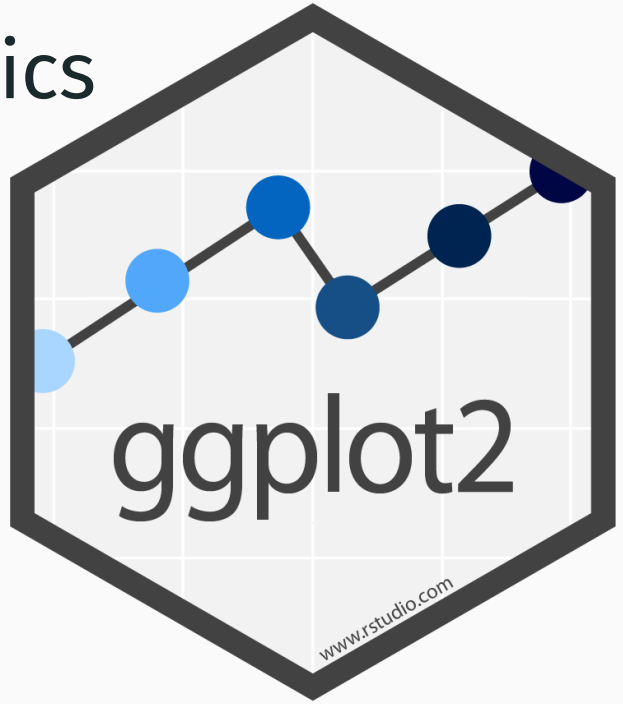
A Gentle Guide to the Grammar of Graphics with `ggplot2`

Garrick Aden-Buie
@grrrck
Slides and code: **bit.ly/gentle-ggplot2**



Brought to you by the letter...

Why *ggplot2*?

Hadley Wickham

The transferrable skills from ggplot2 are not the idiosyncracies of plotting syntax, but 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 *ggplot2*?

My personal reasons

- Functional data visualization

    1. Wrange data
    2. Map data to visual elements
    3. Tweak scales, guides, axis, labels, theme

- Easy to reason about how data drives visualization
- Easy to iterate
- Easy to be consistent

What are we getting into?

`ggplot2` is a huge package: philosophy + functions

...but it's very well organized

What are we getting into?

`ggplot2` is a huge package: philosophy + functions

...but it's very well organized

Lots of examples of not-so-great plots in these slides

...but that's okay

What are we getting into?

`ggplot2` is a huge package: philosophy + functions
...but it's very well organized

Lots of examples of not-so-great plots in these slides
...but that's okay

Going to throw a lot at you
...but you'll know *where* and *what* to look for

What are we getting into?

`ggplot2` is a huge package: philosophy + functions
...but it's very well organized

Lots of examples of not-so-great plots in these slides
...but that's okay

Going to throw a lot at you
...but you'll know *where* and *what* to look for

What are we getting into?

`ggplot2` is a huge package: philosophy + functions
...but it's very well organized

Lots of examples of not-so-great plots in these slides
...but that's okay

Going to throw a lot at you
...but you'll know *where* and *what* to look for

G is for getting started

Easy: install the tidyverse

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

Medium: install just `ggplot2`

```
install.pacakages('ggplot2')
```

Expert: install from GitHub

```
devtools::install_github('tidyverse/ggplot2')
```

G is for getting started

Load the tidyverse

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──

## ✔ ggplot2 3.1.0     ✔ purrr   0.2.5
## ✔ tibble  1.4.2     ✔ dplyr   0.7.7
## ✔ tidyr   0.8.1     ✔ stringr 1.3.1
## ✔ readr   1.1.1     ✔ forcats 0.3.0

## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

G is for getting started

Other packages you'll need for this adventure

We'll use an excerpt of the gapminder dataset provided by the `gapminder` package by Jenny Bryan.

https://github.com/jennybc/gapminder

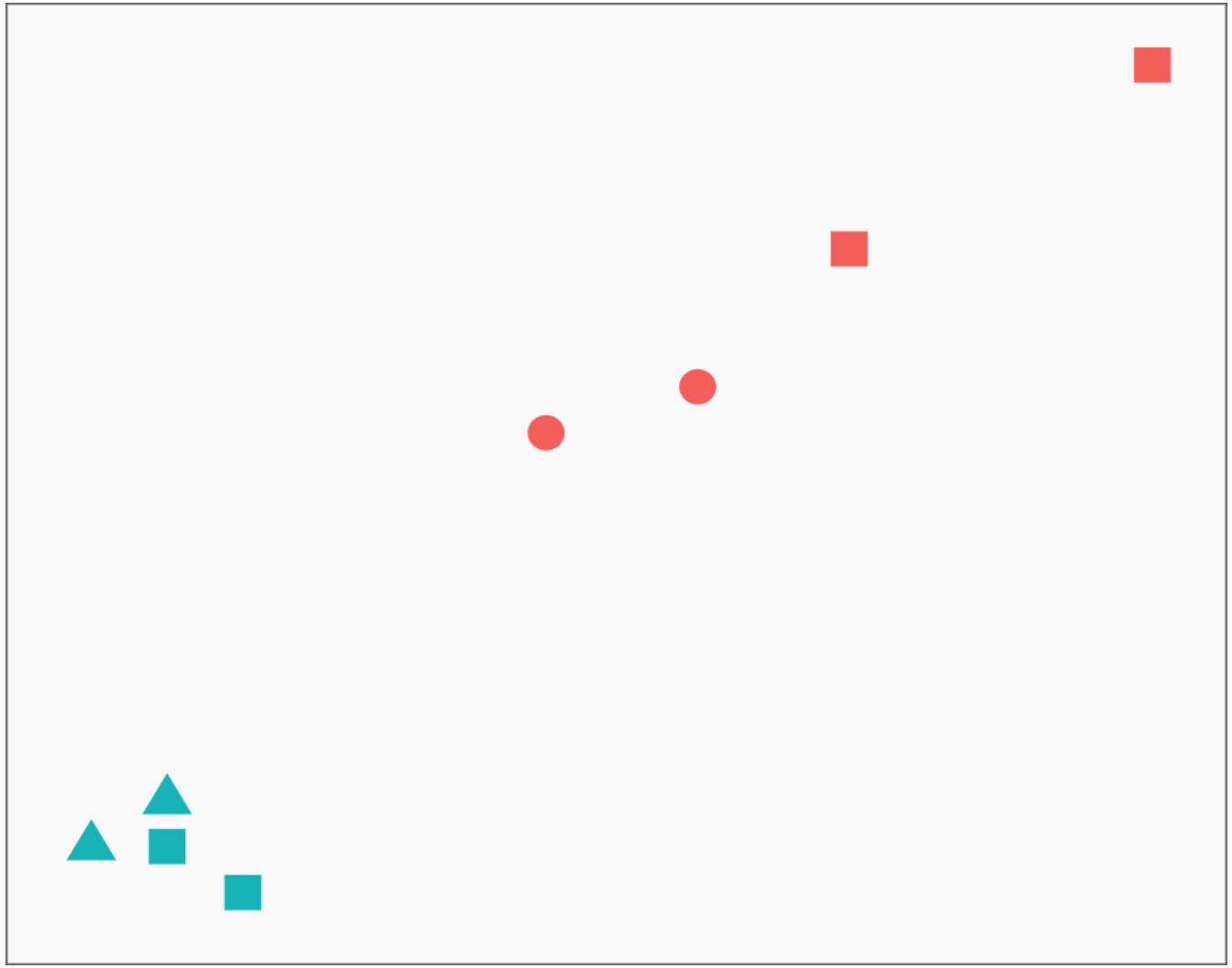
```
## install.packages("gapminder")
library(gapminder)
```

gg is for
Grammar of Graphics

What is a grammar of graphics?

"Good grammar is just the first step in creating a good sentence."

How is the drawing on the right connected to data?

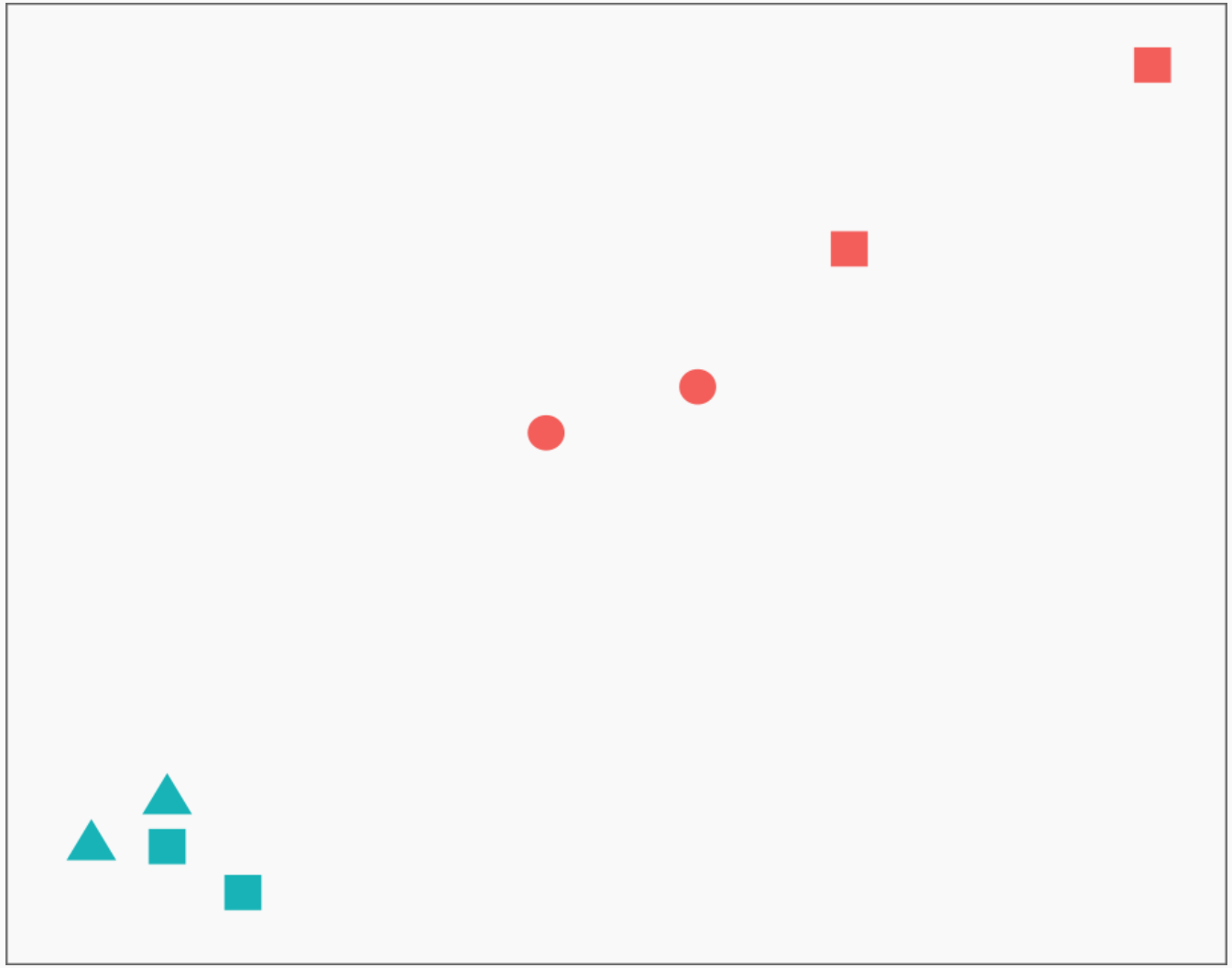


http://vita.had.co.nz/papers/layered-grammar.pdf

Guess the data behind this plot?

