 0.25) +
  scale_x_log10() +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```



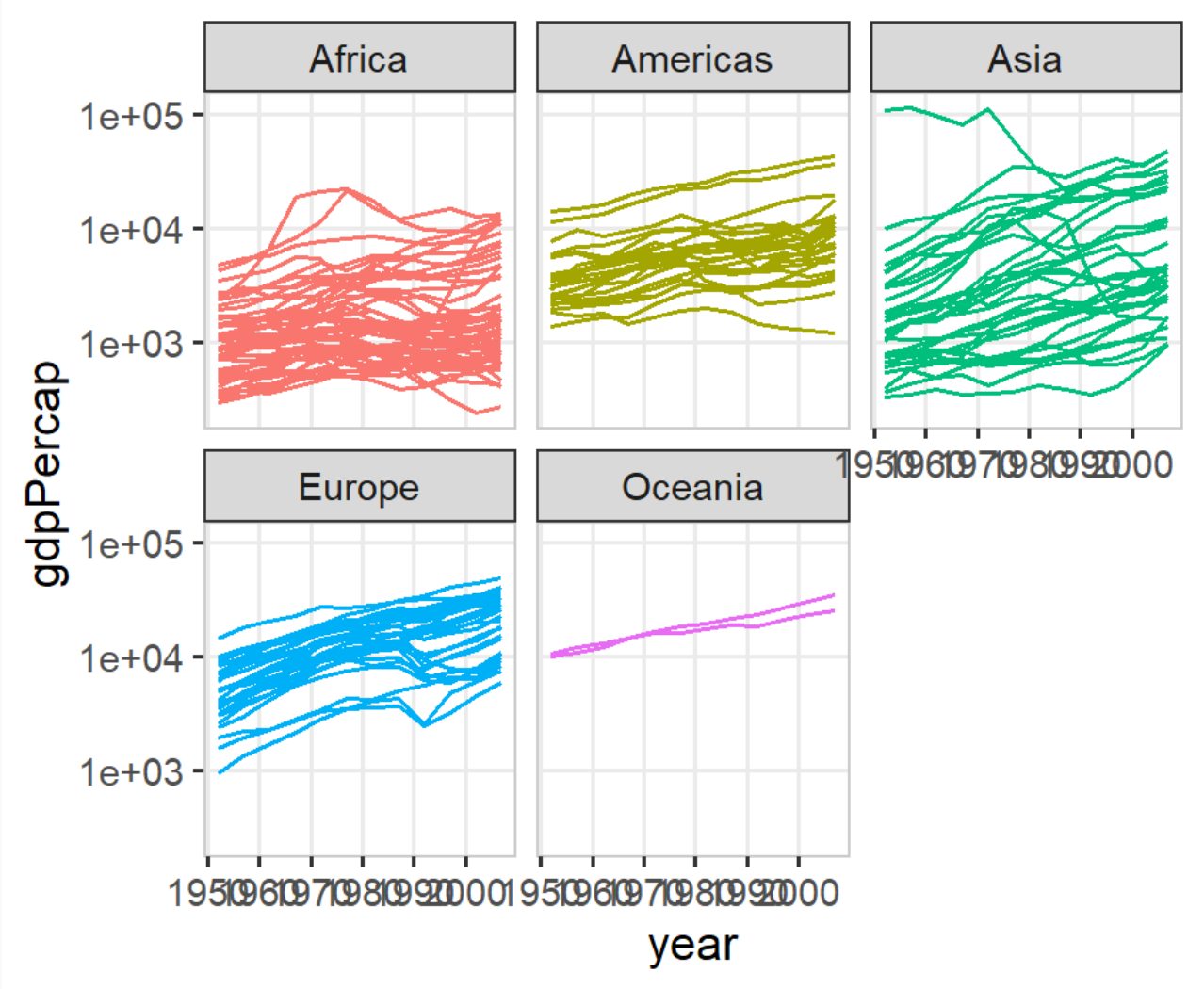
Okay, that line just connected all of the points sequentially...

```
ggplot(gapminder,
  aes(x = gdpPercap,
      y = lifeExp,
      color = continent)) +
  geom_line(
    aes(group = country)
    ) +
  #geom_point(size = 0.25) +
  scale_x_log10() +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```



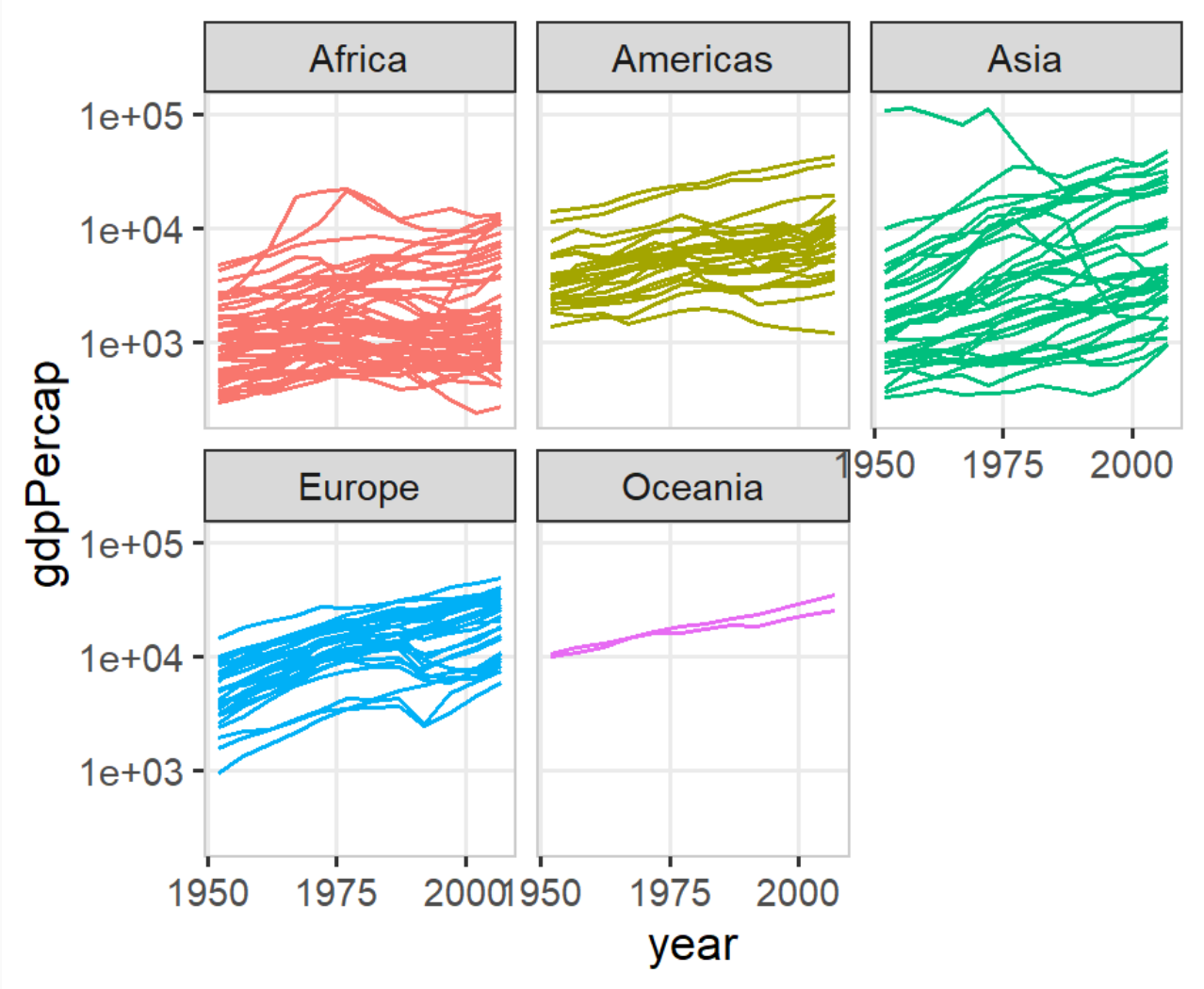
We need time on x-axis!

```
ggplot(gapminder,
  aes(x = year,
      y = gdpPercap,
      color = continent)) +
  geom_line(
    aes(group = country)) +
  #geom_point(size = 0.25) +
  scale_y_log10() +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```



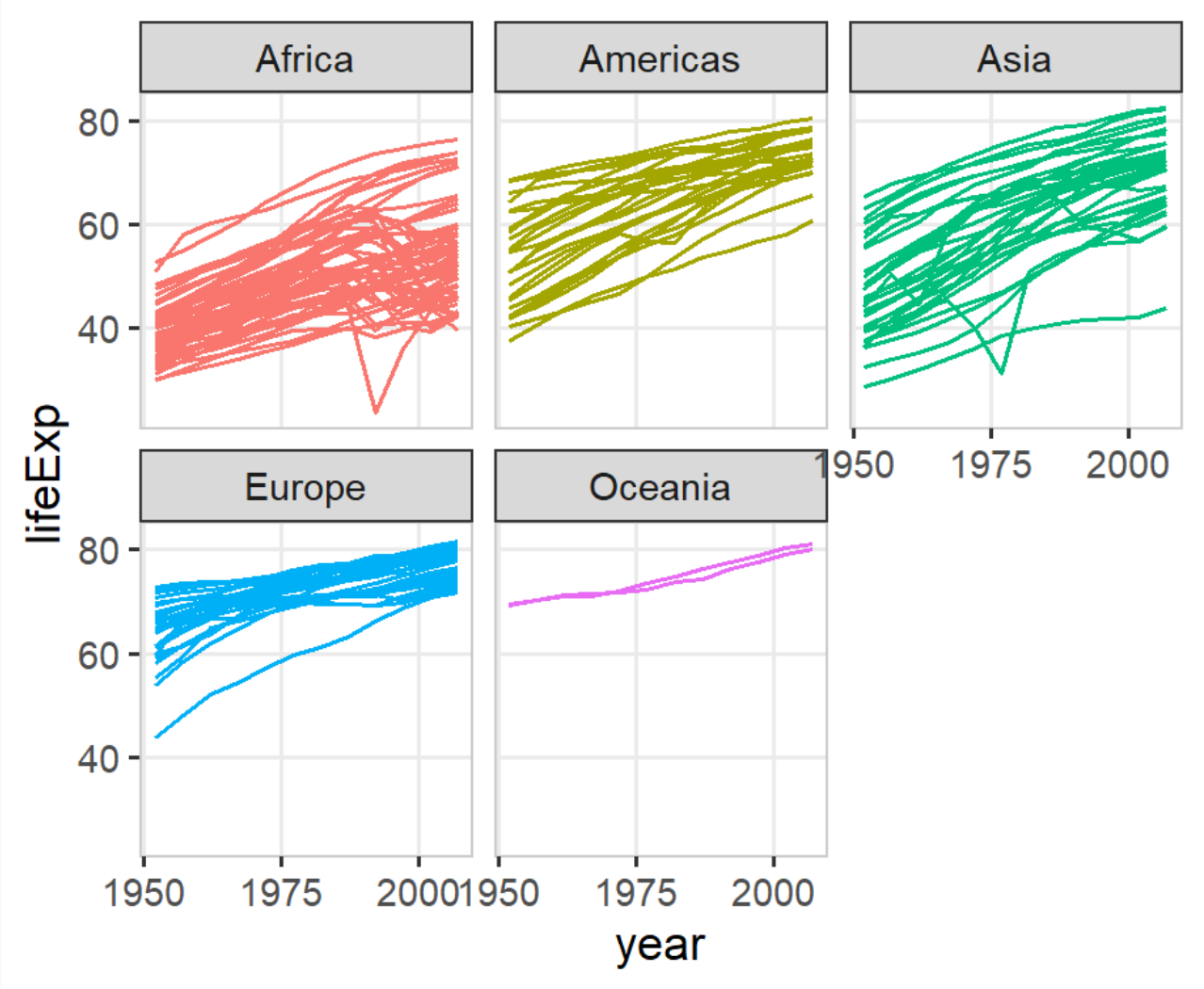
Can't see x-axis labels, though

```
ggplot(gapminder,
  aes(x = year,
      y = gdpPercap,
      color = continent)) +
  geom_line(
    aes(group = country)) +
  #geom_point(size = 0.25) +
  scale_y_log10() +
  scale_x_continuous(breaks =
    seq(1950, 2000, 25)
  ) +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```



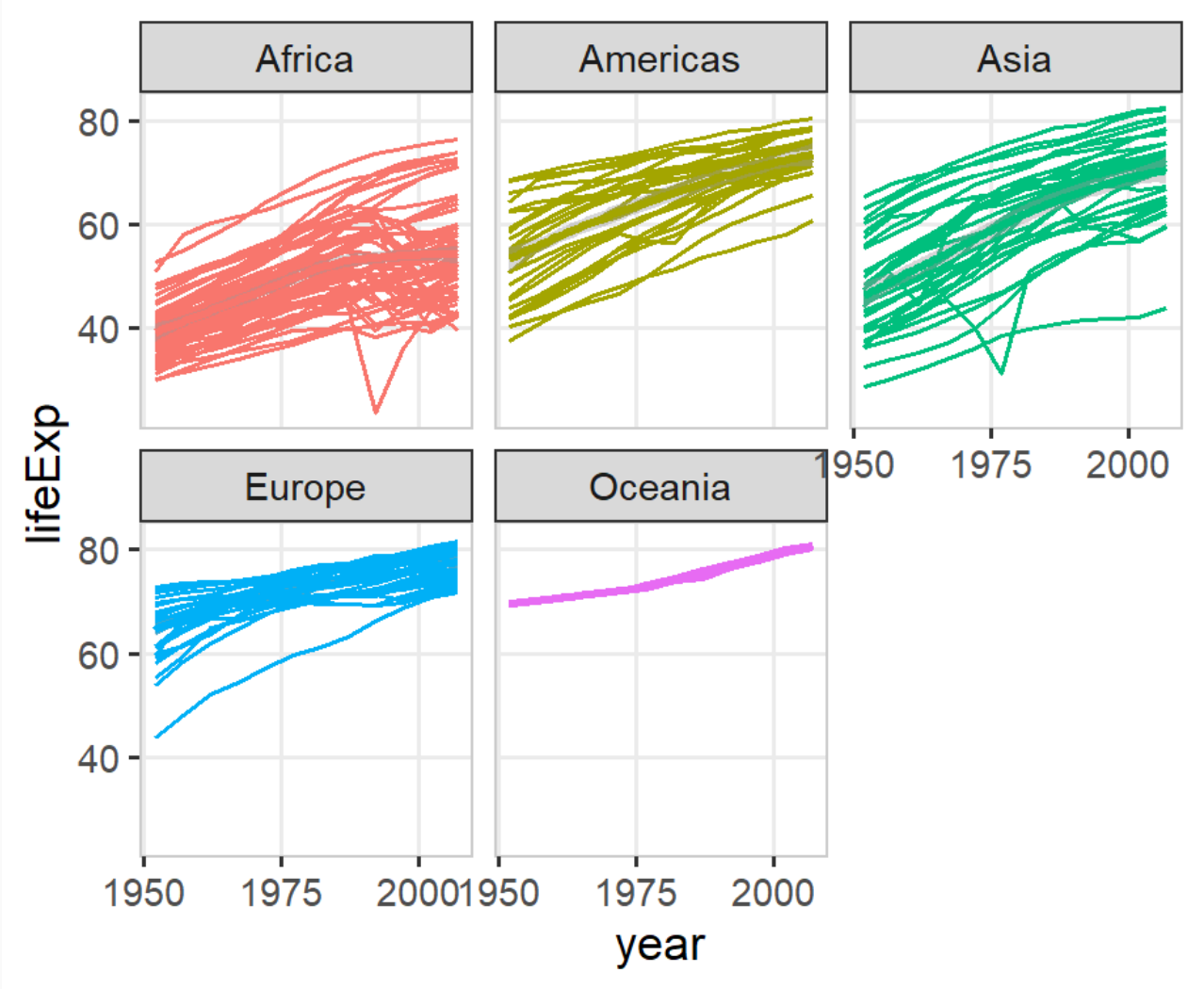
What about life expectancy?