MPG Ratings of Cars

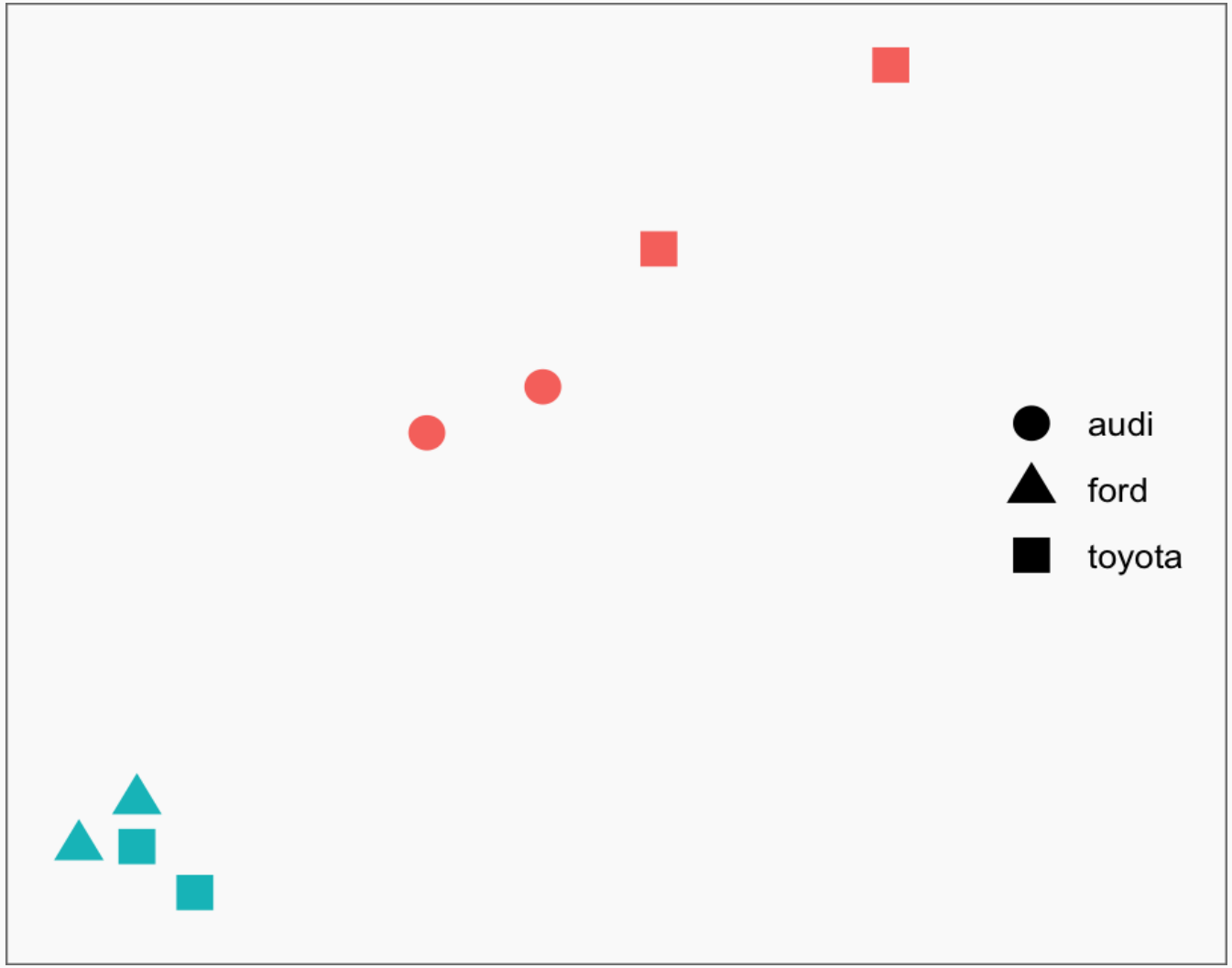
- Manufacturer
- Car Type (Class)
- City MPG
- Highway MPG



Guess the data behind this plot?

MPG Ratings of Cars

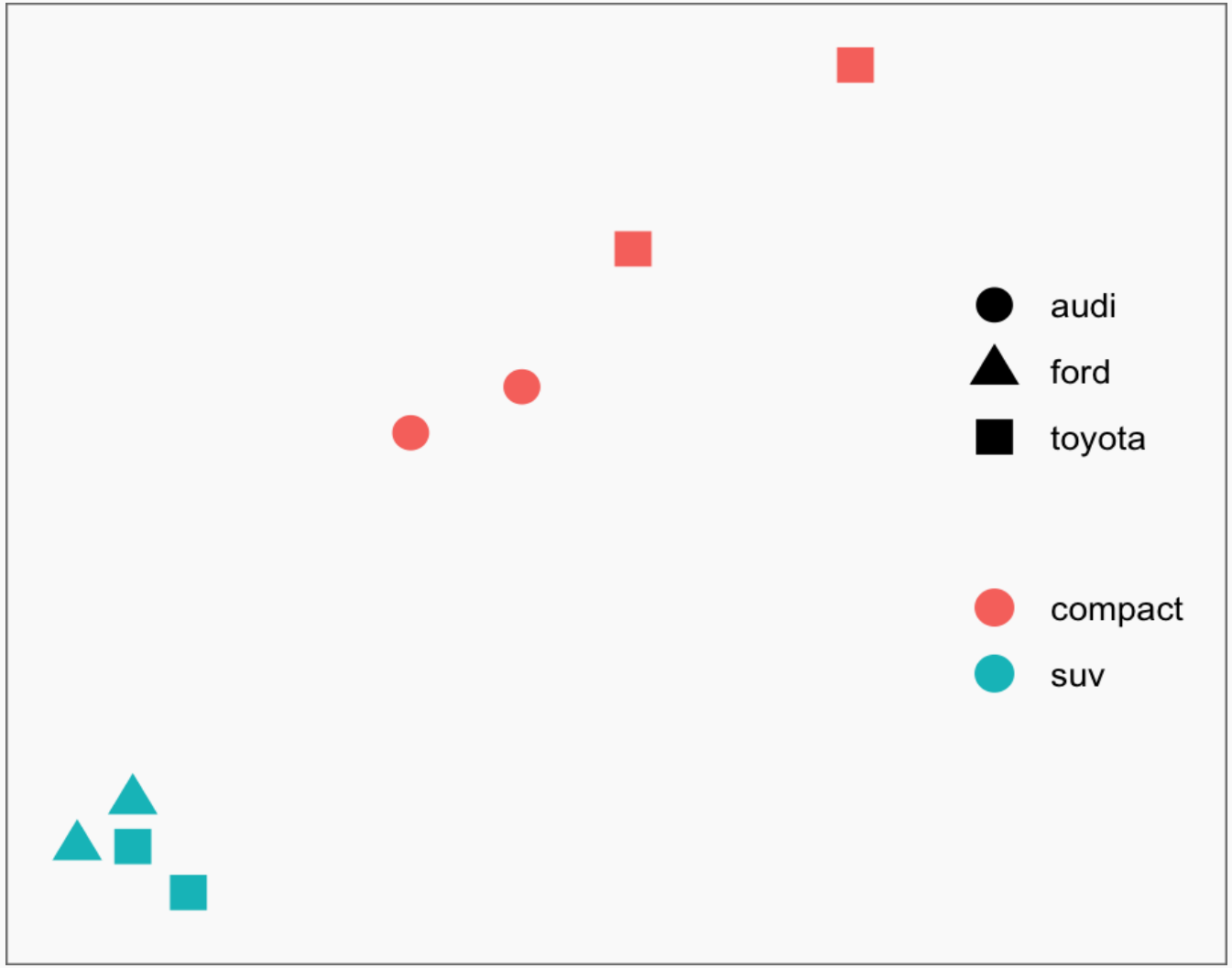
- Manufacturer
- Car Type (Class)
- City MPG
- Highway MPG



Guess the data behind this plot?

MPG Ratings of Cars

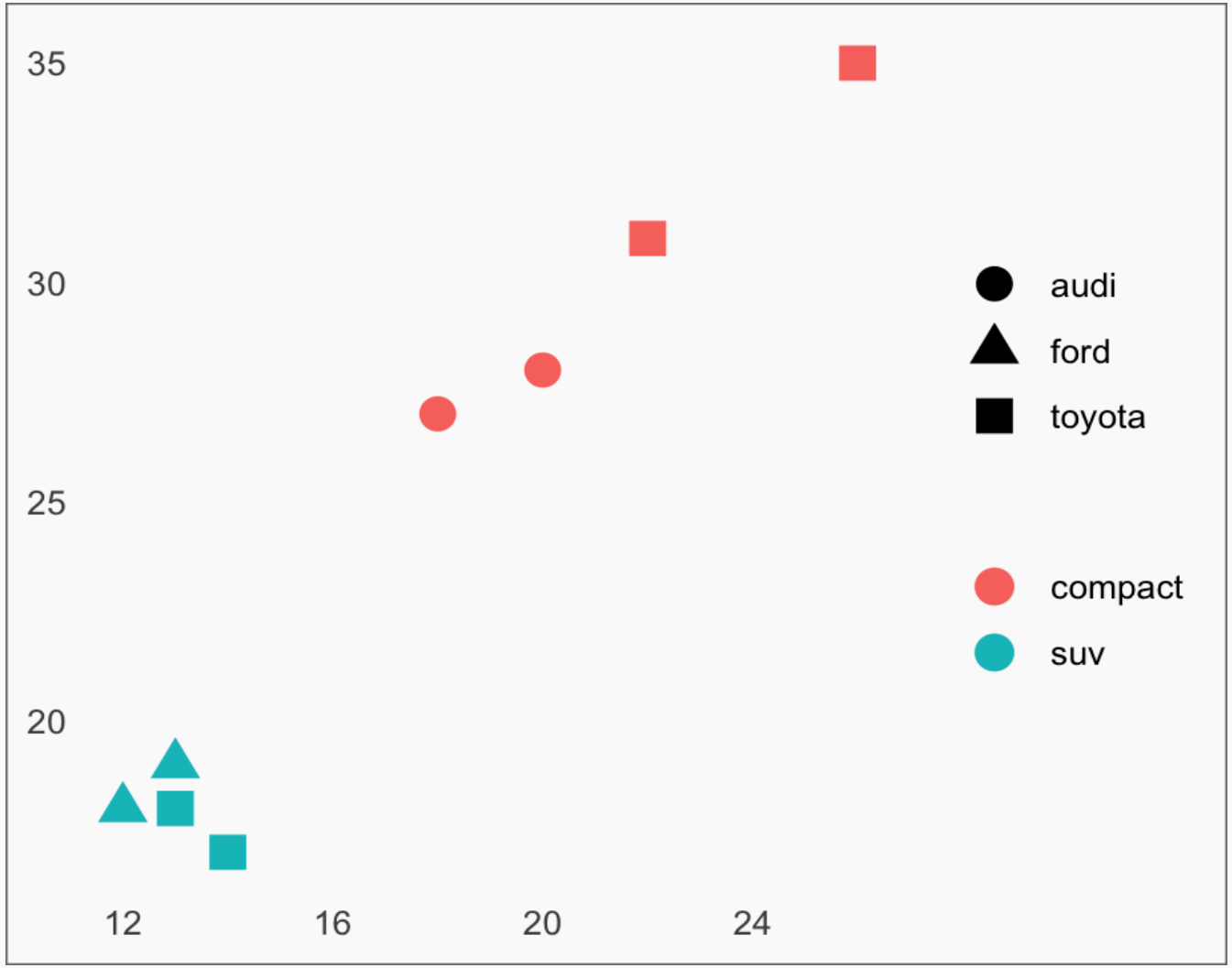
- Manufacturer
- Car Type (Class)
- City MPG
- Highway MPG



Guess the data behind this plot?

MPG Ratings of Cars

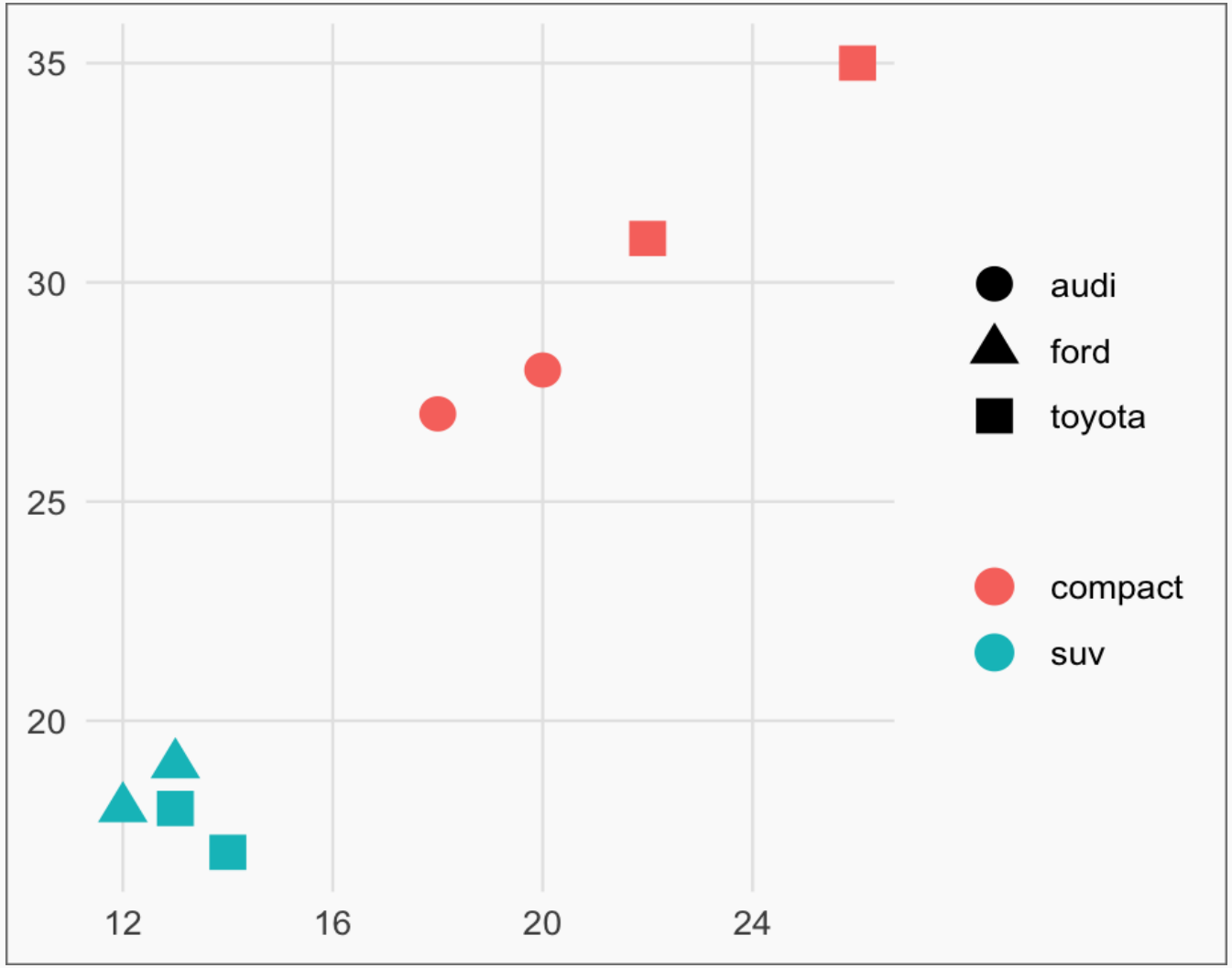
- Manufacturer
- Car Type (Class)
- City MPG
- Highway MPG



Guess the data behind this plot?

MPG Ratings of Cars

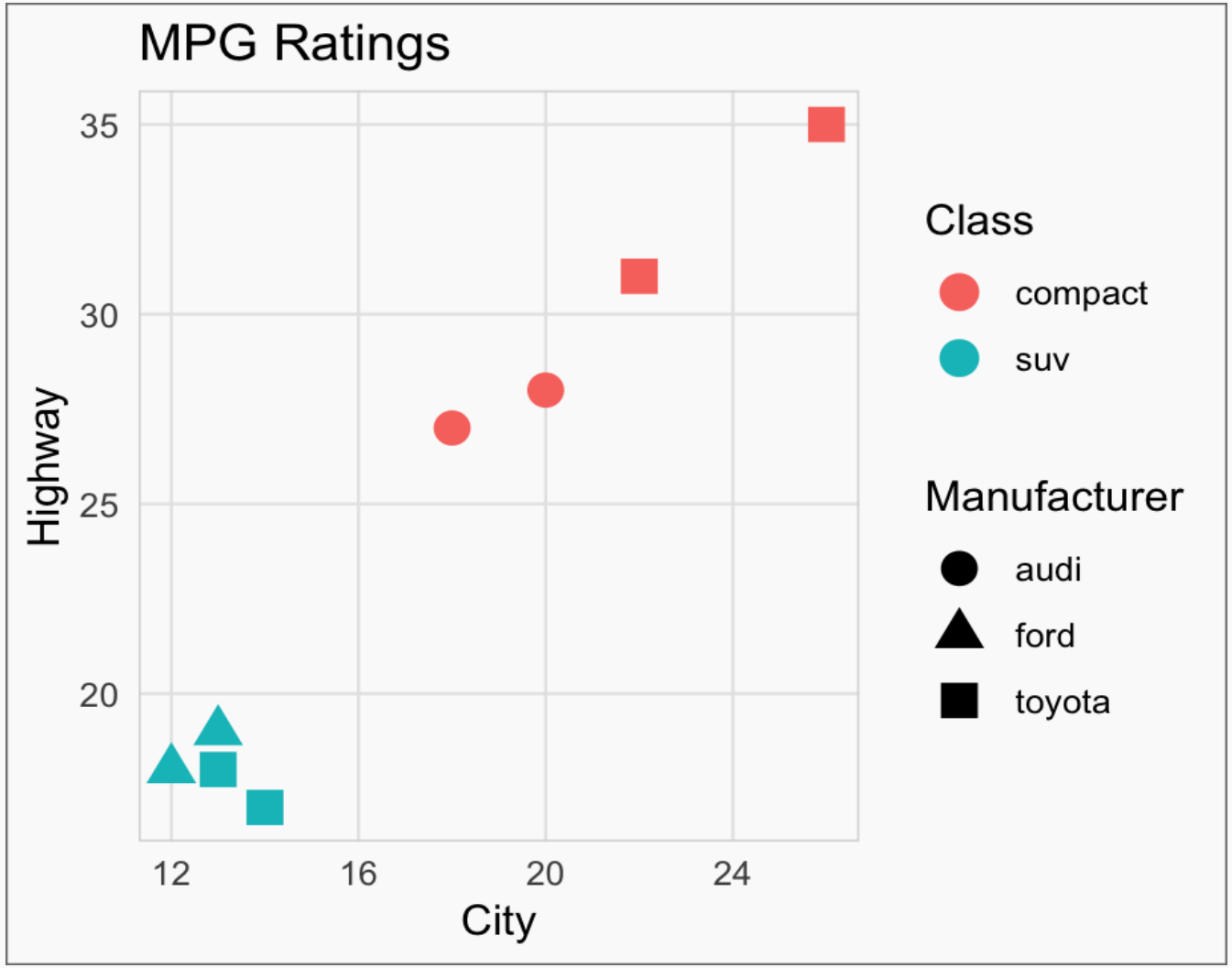
- Manufacturer
- Car Type (Class)
- City MPG
- Highway MPG



Guess the data behind this plot?

MPG Ratings of Cars

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



Guess the data behind this plot?

MPG Ratings of Cars

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

manufacturer	class	cty	hwy	model
audi	compact	18	27	a4
audi	compact	20	28	a4 quattro
ford	suv	12	18	expedition 2wd
ford	suv	13	19	explorer 4wd
toyota	suv	14	17	4runner 4wd
toyota	compact	22	31	camry solara
toyota	compact	26	35	corolla
toyota	suv	13	18	land cruiser wagon 4wd

How do we express visuals in words?

- **Data** to be visualized

How do we express visuals in words?

- **Data** to be visualized
- **Geometric objects** that appear on the plot

How do we express visuals in words?

- **Data** to be visualized
- **Geometric objects** that appear on the plot
- **Aesthetic mappings** from data to visual component

How do we express visuals in words?

- **Data** to be visualized
- **Geometric objects** that appear on the plot
- **Aesthetic mappings** from data to visual component
- **Statistics** transform data on the way to visualization

How do we express visuals in words?

- **Data** to be visualized
- **Geometric objects** that appear on the plot
- **Aesthetic mappings** from data to visual component
- **Statistics** transform data on the way to visualization
- **Coordinates** organize location of geometric objects

How do we express visuals in words?

- **Data** to be visualized
- **Geometric objects** that appear on the plot
- **Aesthetic mappings** from data to visual component
- **Statistics** transform data on the way to visualization
- **Coordinates** organize location of geometric objects
- **Scales** define the range of values for aesthetics

How do we express visuals in words?

- **Data** to be visualized
- **Geometric objects** that appear on the plot
- **Aesthetic mappings** from data to visual component
- **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

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 observational unit forms a table

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 observational unit forms a table

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
Canada	30.30584	31.90227	33.39014
China	1230.07500	1280.40000	1318.68310
United States	272.91176	287.67553	301.13995

gg is for Grammar of Graphics

Data

```
ggplot(data)
```

country	1997	2002	2007
Canada	30.30584	31.90227	33.39014
China	1230.07500	1280.40000	1318.68310
United States	272.91176	287.67553	301.13995

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

country	year	pop
Canada	1997	30.306
China	1997	1230.075
United States	1997	272.912
Canada	2002	31.902

gg is for Grammar of Graphics

Data

Aesthetics

```
+ aes()
```

Map data to visual elements or parameters

- year
- pop
- country

gg is for Grammar of Graphics

Data

Aesthetics

```
+ aes()
```

Map data to visual elements or parameters

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

gg is for Grammar of Graphics

Data

Aesthetics

```
+ aes()
```

Map data to visual elements or parameters

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

gg is for Grammar of Graphics

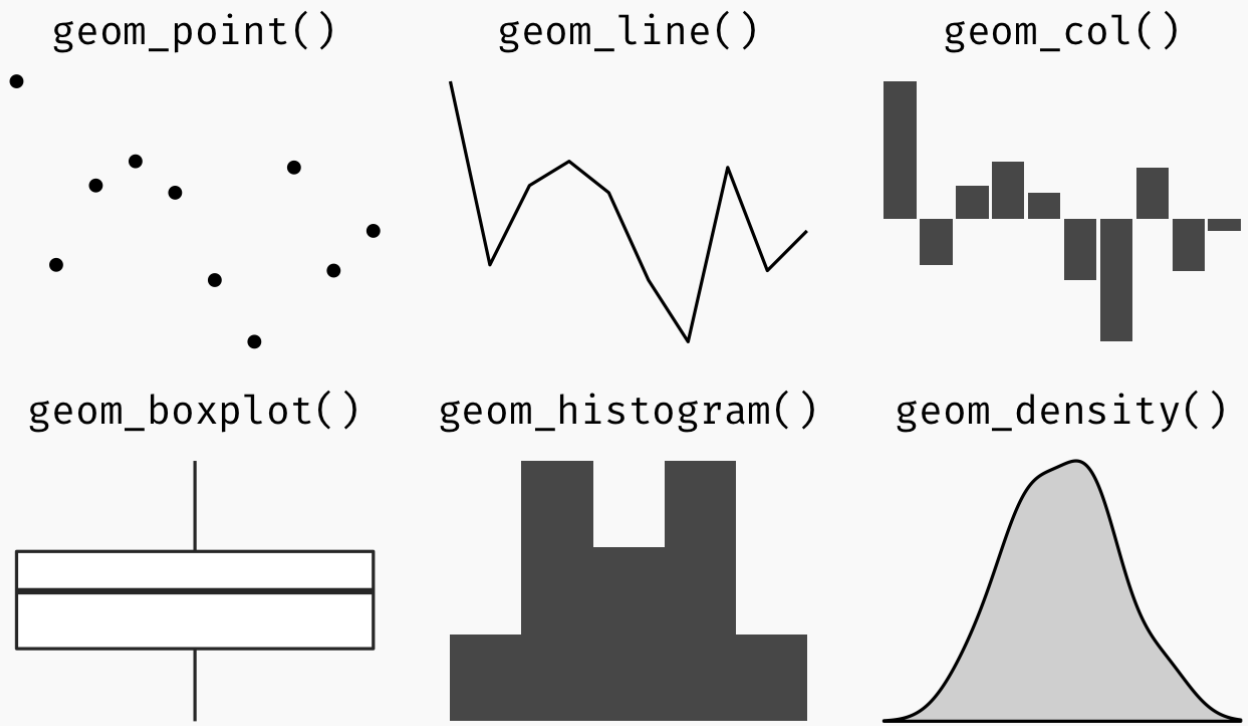
Data

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"
```

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"
```