```
ggplot(gapminder,
  aes(x = year,
      y = lifeExp,
      color = continent)) +
  geom_line(
    aes(group = country)) +
  #geom_point(size = 0.25) +
  #scale_y_log10() +
  scale_x_continuous(breaks =
    seq(1950, 2000, 25)) +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```



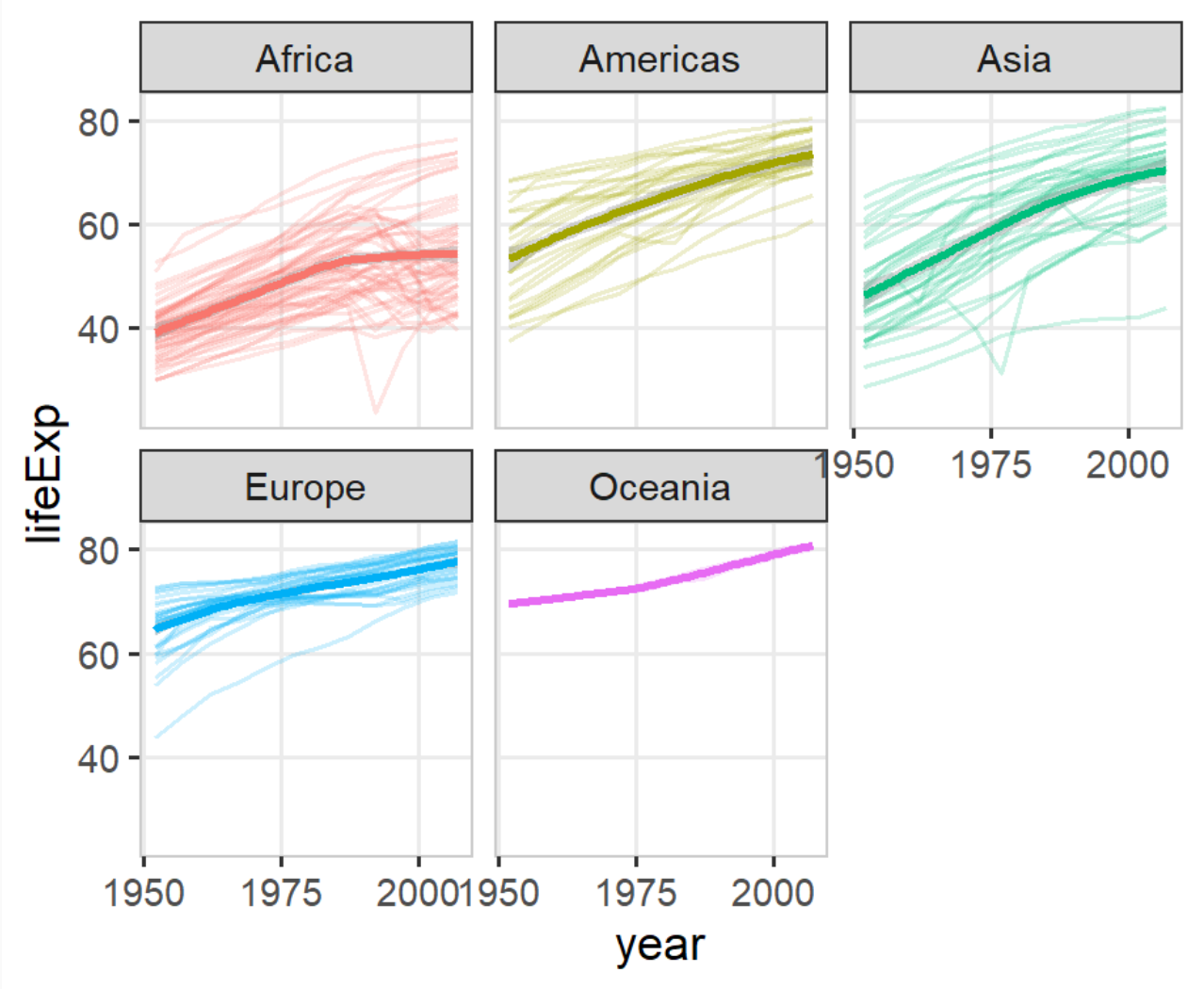
Okay, let's add a trend line

```
ggplot(gapminder,
  aes(x = year,
      y = lifeExp,
      color = continent)) +
  geom_line(
    aes(group = country)) +
  # geom_point(size = 0.25) +
  geom_smooth() +
  scale_x_continuous(breaks =
    seq(1950, 2000, 25)) +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```



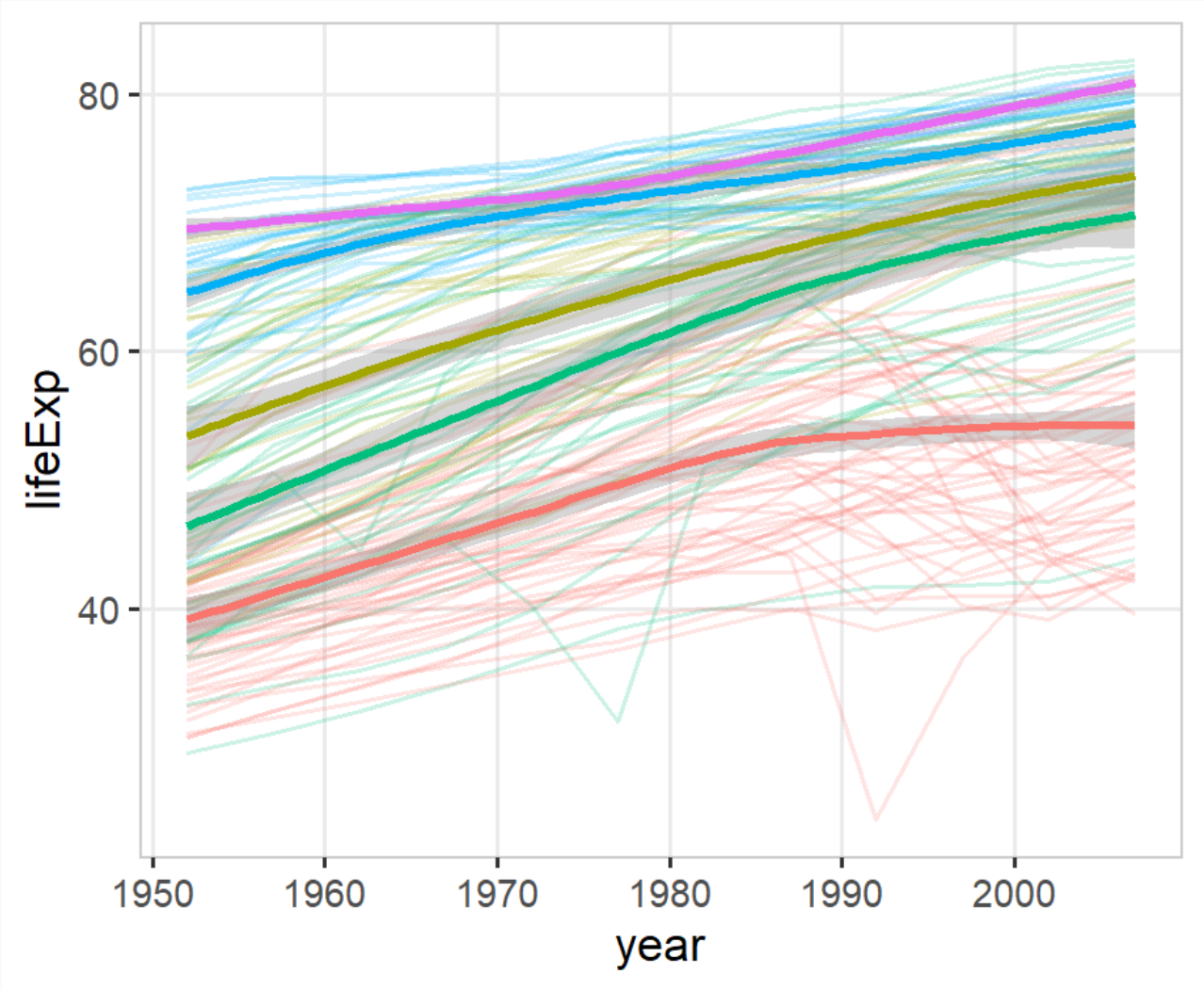
De-emphasize individual countries

```
ggplot(gapminder,
  aes(x = year,
      y = lifeExp,
      color = continent)) +
  geom_line(
    aes(group = country),
    alpha = 0.2) +
  #geom_point(size = 0.25) +
  geom_smooth() +
  scale_x_continuous(breaks =
    seq(1950, 2000, 25)) +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```



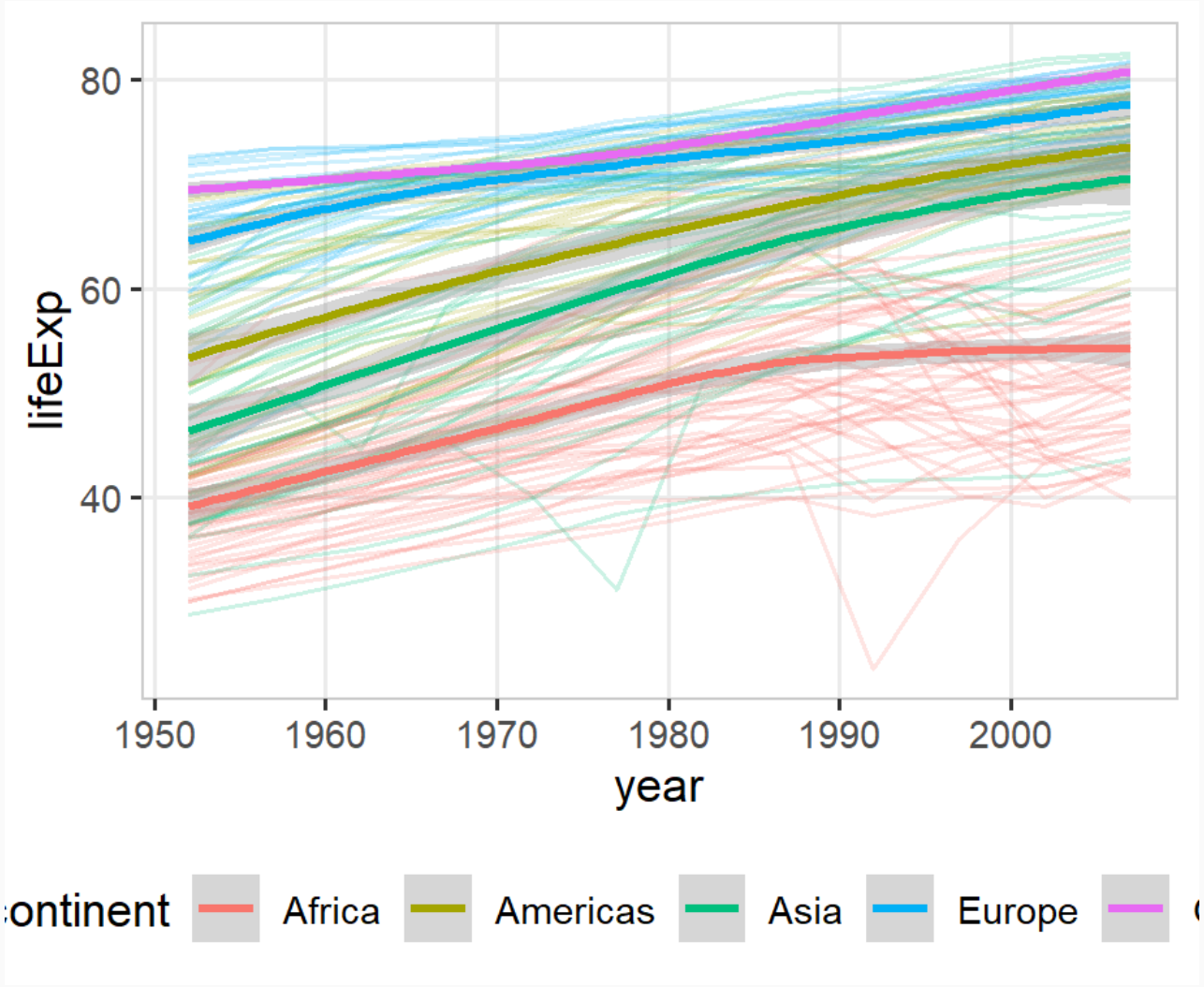
Let's compare continents

```
ggplot(gapminder,
  aes(x = year,
      y = lifeExp,
      color = continent)) +
  geom_line(
    aes(group = country),
    alpha = 0.2) +
  geom_smooth() +
  # scale_x_continuous(
  #   breaks =
  #     seq(1950, 2000, 25))+
  # facet_wrap(~ continent) +
  guides(color = FALSE)
```