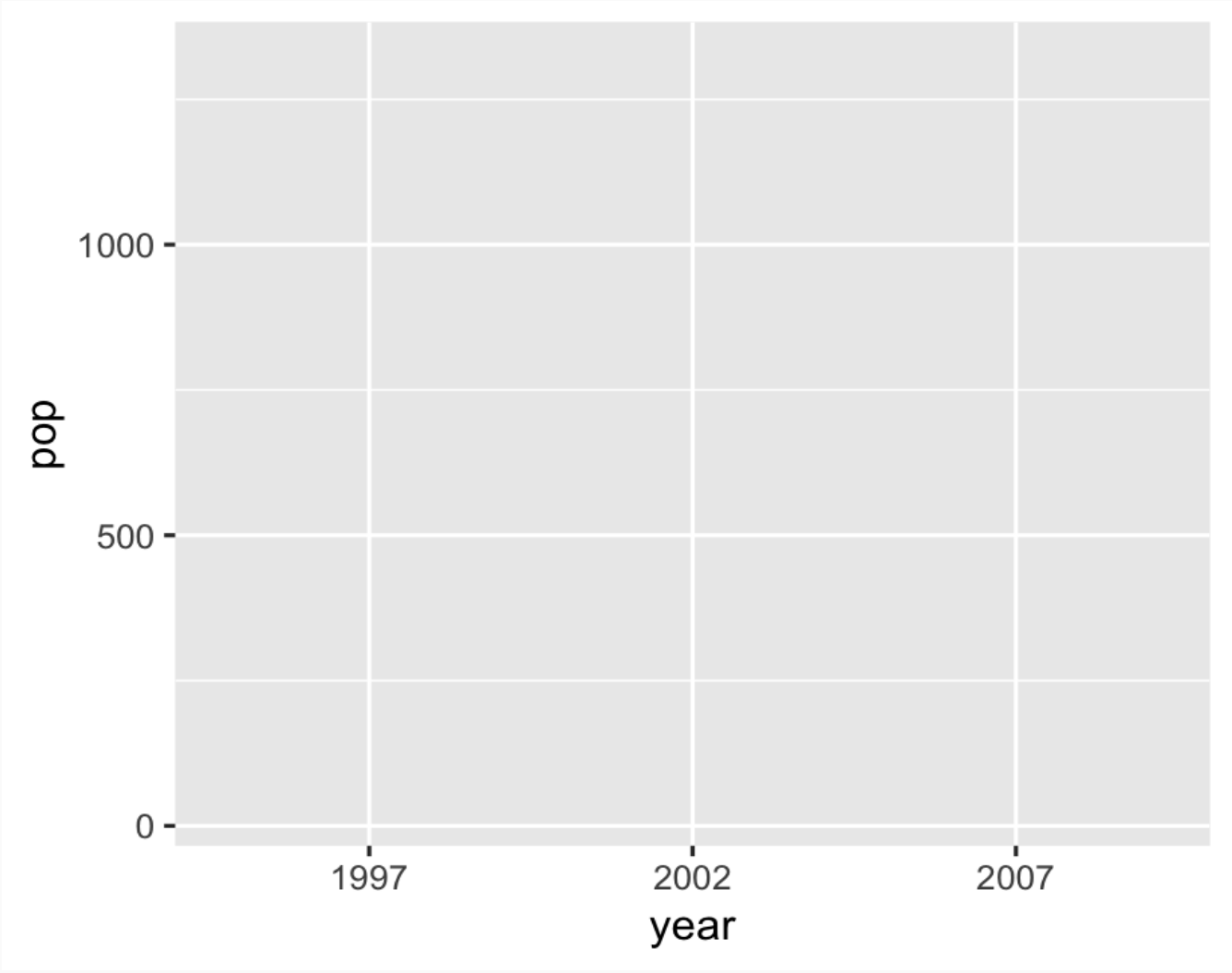
Or just start typing `geom_` in RStudio

Our first plot!

```
ggplot(tidy_pop)
```

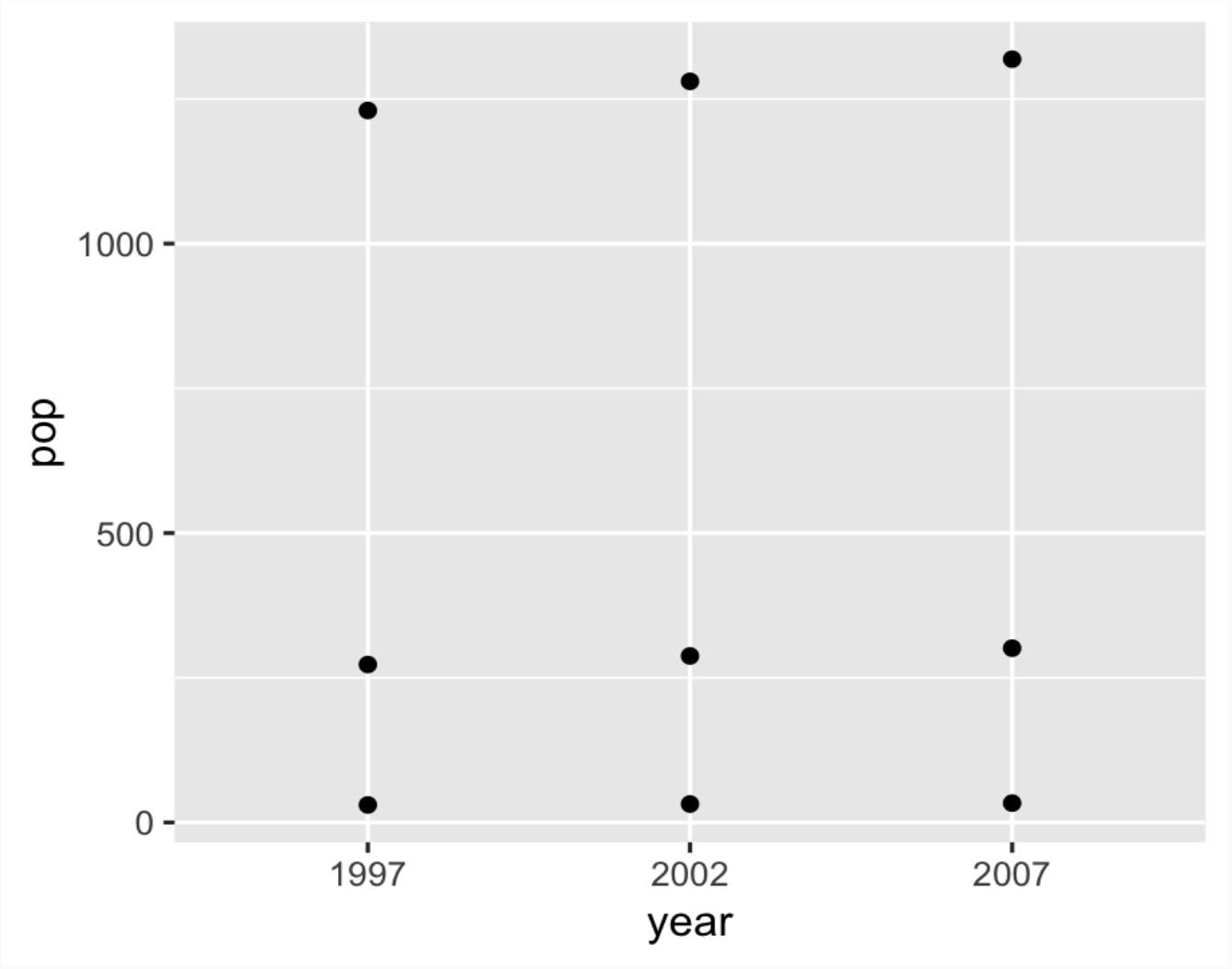
Our first plot!

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



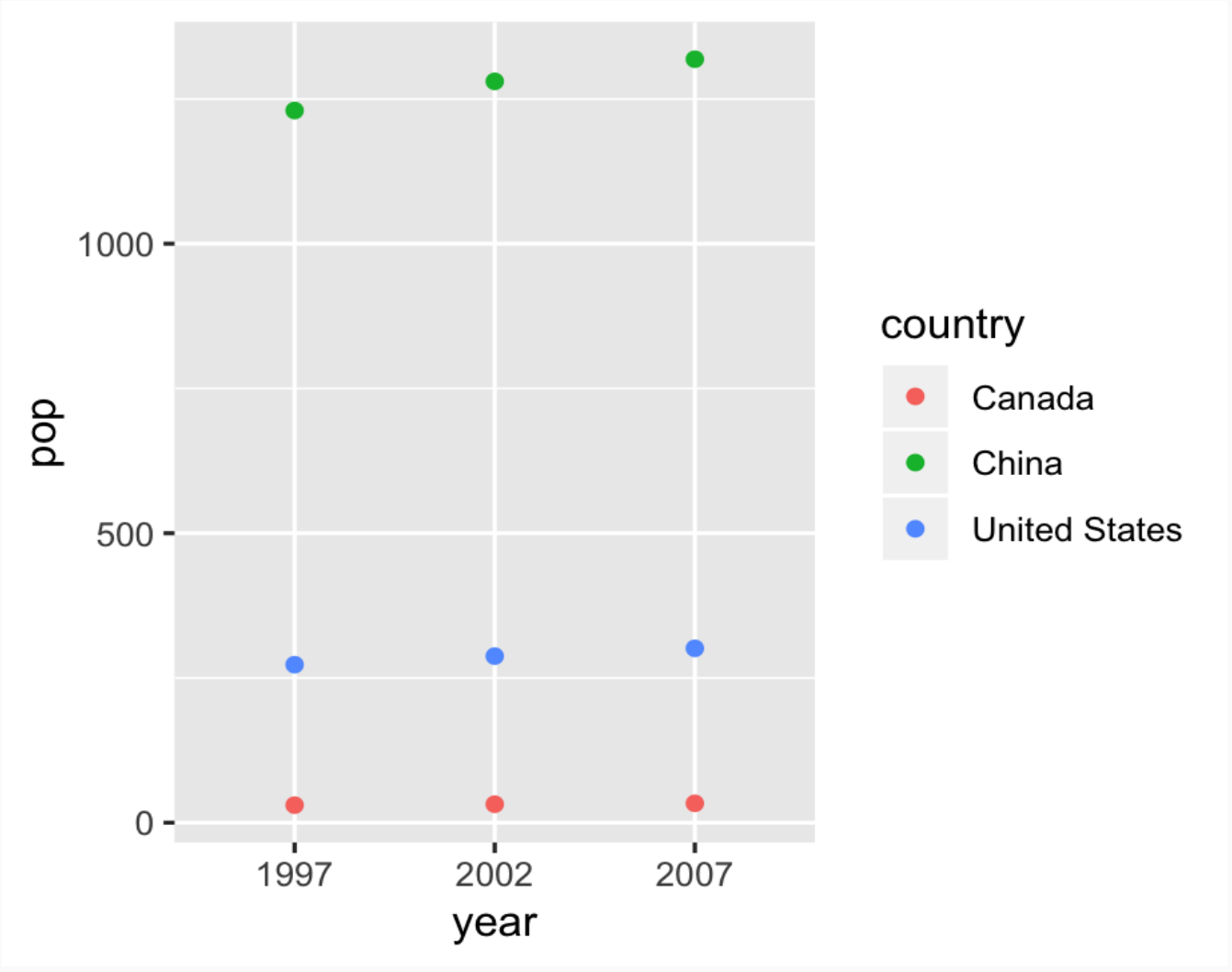
Our first plot!

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



Our first plot!

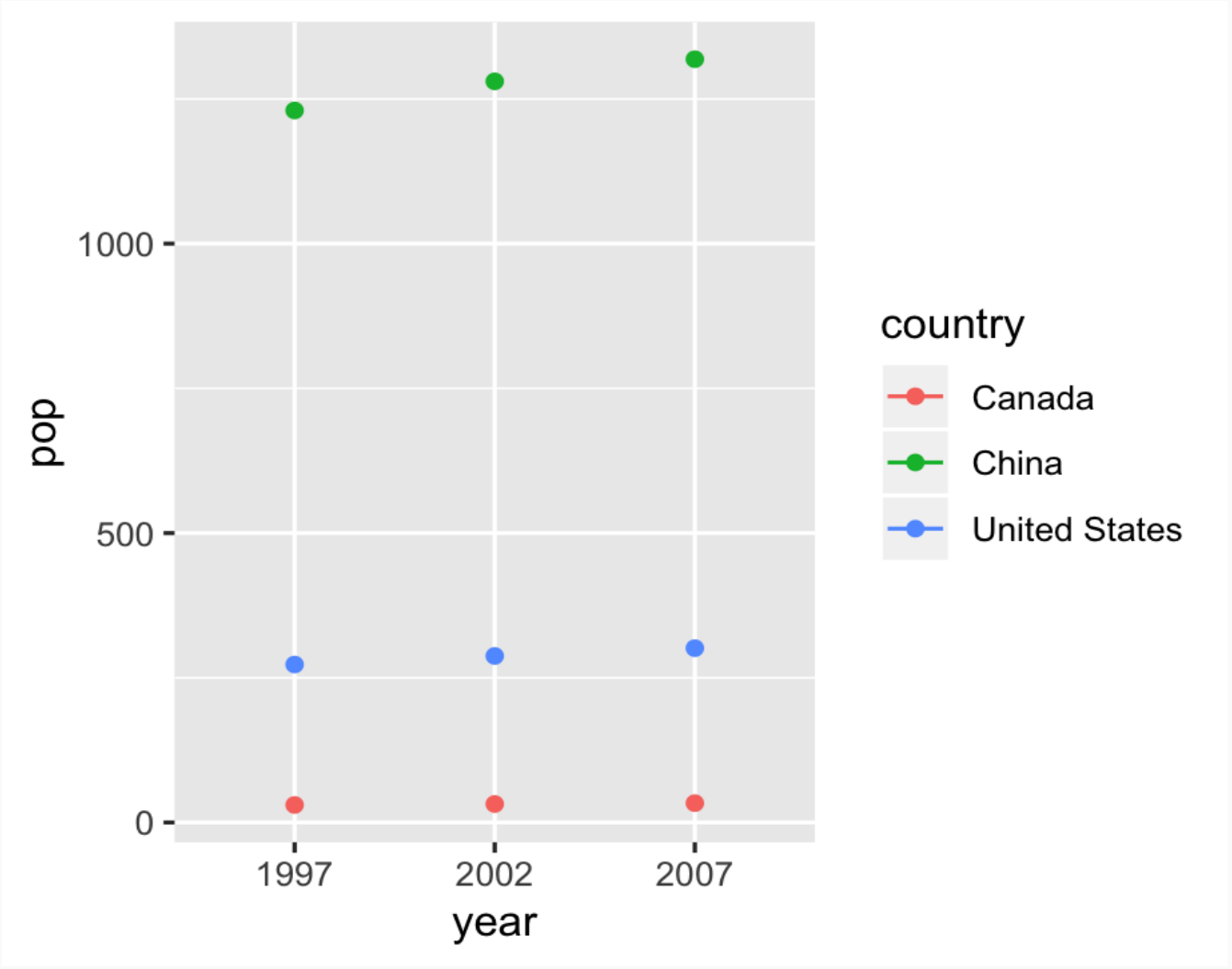
```
ggplot(tidy_pop) +
  aes(x = year,
      y = pop,
      color = country) +
  geom_point()
```



Our first plot!

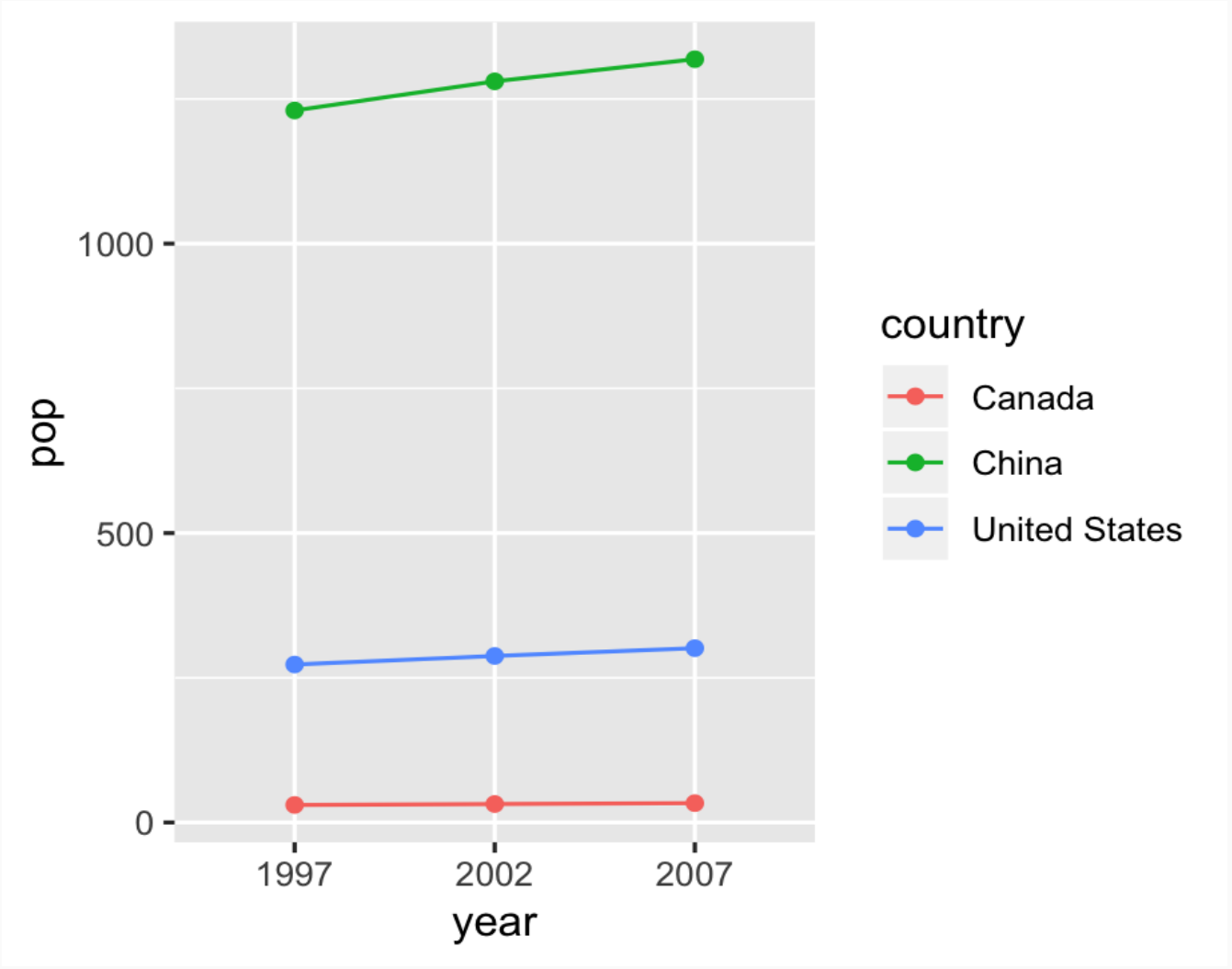
```
ggplot(tidy_pop) +
  aes(x = year,
      y = pop,
      color = country) +
  geom_point() +
  geom_line()
```

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



Our first plot!

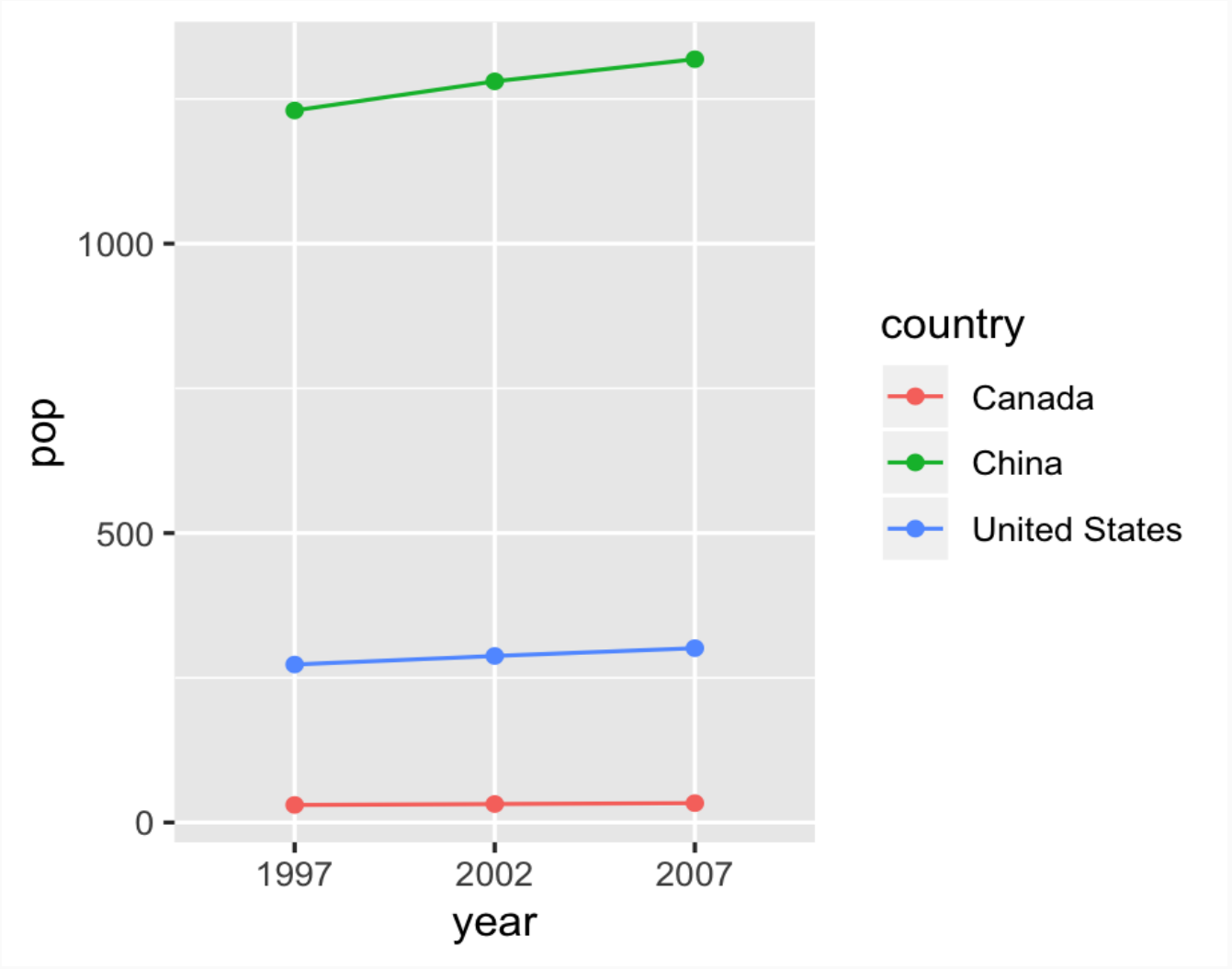
```
ggplot(tidy_pop) +
  aes(x = year,
      y = pop,
      color = country) +
  geom_point() +
  geom_line(
    aes(group = country))
```