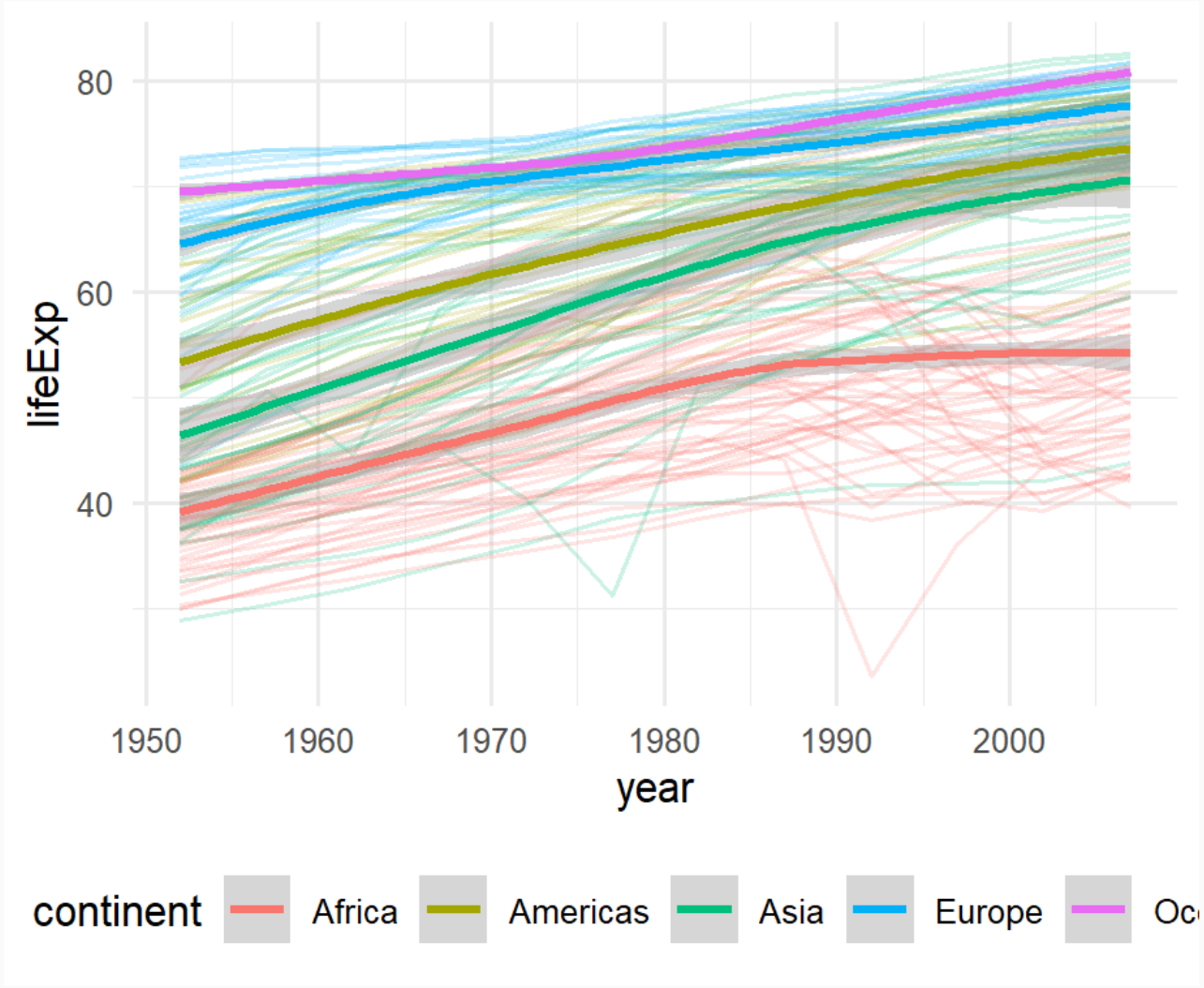
Wait, what color is each continent?

```
ggplot(gapminder,
  aes(x = year,
      y = lifeExp,
      color = continent)) +
  geom_line(
    aes(group = country),
    alpha = 0.2) +
  geom_smooth() +
  theme(
  legend.position = "bottom")
```



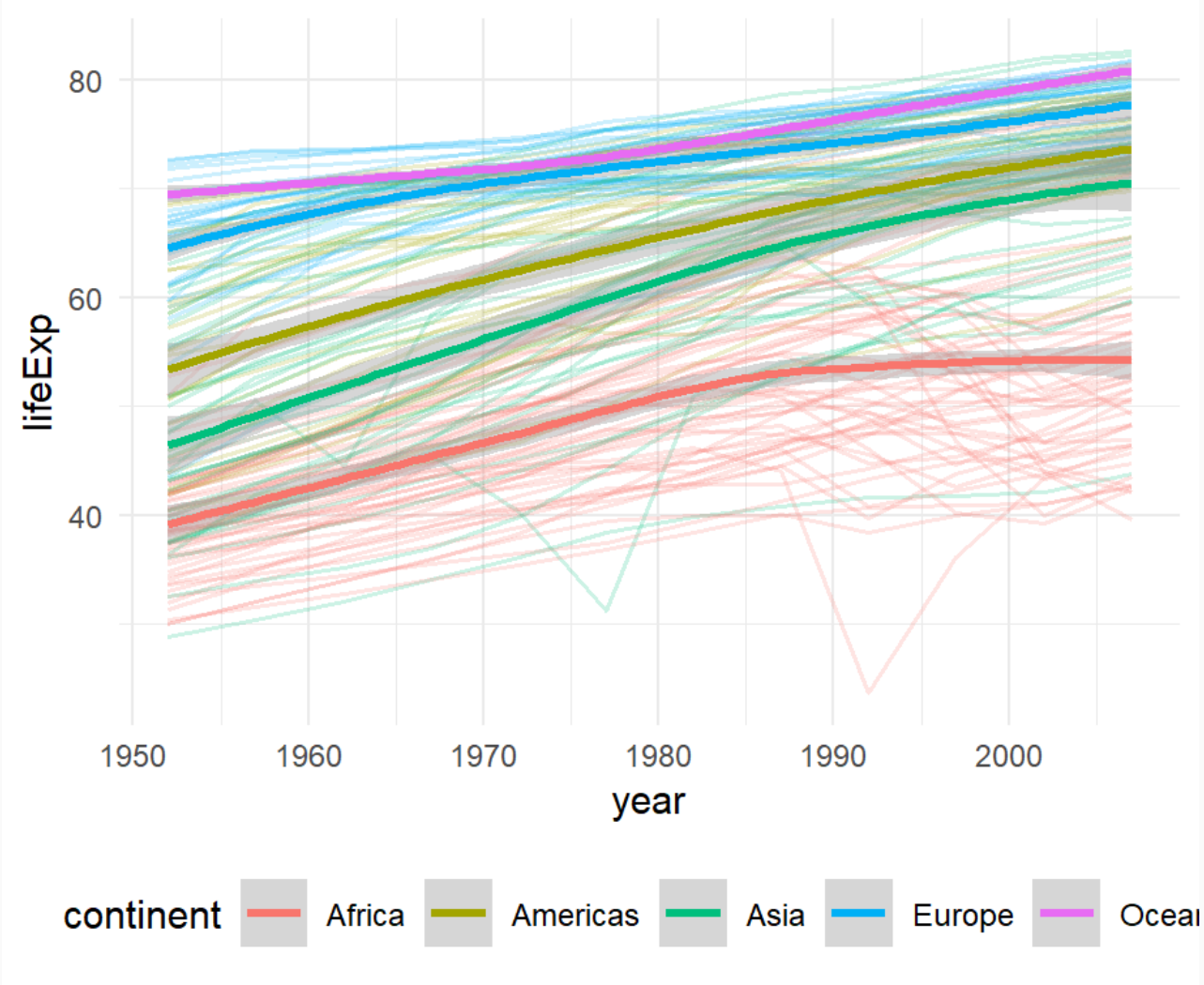
Let's try the minimal theme

```
ggplot(gapminder,
  aes(x = year,
      y = lifeExp,
      color = continent)) +
  geom_line(
    aes(group = country),
    alpha = 0.2) +
  geom_smooth() +
  theme_minimal() +
  theme(
  legend.position = "bottom")
```



Fonts are kind of big

```
ggplot(gapminder,
  aes(x = year,
      y = lifeExp,
      color = continent)) +
  geom_line(
    aes(group = country),
    alpha = 0.2) +
  geom_smooth() +
  theme_minimal(
    base_size = 10) +
  theme(
  legend.position = "bottom")
```



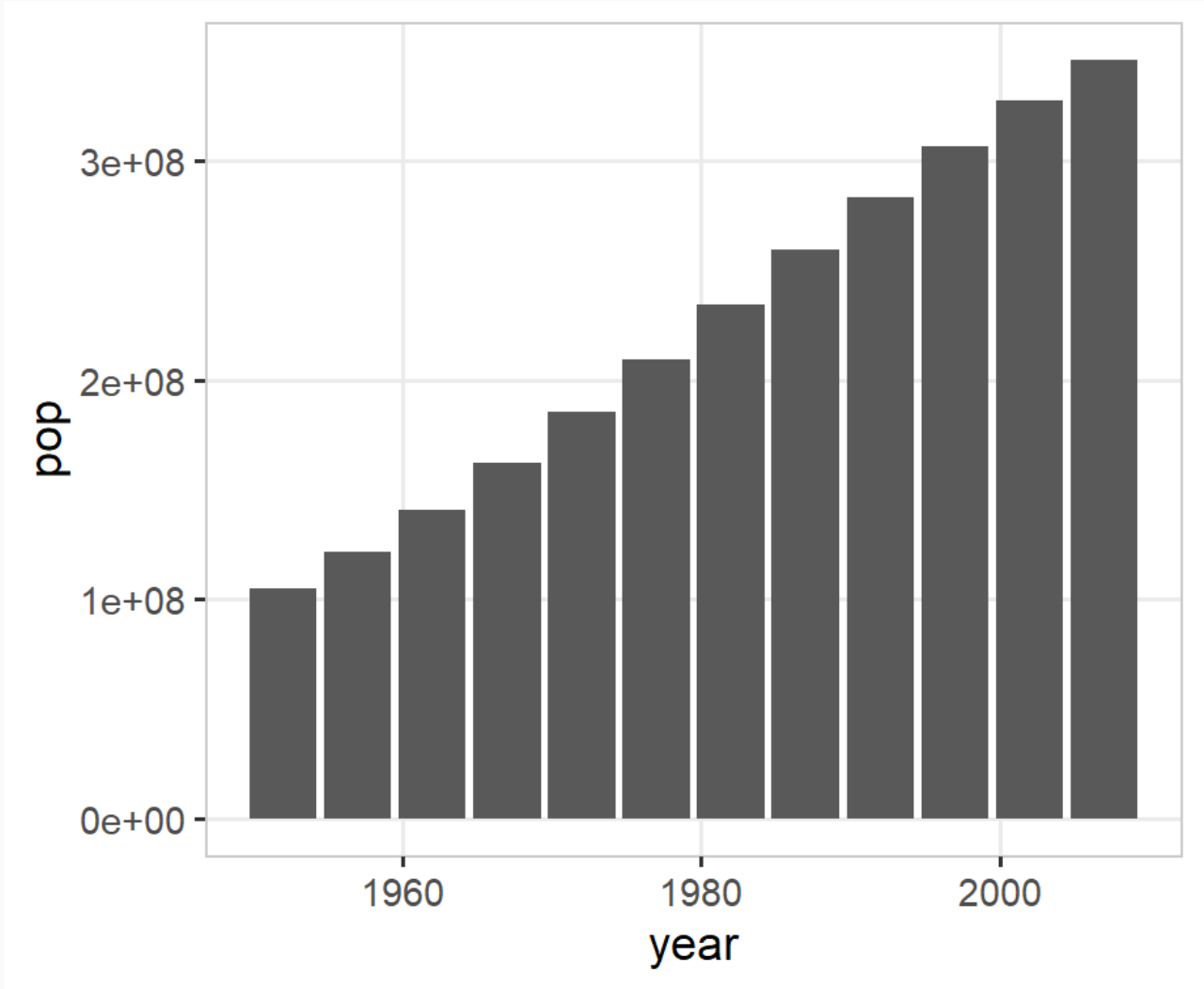
Great, let's switch gears

```
americas ←
  gapminder %>%
  filter(
    country %in% c(
      "Brazil",
      "Canada",
      "Mexico",
      "Ecuador"
    )
  )
```

Let's look at four countries in more detail. How do their populations compare to each other?