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, data, stat, position)
```

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

gg is for Grammar of Graphics

Data

Aesthetics

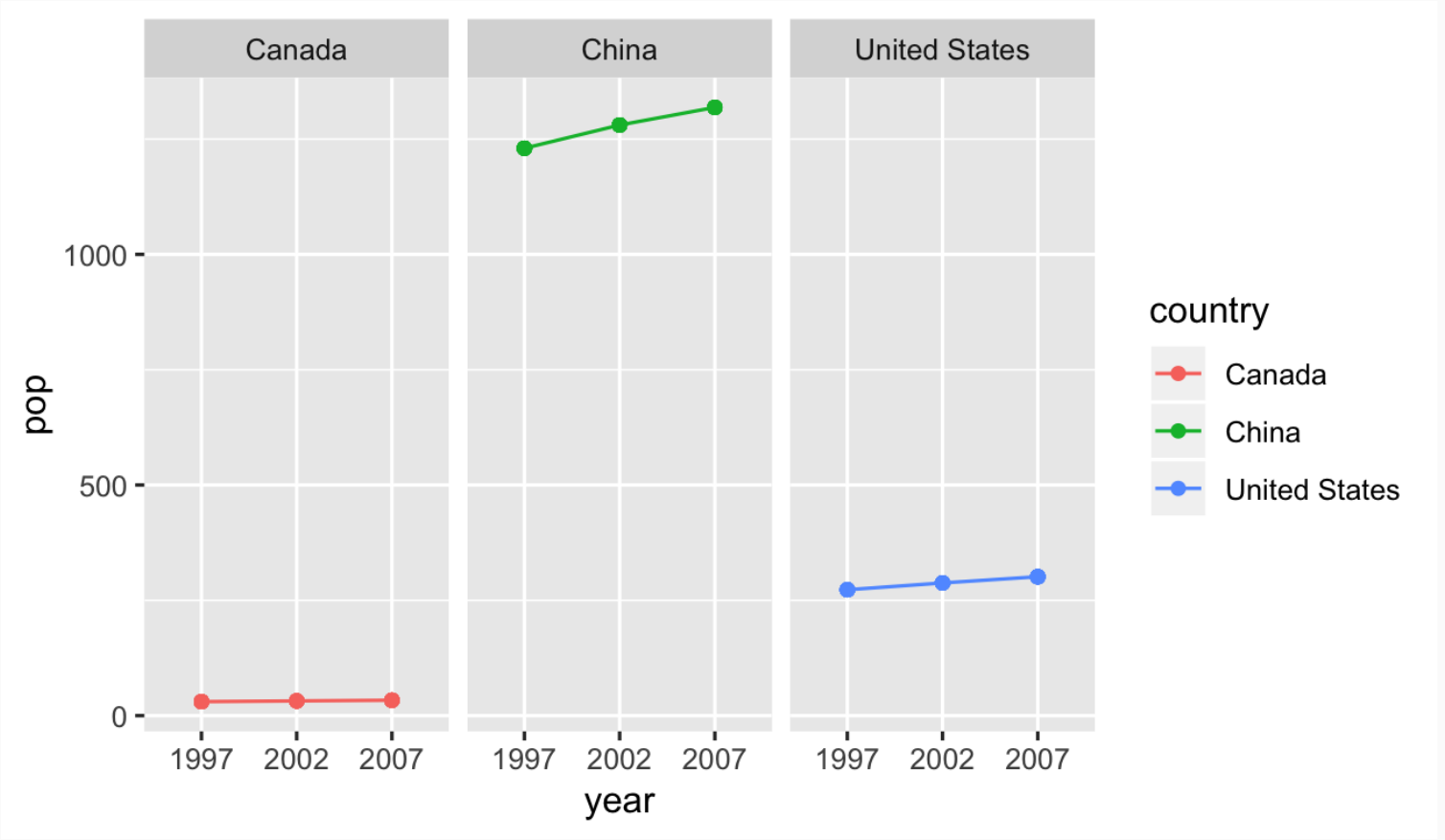
Geoms

Facet

```
+facet_wrap()

+facet_grid()
```

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



gg is for Grammar of Graphics

Data

Aesthetics

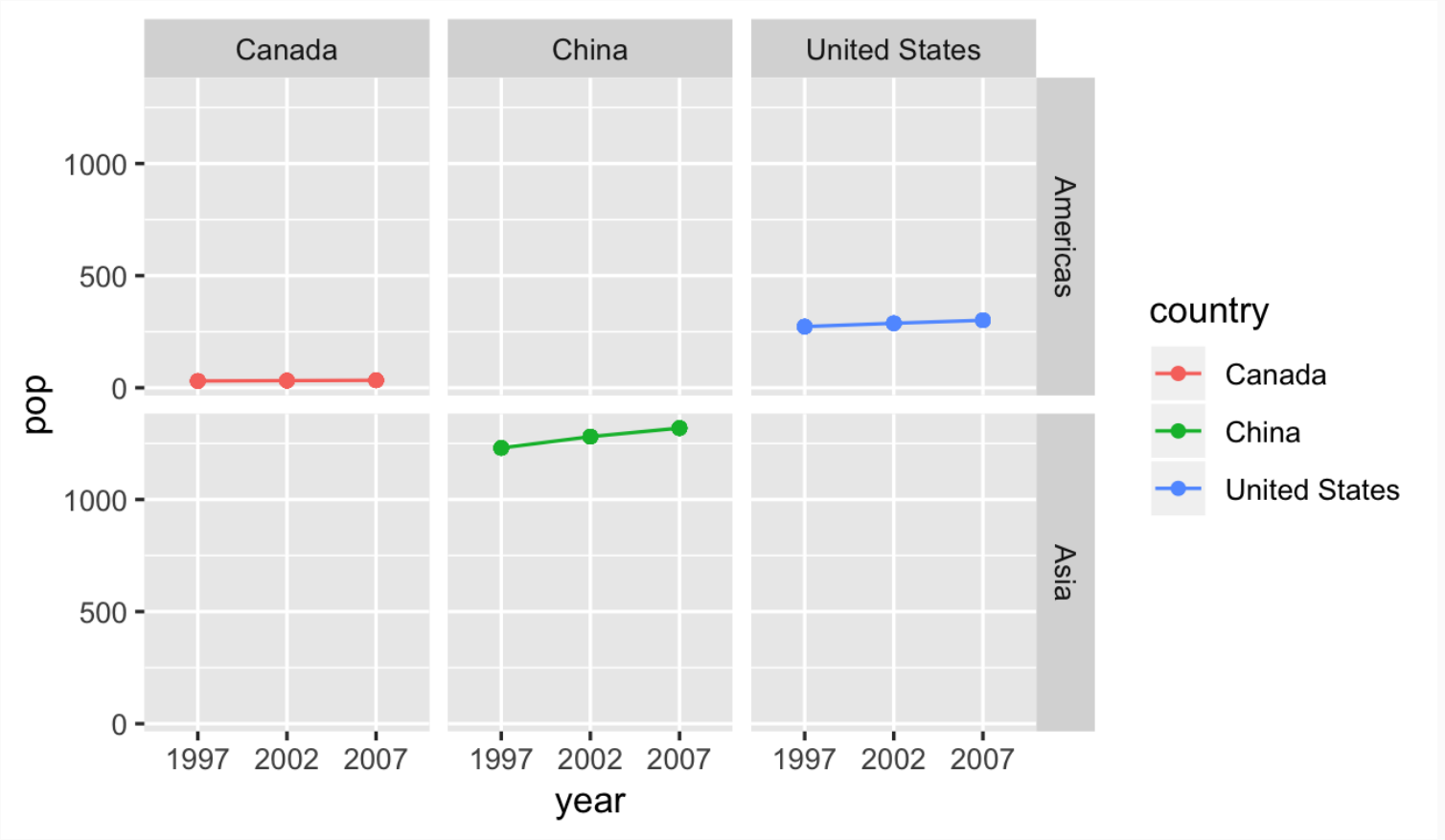
Geoms

Facet

```
+facet_wrap()

+facet_grid()
```

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



gg is for Grammar of Graphics

Data

Aesthetics

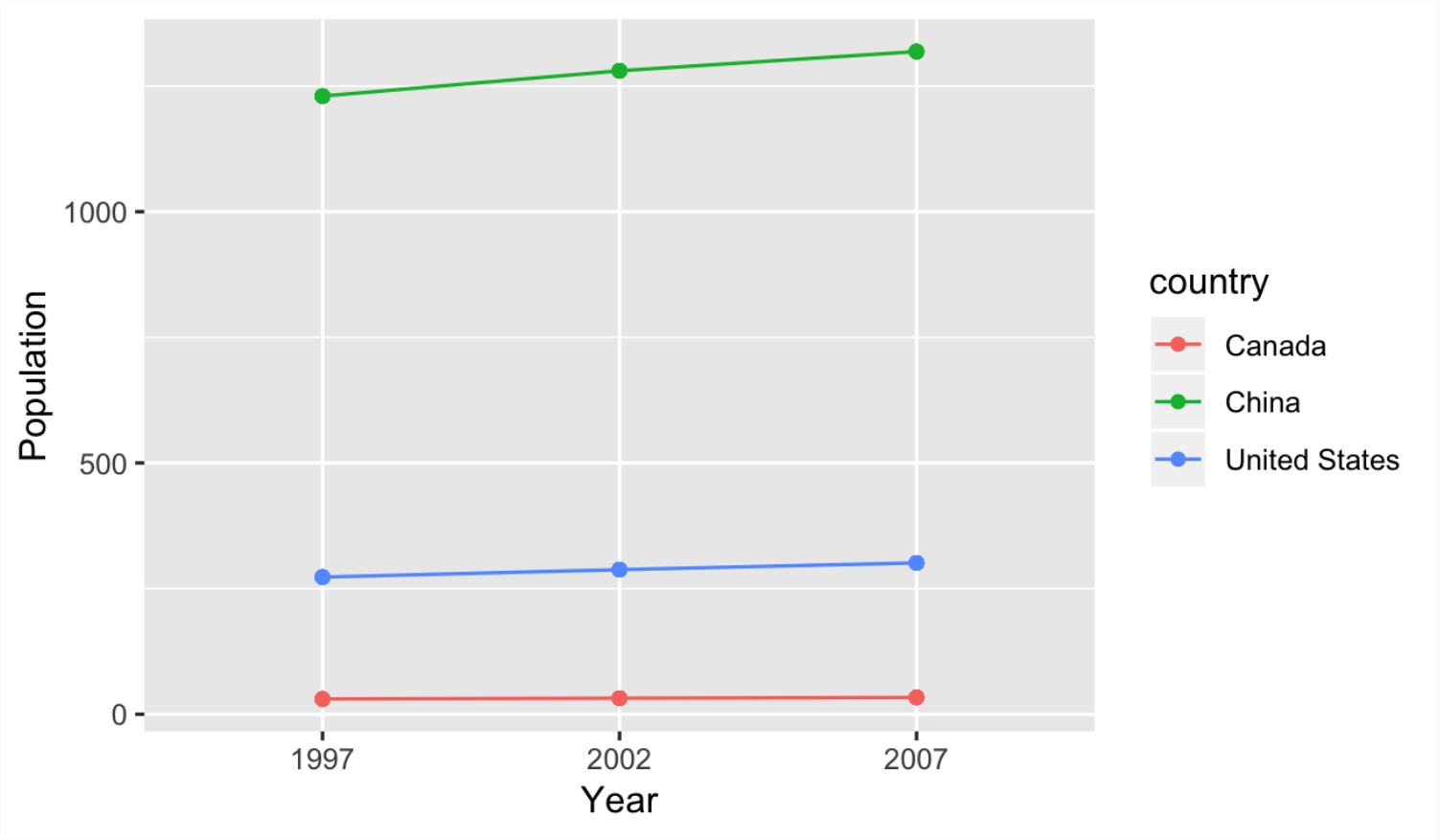
Geoms

Facet

Labels

```
+ labs()
```

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



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

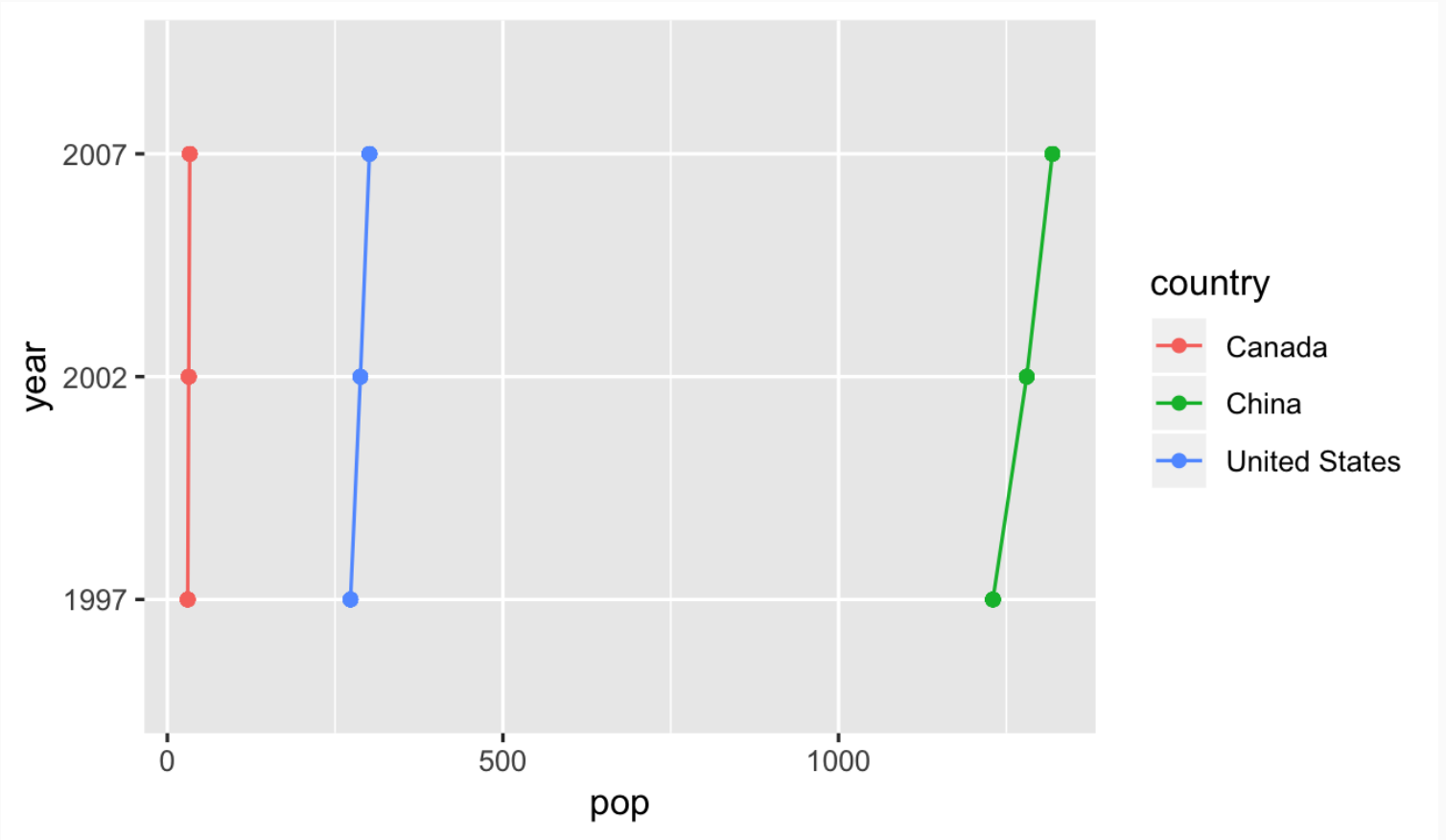
Facet

Labels

Coords