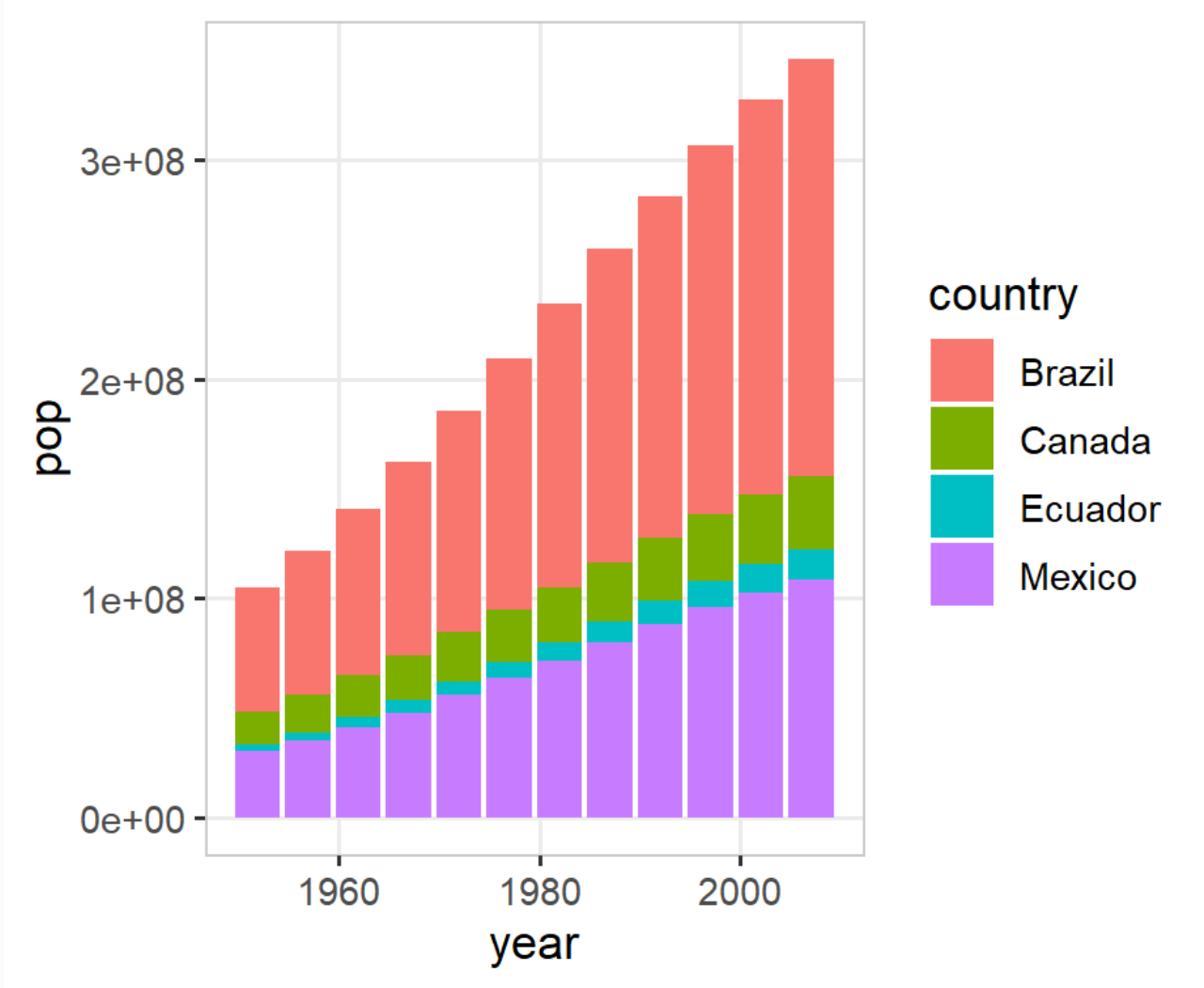
country	continent	year	lifeExp	pop	gdpPercap
Brazil	Americas	1952	50.917	56602560	2108.944
Brazil	Americas	1957	53.285	65551171	2487.366
Brazil	Americas	1962	55.665	76039390	3336.586
Brazil	Americas	1967	57.632	88049823	3429.864
Brazil	Americas	1972	59.504	100840058	4985.711
Brazil	Americas	1977	61.489	114313951	6660.119
Brazil	Americas	1982	63.336	128962939	7030.836
Brazil	Americas	1987	65.205	142938076	7807.096
Brazil	Americas	1992	67.057	155975974	6950.283
Brazil	Americas	1997	69.388	168546719	7957.981

```
ggplot(americas,
  aes(x = year,
      y = pop)
  ) +
  geom_col()
```



Yeah, but how many people are in each country?

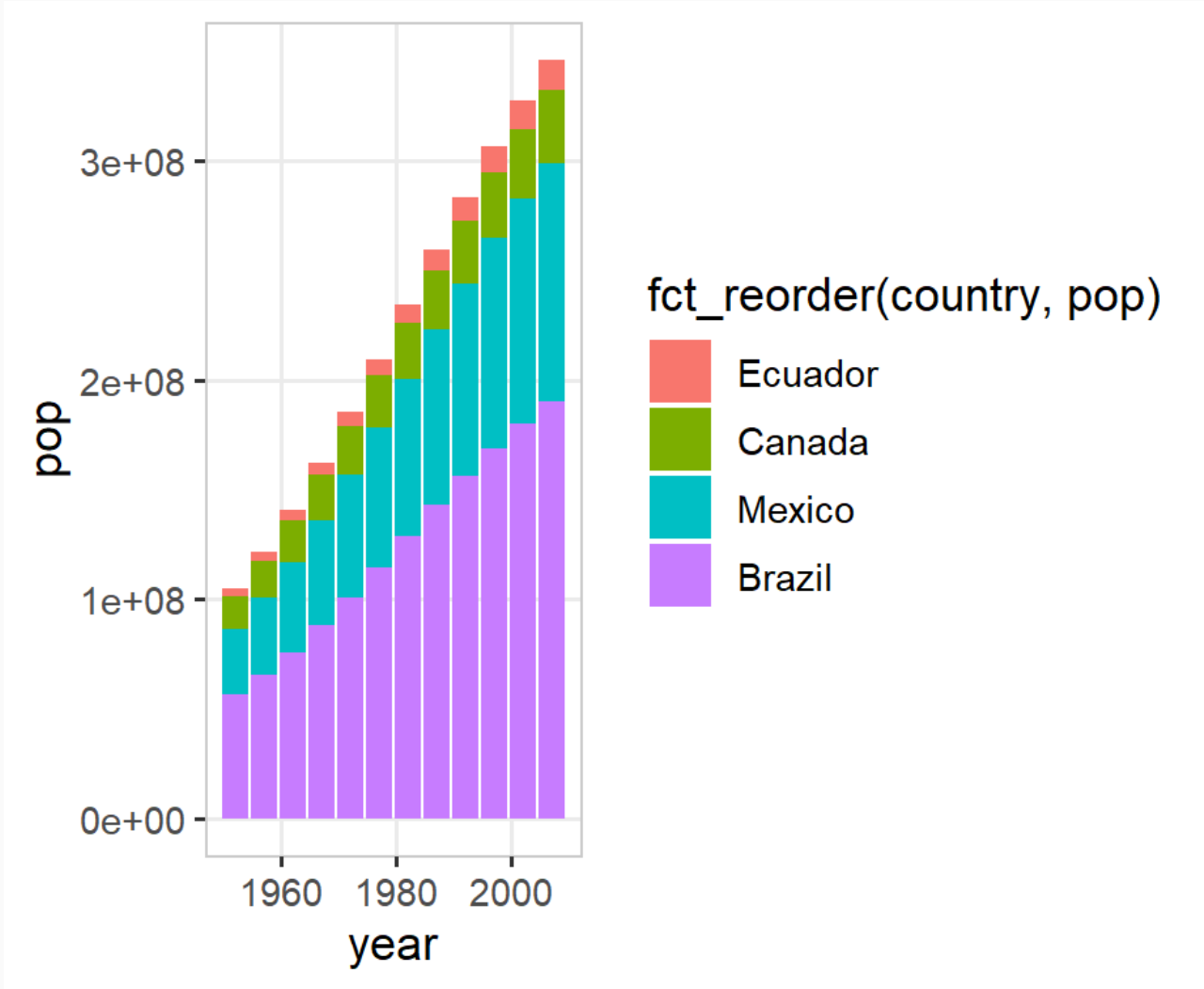
```
ggplot(americas,
  aes(x = year,
      y = pop,
      fill = country)
  ) +
  geom_col()
```



Can we reorder by population size?

Excellent tutorial by Jenny Bryan "Be the Boss of your factors"

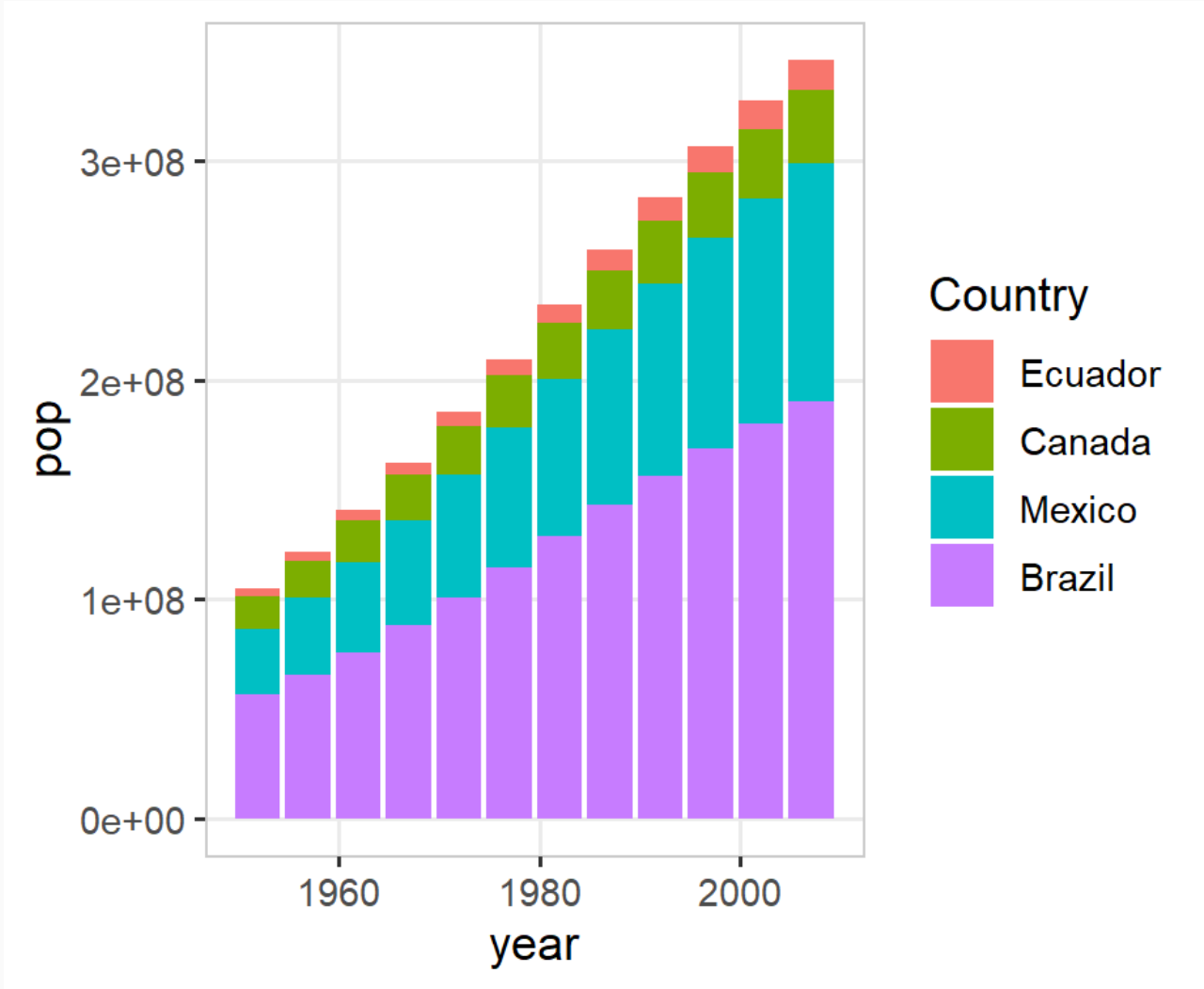
```
ggplot(americas,
  aes(x = year,
      y = pop,
      fill = fct_reorder(
        country, pop)
      )
  ) +
  geom_col()
```



Can we change the labels?

Excellent tutorial by Jenny Bryan "Be the Boss of your factors"

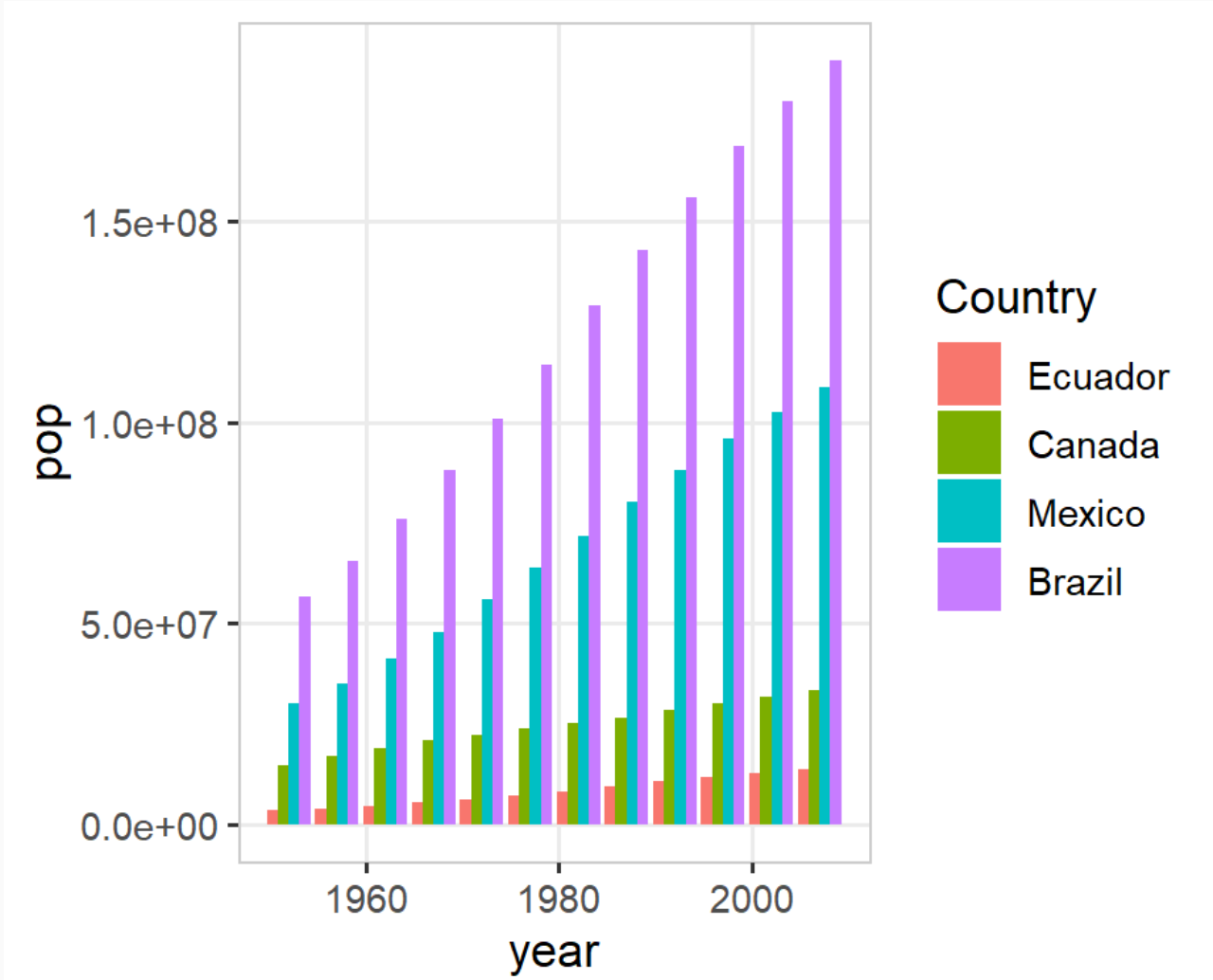
```
ggplot(americas,
  aes(x = year,
      y = pop,
      fill = fct_reorder(
        country, pop)
      )
  ) +
  geom_col() +
  labs(fill = "Country")
```



Bars are "stacked", can we separate?

```
ggplot(americas,
  aes(x = year,
      y = pop,
      fill = fct_reorder(
        country, pop)
      )
  ) +
  geom_col(
    position = "dodge") +
  labs(fill = "Country")
```

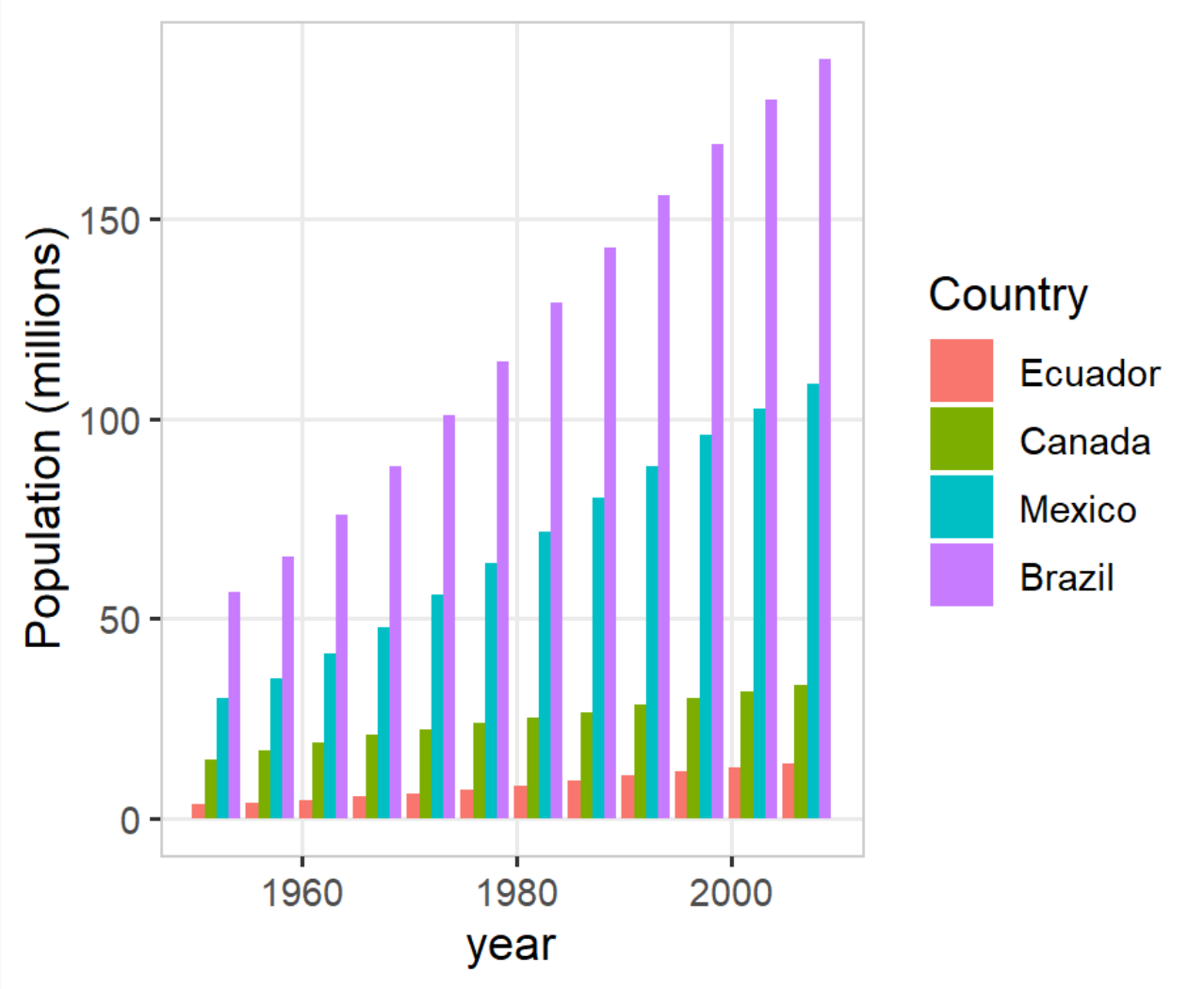
`position = "dodge"` places objects *next to each other* instead of overlapping



🤓 What is scientific notation anyway?

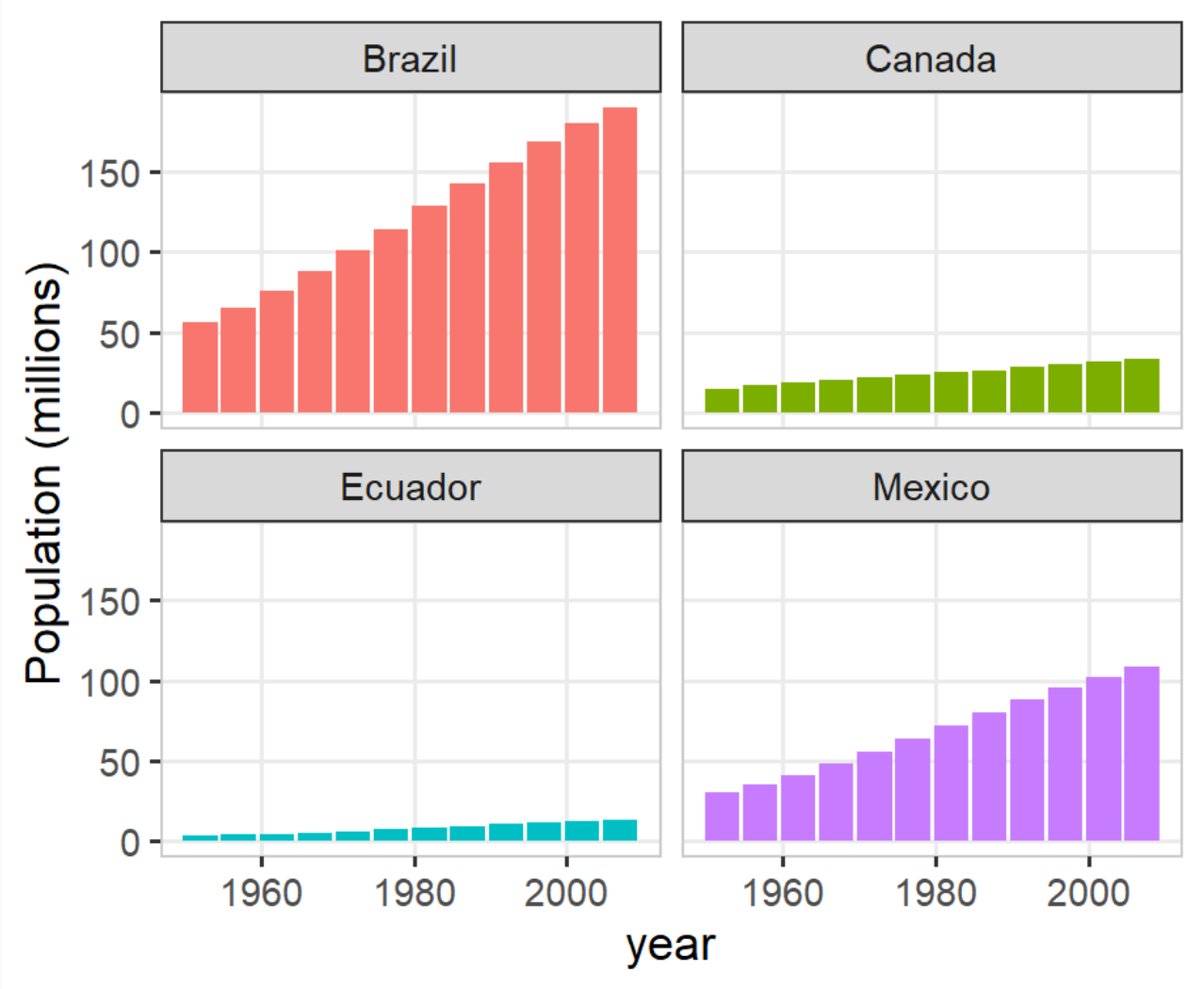
```
ggplot(americas,
  aes(x = year,
      y = pop / 10^6,
      fill = fct_reorder(
        country, pop)
      )
  ) +
  geom_col(
    position = "dodge"
  ) +
  labs(fill = "Country",
       y = "Population (millio
       )
```

ggplot aesthetics can take expressions!



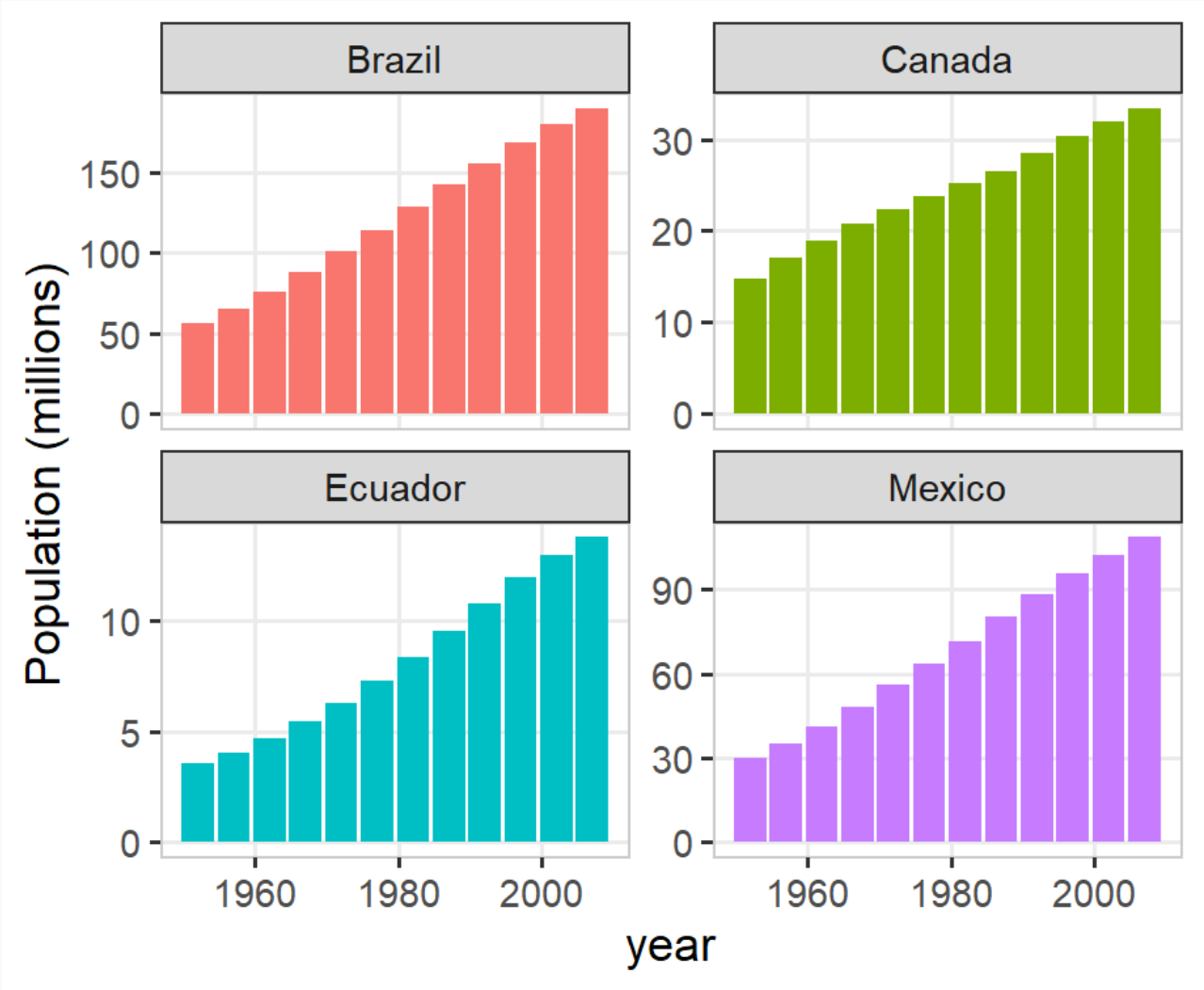
Might be easier to see countries individually

```
.ot(americas,
es(x = year,
   y = pop / 10^6,
   fill = country)
 +
eom_col(
 position = "dodge"
 +
abs(y = "Population (millions)")
acet_wrap(~ country) +
uides(fill = FALSE)
```



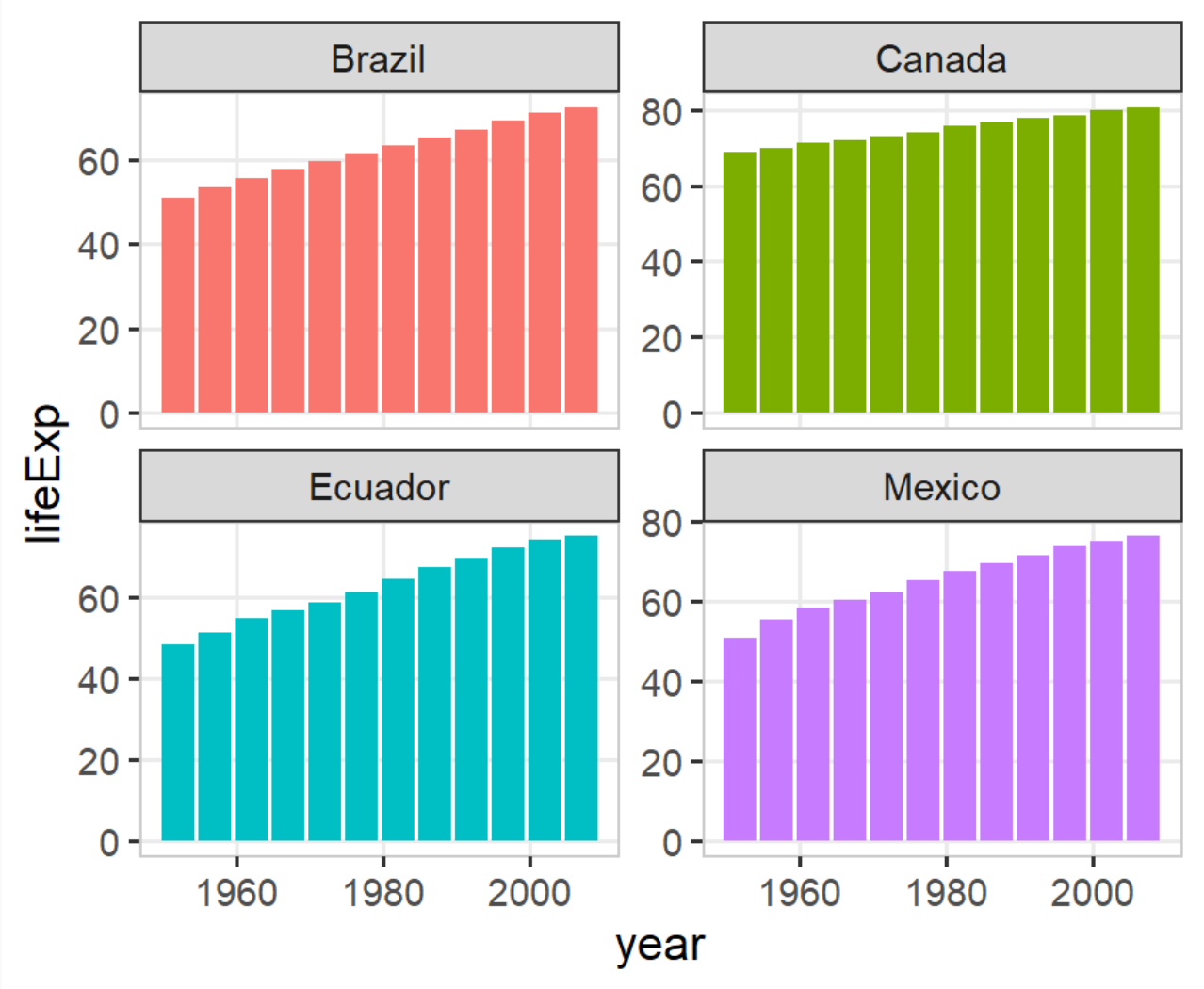
Let range of y-axis vary in each plot

```
ggplot(americas,
  aes(x = year,
      y = pop / 10^6,
      fill = country)
  ) +
  geom_col(
    position = "dodge"
  ) +
  labs(y = "Population (millior
  facet_wrap(~ country,
    scales = "free_y") +
  guides(fill = FALSE)
```



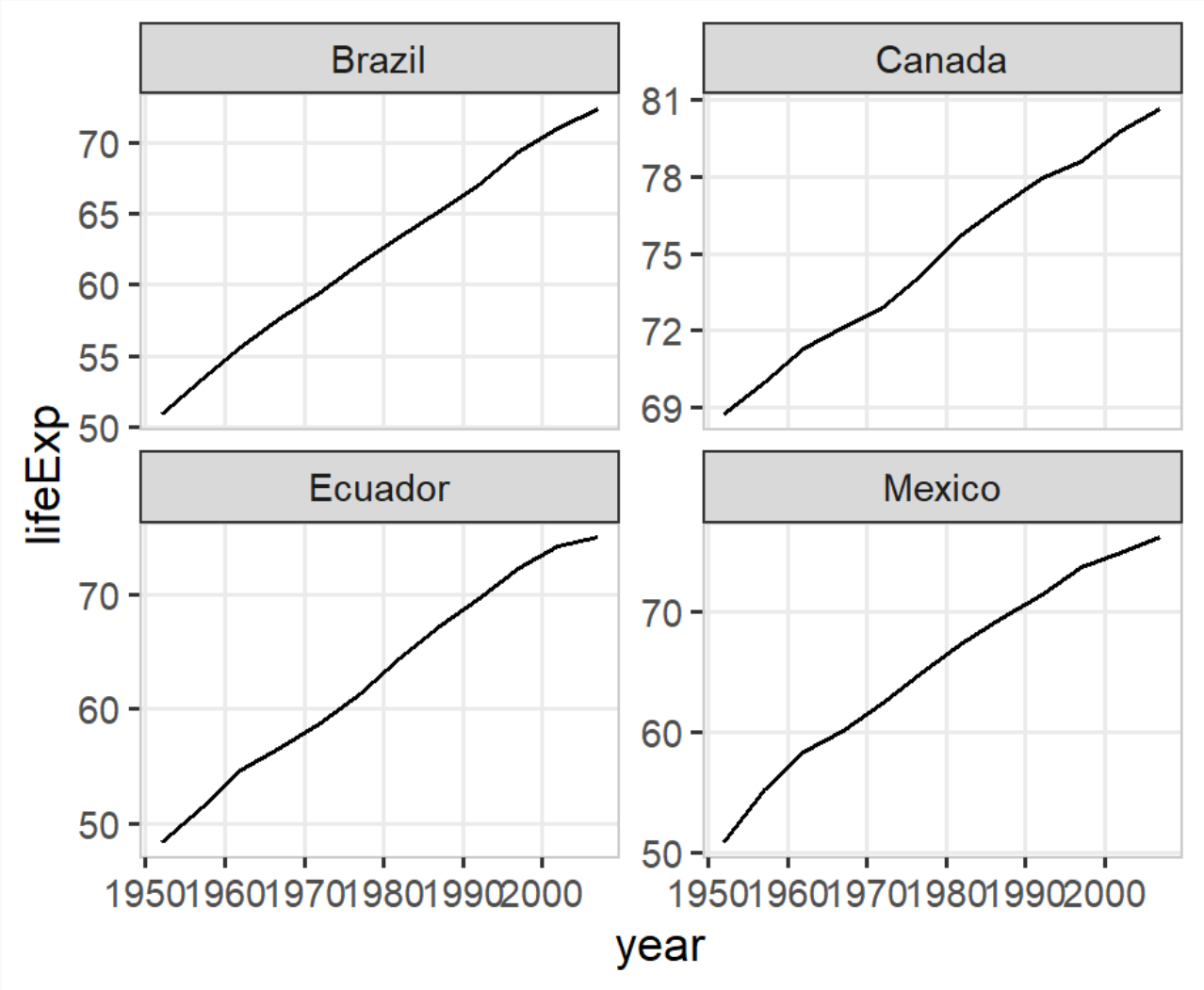
What about life expectancy again?

```
ggplot(americas,
  aes(x = year,
      y = lifeExp,
      fill = country)
  ) +
  geom_col(
    position = "dodge"
  ) +
  facet_wrap(~ country,
             scales = "free_y")
  guides(fill = FALSE)
```



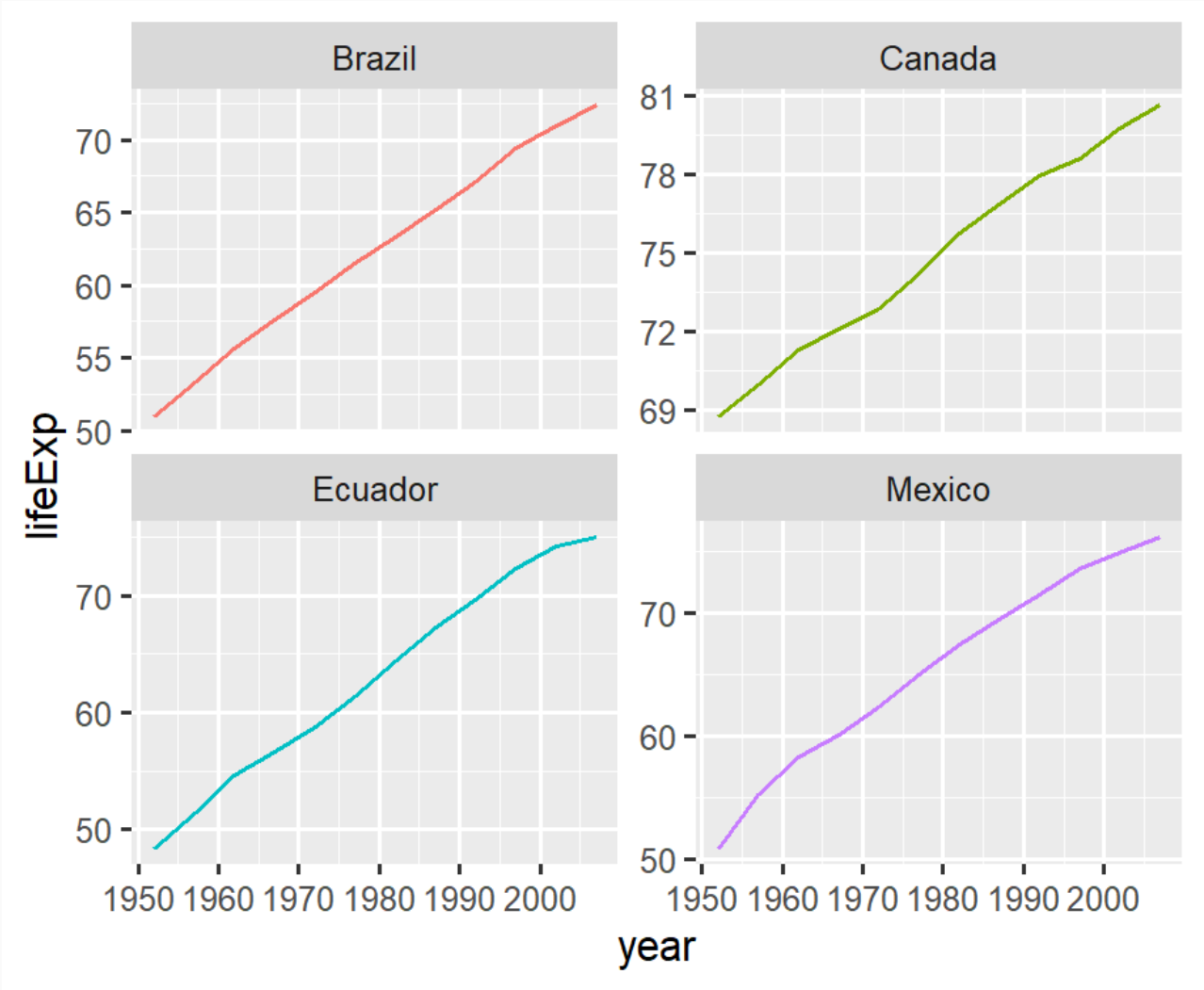
This should really be 📈 instead of 📊

```
ggplot(americas,
  aes(x = year,
      y = lifeExp,
      fill = country)
  ) +
  geom_line() +
  facet_wrap(~ country,
             scales = "free_y")
  guides(fill = FALSE)
```



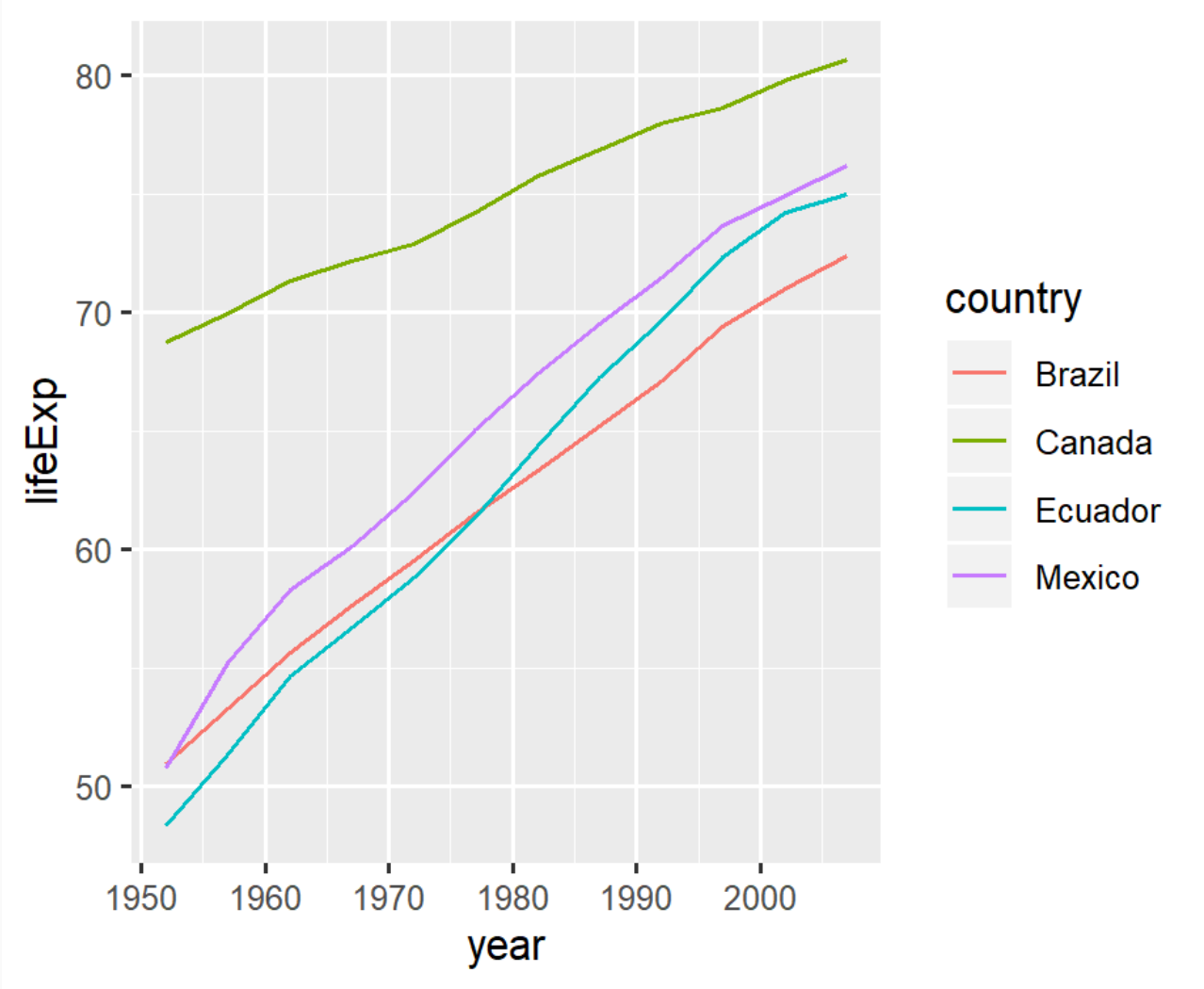
 are **fill**ed,  are **color**ed

```
ggplot(americas) +
  aes(
    x = year,
    y = lifeExp,
    color = country
  ) +
  geom_line() +
  facet_wrap(~ country,
    scales = "free_y") +
  guides(color = FALSE)
```



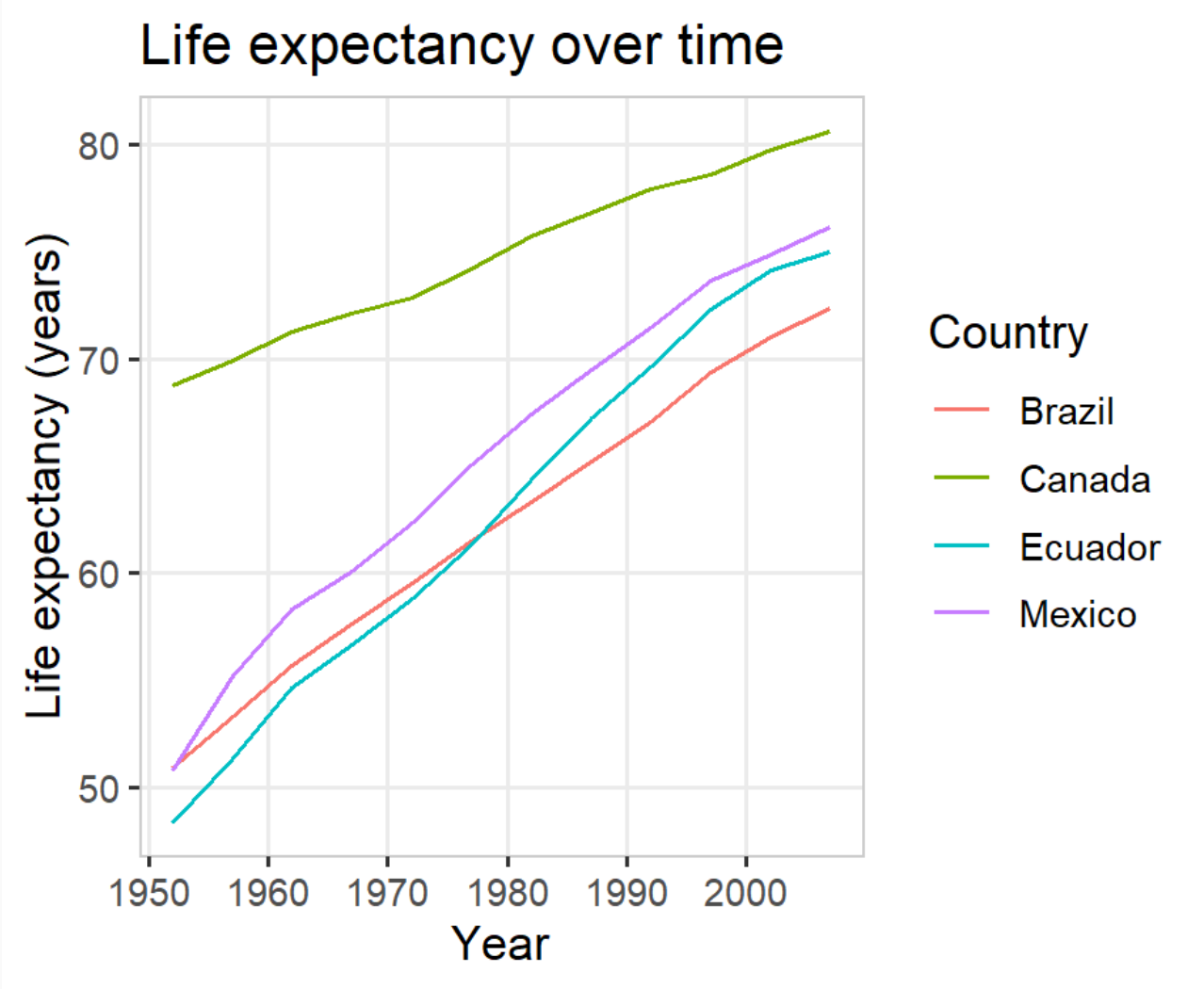
Altogether now!

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



Let's update the labels

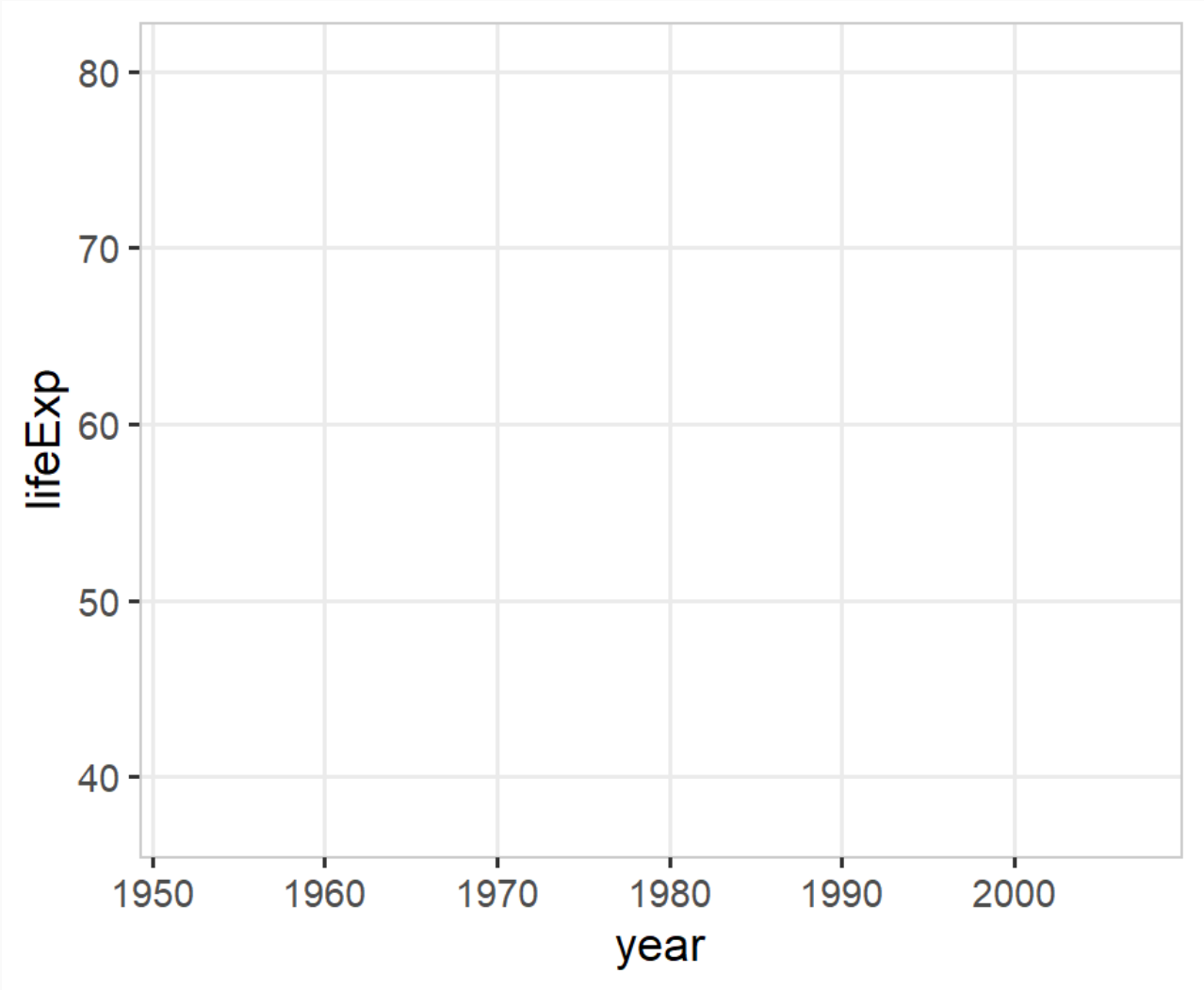
```
ggplot(americas,
  aes(x = year,
      y = lifeExp,
      color = country)
  ) +
  geom_line() +
  labs(x = "Year",
       y = "Life expectancy (ye
       color = "Country",
       title = "Life expectancy
```



Okay, changing gears again. What is range of life expectancy in Americas?

```
gapminder %>%
  filter(
    continent == "Americas"
  ) %>%
  ggplot(
    aes(x = year,
        y = lifeExp)
  )
```

You can pipe into `ggplot()`!
Just watch for `%>%` changing to `+`

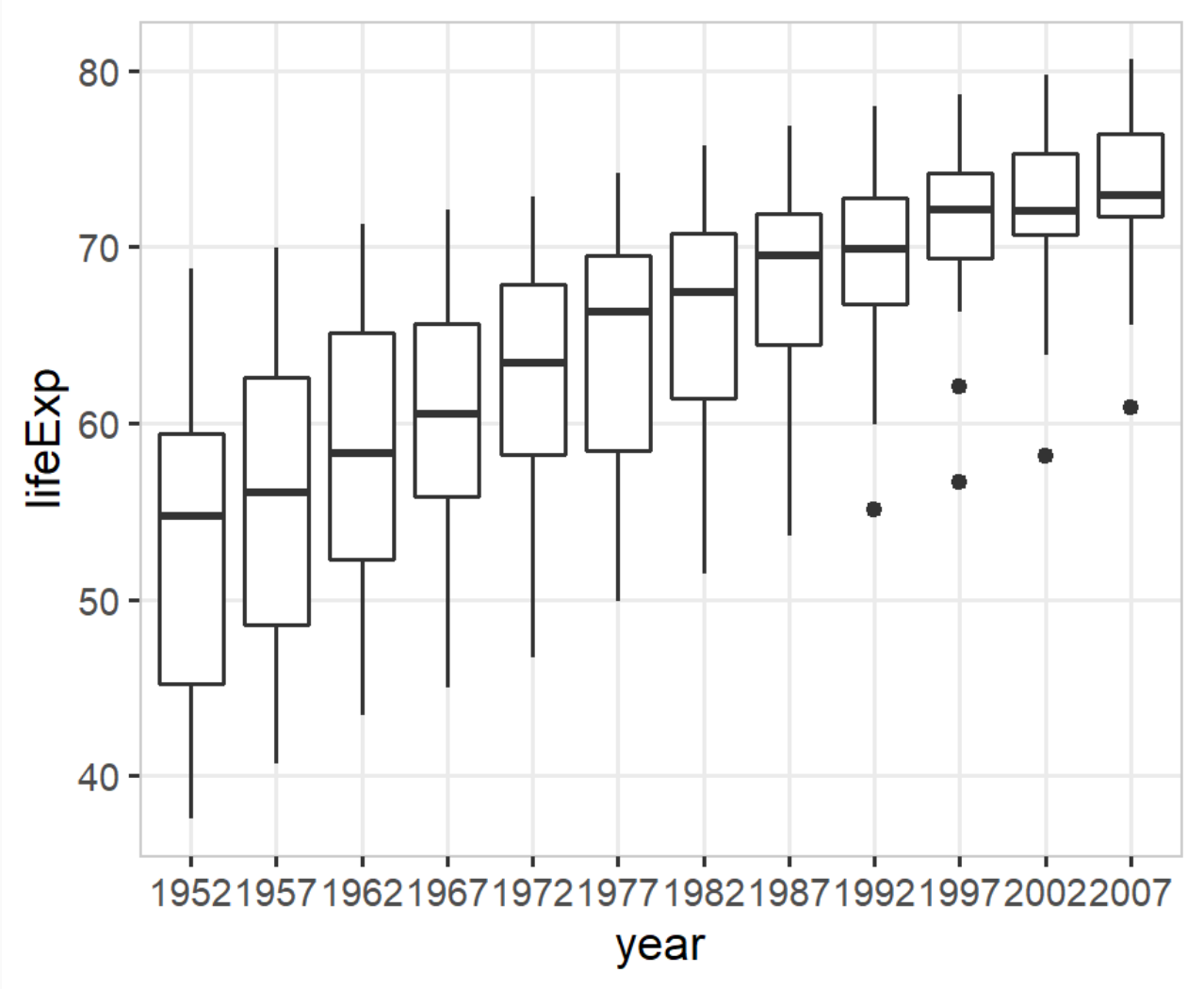


Boxplot for life expectancy range

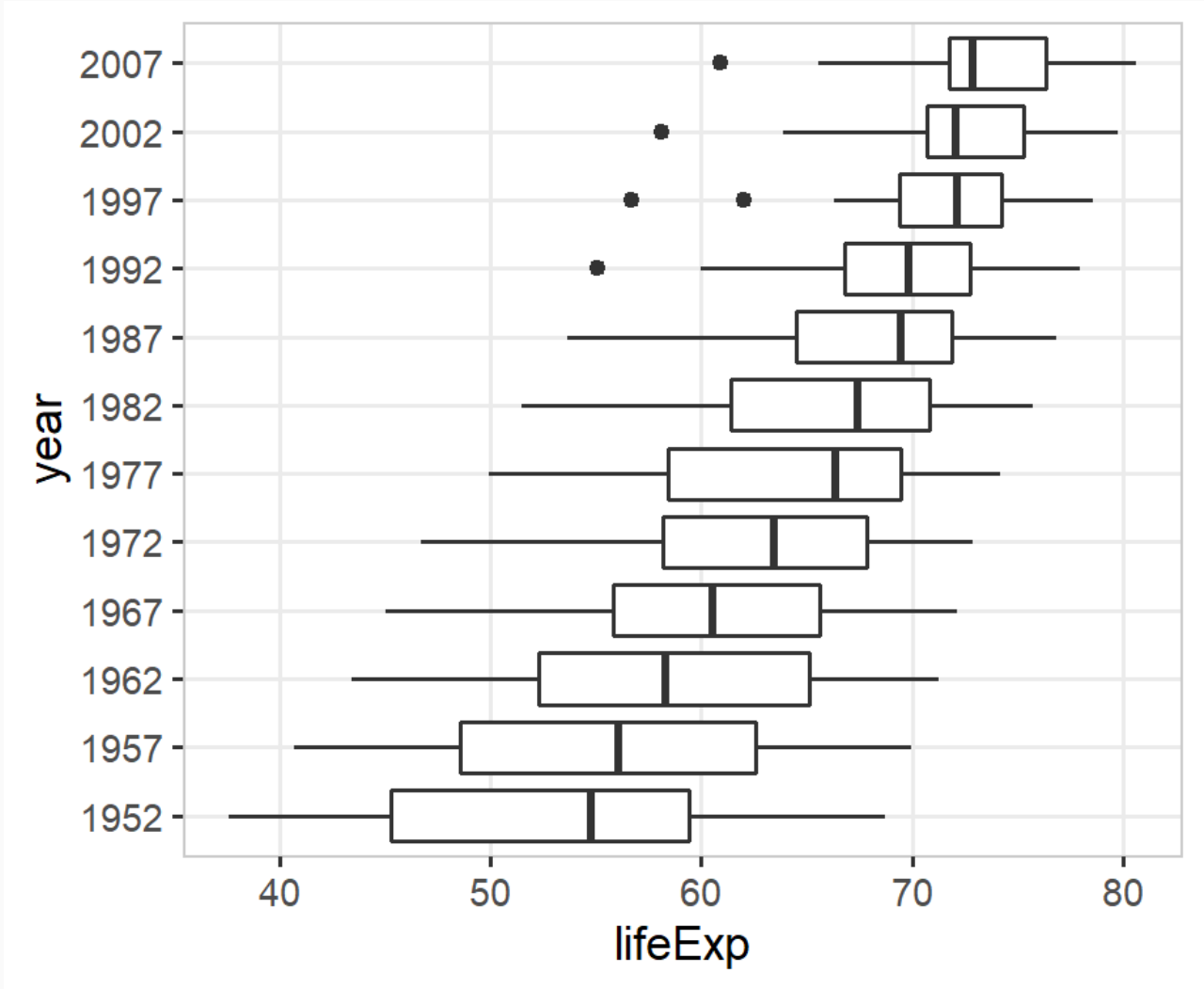
```
gapminder %>%
  filter(
    continent == "Americas"
  ) %>%
  ggplot(
    aes(x = year,
        y = lifeExp)
  ) +
  geom_boxplot()
```

Why not boxplots by year?

```
gapminder %>%
  filter(
    continent == "Americas"
  ) %>%
  mutate(
    year = factor(year)
  ) %>%
  ggplot(
    aes(x = year,
        y = lifeExp)
  ) +
  geom_boxplot()
```



```
gapminder %>%
  filter(
    continent == "Americas"
  ) %>%
  mutate(
    year = factor(year)
  ) %>%
  ggplot(
    aes(x = year,
        y = lifeExp)
  ) +
  geom_boxplot() +
  coord_flip()
```



Recap and where to go next

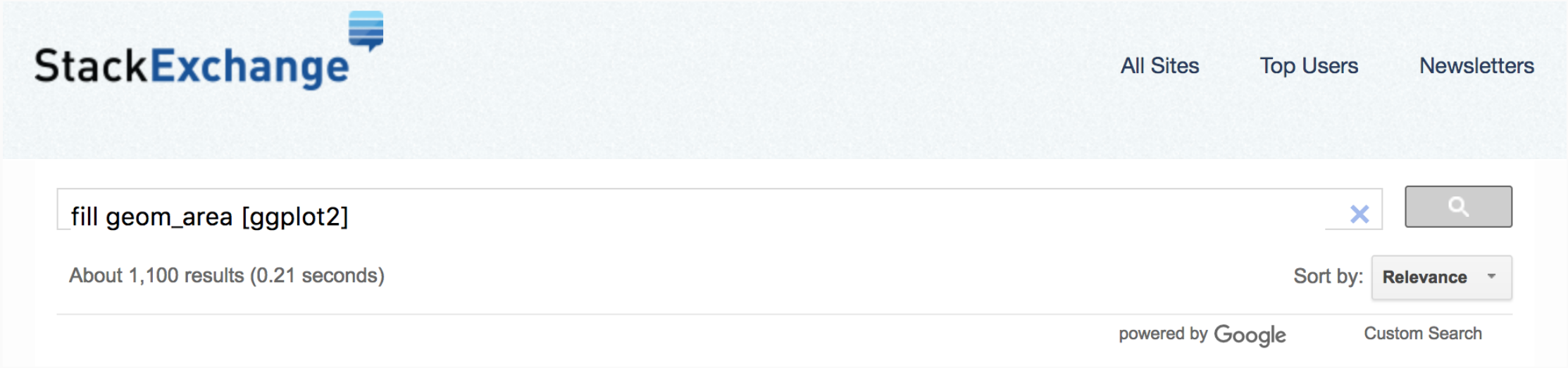
Plots are built in layers

- **Data** must be in a "tidy" format.
- **Aesthetic mappings** link variables in the data to graphical properties in the **geom**etric objects.
- **Geometric objects** dictate how the **aesthetics** are interpreted as a graphical representation (points, lines, polygons, etc.)
- **Statistics** transform the input variables to displayed values. E.g. calculate the summary statistics for a boxplot (quantiles).
- **Coordinates** organize location of geometric objects, i.e. define the physical mapping of the aesthetics.

Plots are built in layers

- **Scales** define the range of values for aesthetics (e.g. categories -> colours).
- **Facets** define the number of panels and how to split data among them (e.g. by country).
- **Themes** control every part of the graphic that is not linked to the data (i.e. font, visual appearance).

Stack Exchange is Great!



Stack Exchange is Great!

1 Answer

active oldest votes

7

You need to make a new grouping variable for each positive/negative segment. To make the transitions less "blocky", you can just first interpolate the data:

```
require(ggplot2)

# Load data
df = read.table('data.txt', header=T)
df$created = as.POSIXct(df$created, tz='UTC')

# Interpolate data
lin_interp = function(x, y, length.out=100) {
    approx(x, y, xout=seq(min(x), max(x), length.out=length.out))$y
}
created.interp = lin_interp(df$created, df$created)
created.interp = as.POSIXct(created.interp, origin='1970-01-01', tz='UTC')
score.interp   = lin_interp(df$created, df$score)
df.interp = data.frame(created=created.interp, score=score.interp)
```

ggplot2 Extensions: ggplot2-exts.org

ggplot2 and beyond

Learn more

- **Draw anything with ggplot2**: https://github.com/thomasp85/ggplot2_workshop
- **Be the boss of your factors and other useful tips:** https://stat545.com/index.html
- **ggplot2 docs:** http://ggplot2.tidyverse.org/
- **R4DS - Data visualization:** http://r4ds.had.co.nz/data-visualisation.html
- **Hadley Wickham's ggplot2 book:** https://www.amazon.com/dp/0387981403/

ggplot2 and beyond

Noteworthy RStudio Add-Ins

- esquisse: Interactively build ggplot2 plots
- ggplotThemeAssist: Customize your ggplot theme interactively
- ggedit: Layer, scale, and theme editing

Practice and Review

#TidyTuesday

- https://github.com/rfordatascience/tidytuesday

Fun Datasets

- `fivethirtyeight`
- `nycflights`
- `ggplot2movies`

Thanks!

- Slides and code adapted from Garrick Aden-Buie GitHub: http://github.com/gadenbuie/gentle-ggplot2