```
+ coord_*()
```

```
g + coord_flip()
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

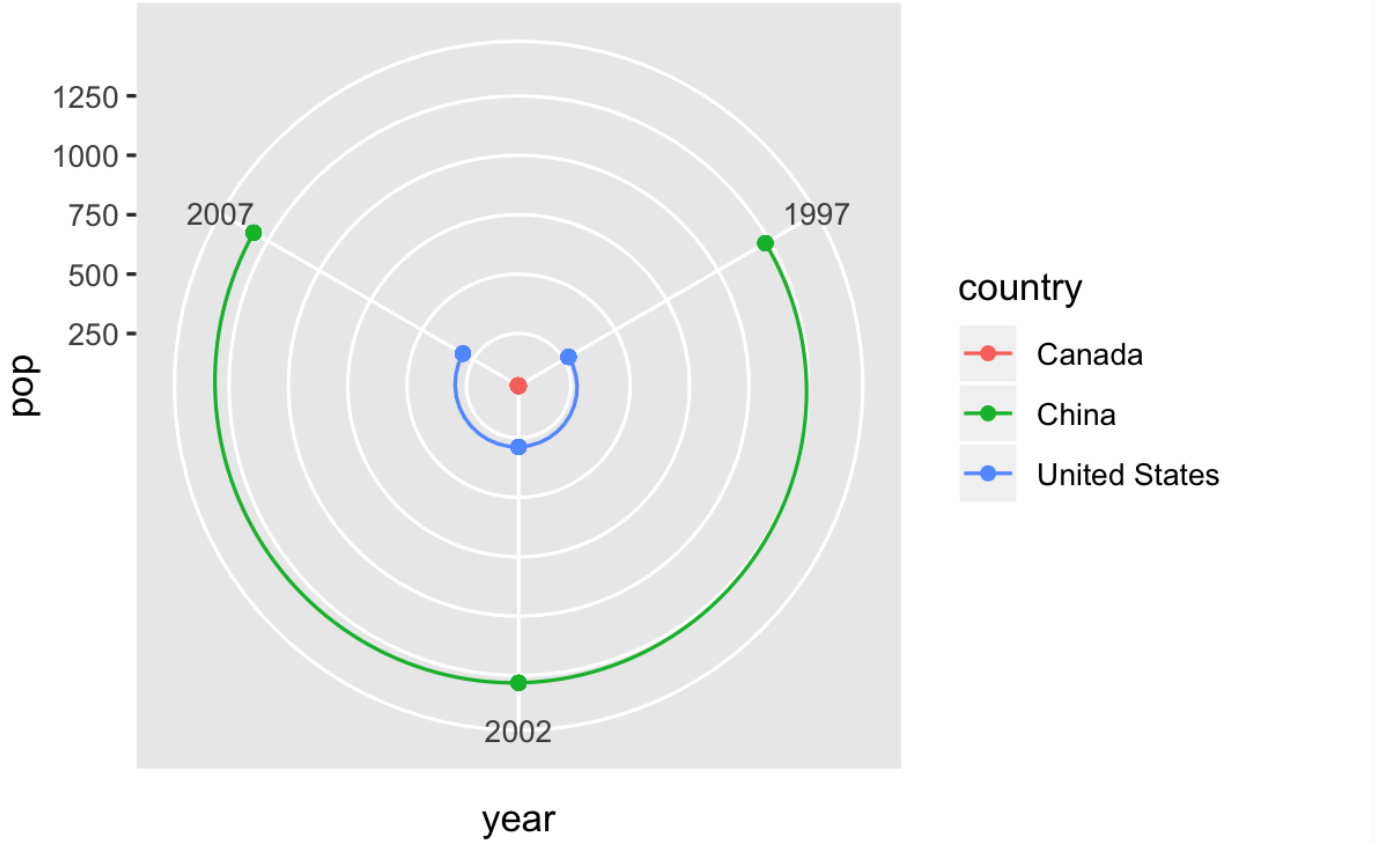
Facet

Labels

Coords

```
+ coord_*()
```

```
g + coord_polar()
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Coords

Scales

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

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

What parameter do you want to adjust? → `<aes>`
What type is the parameter? → `<type>`

- I want to change my discrete x-axis
  `scale_x_discrete()`
- I want to change range of point sizes from continuous variable
  `scale_size_continuous()`
- I want to rescale y-axis as log
  `scale_y_log10()`
- I want to use a different color palette
  `scale_fill_discrete()`
  `scale_color_manual()`

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

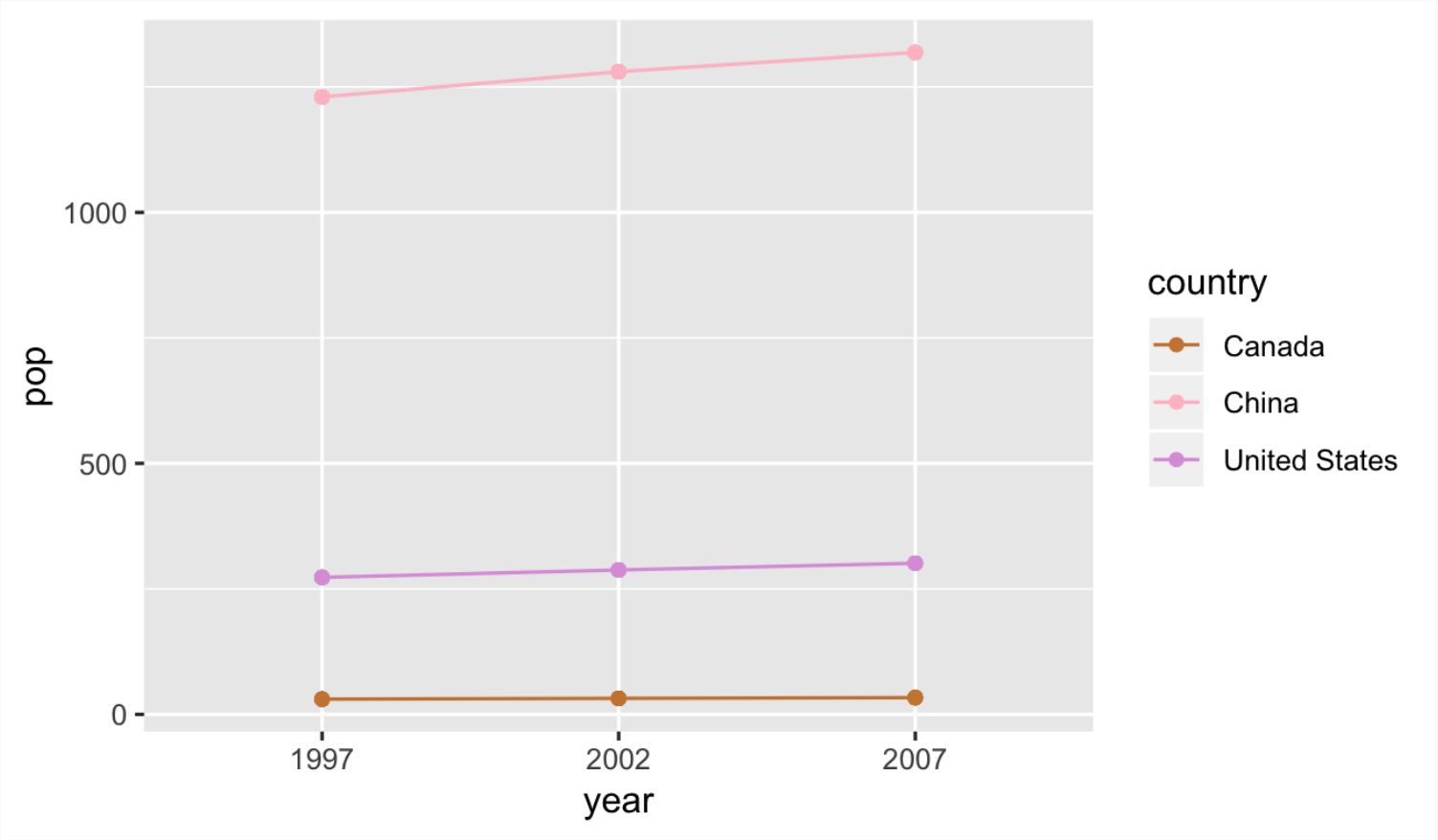
Labels

Coords

Scales

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

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



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

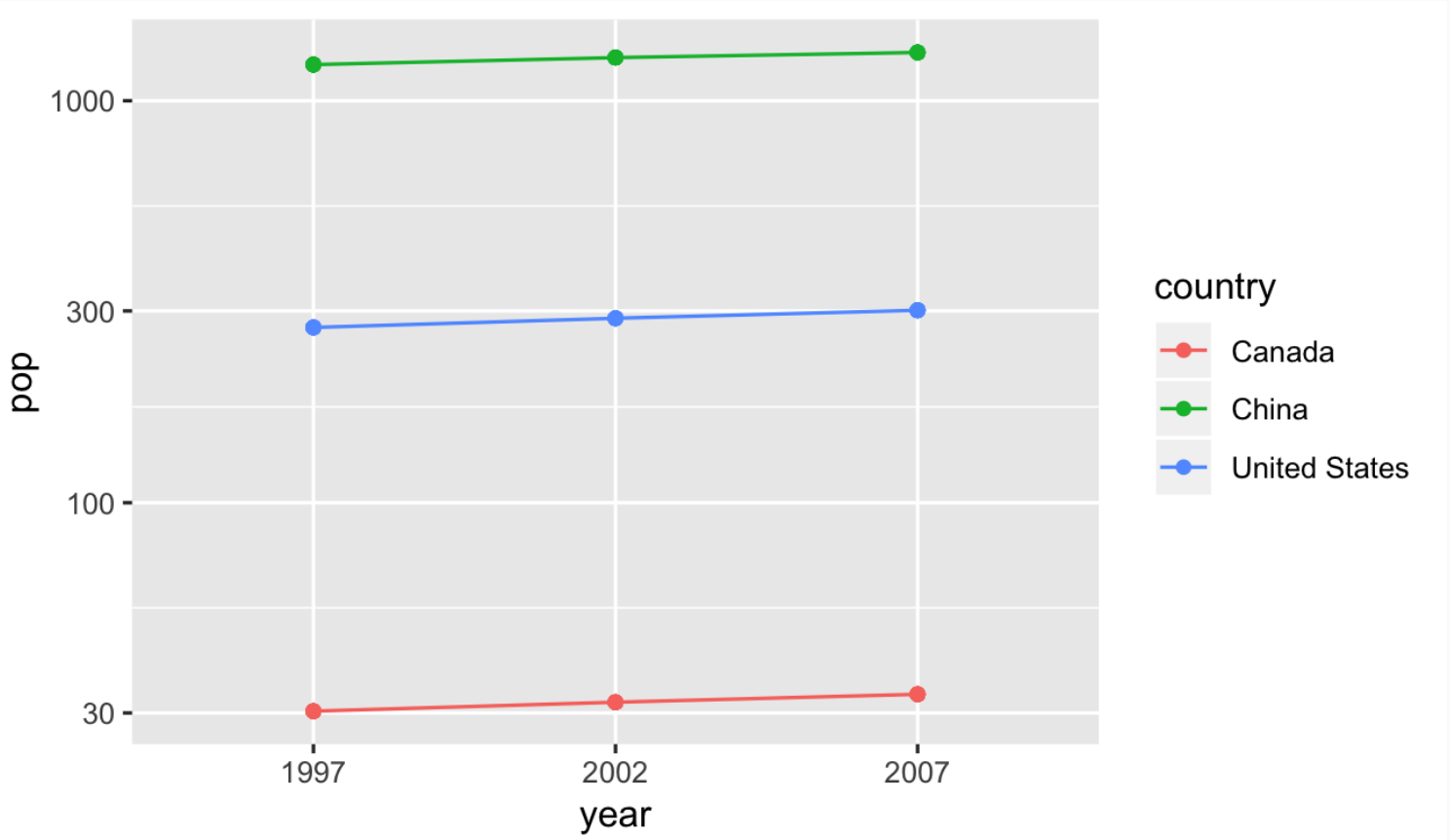
Labels

Coords

Scales

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

```
g + scale_y_log10()
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

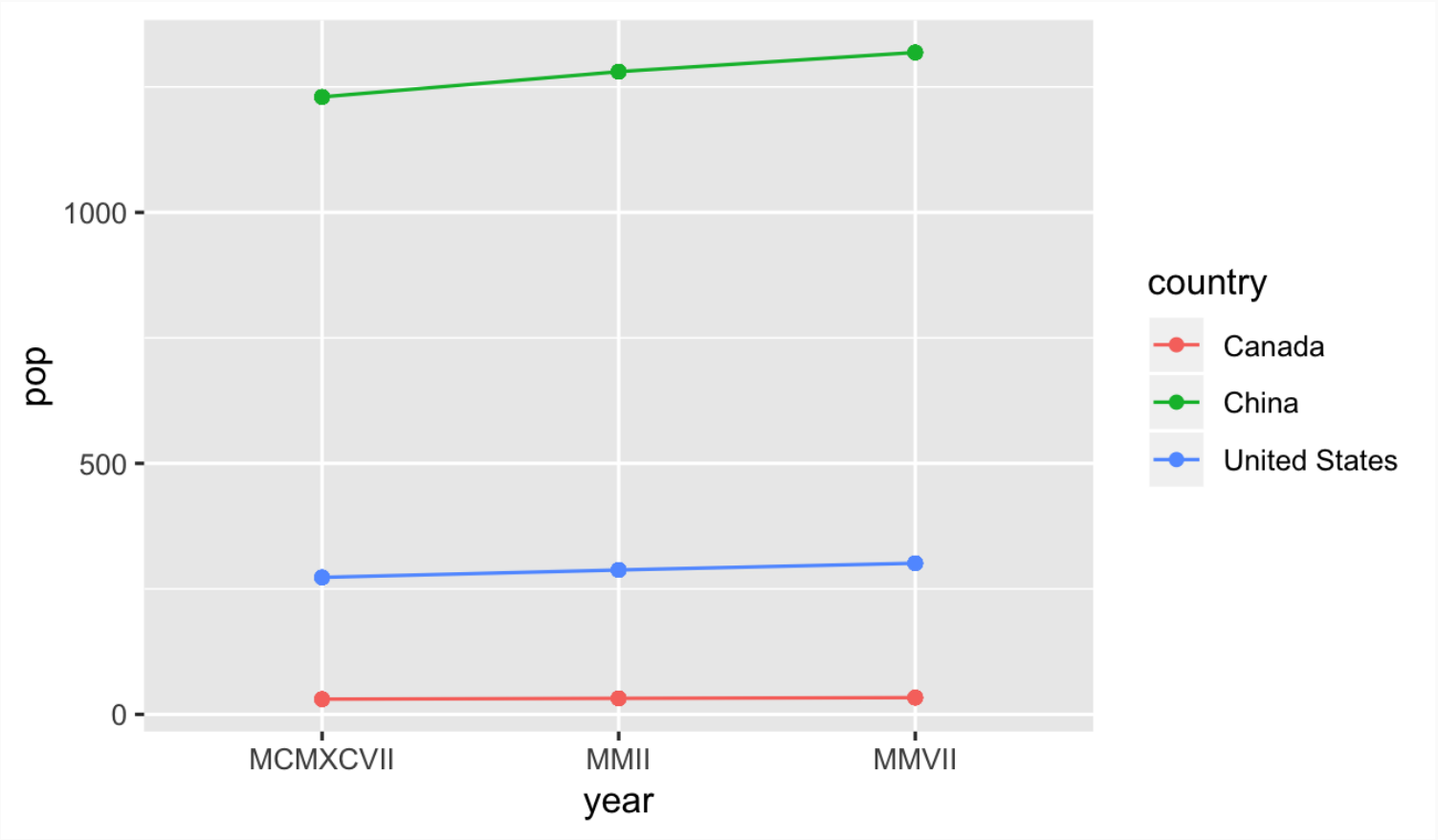
Labels

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()
```

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

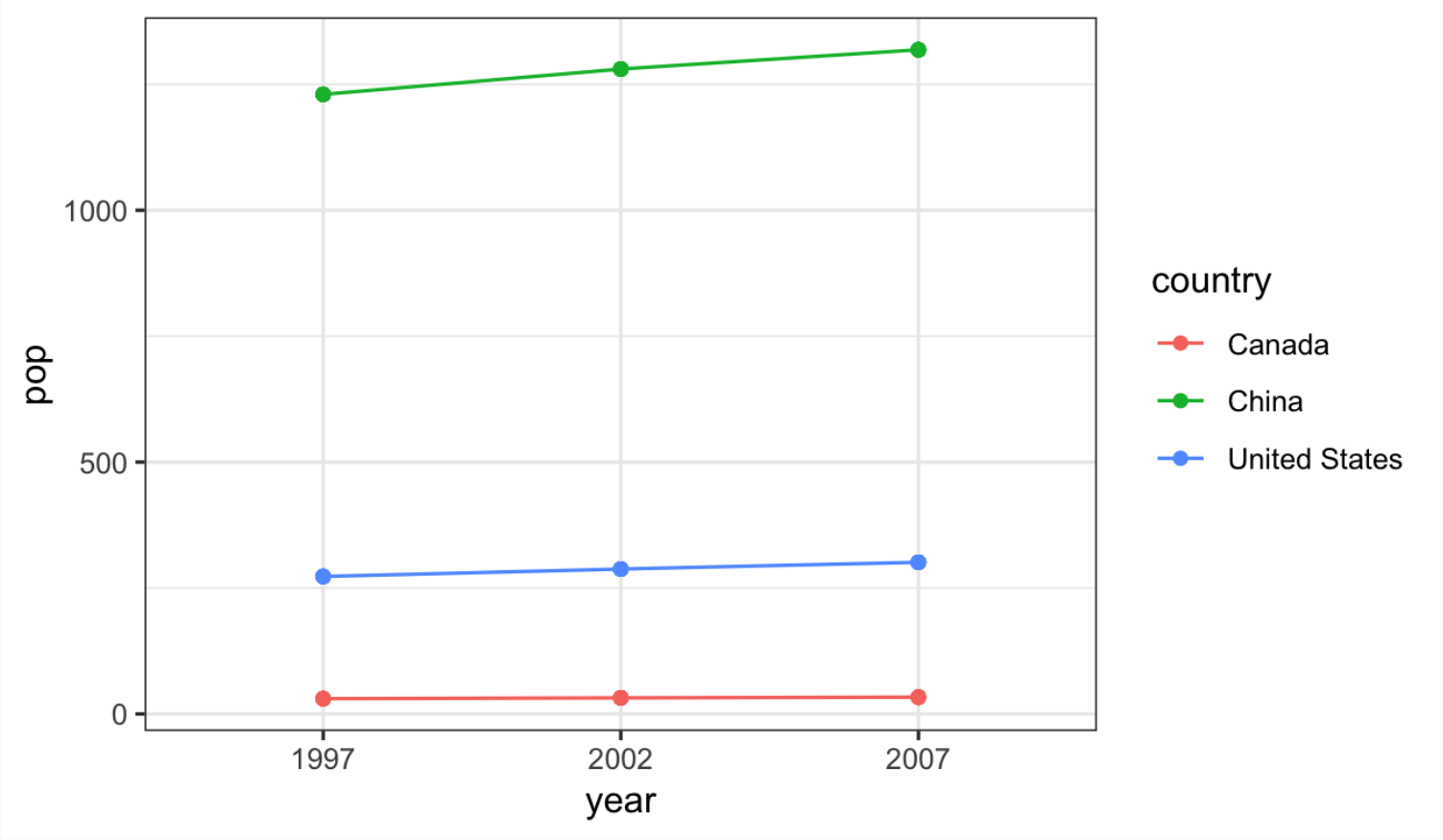
Coords

Scales

Theme

```
+ theme()
```

```
g + theme_bw()
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

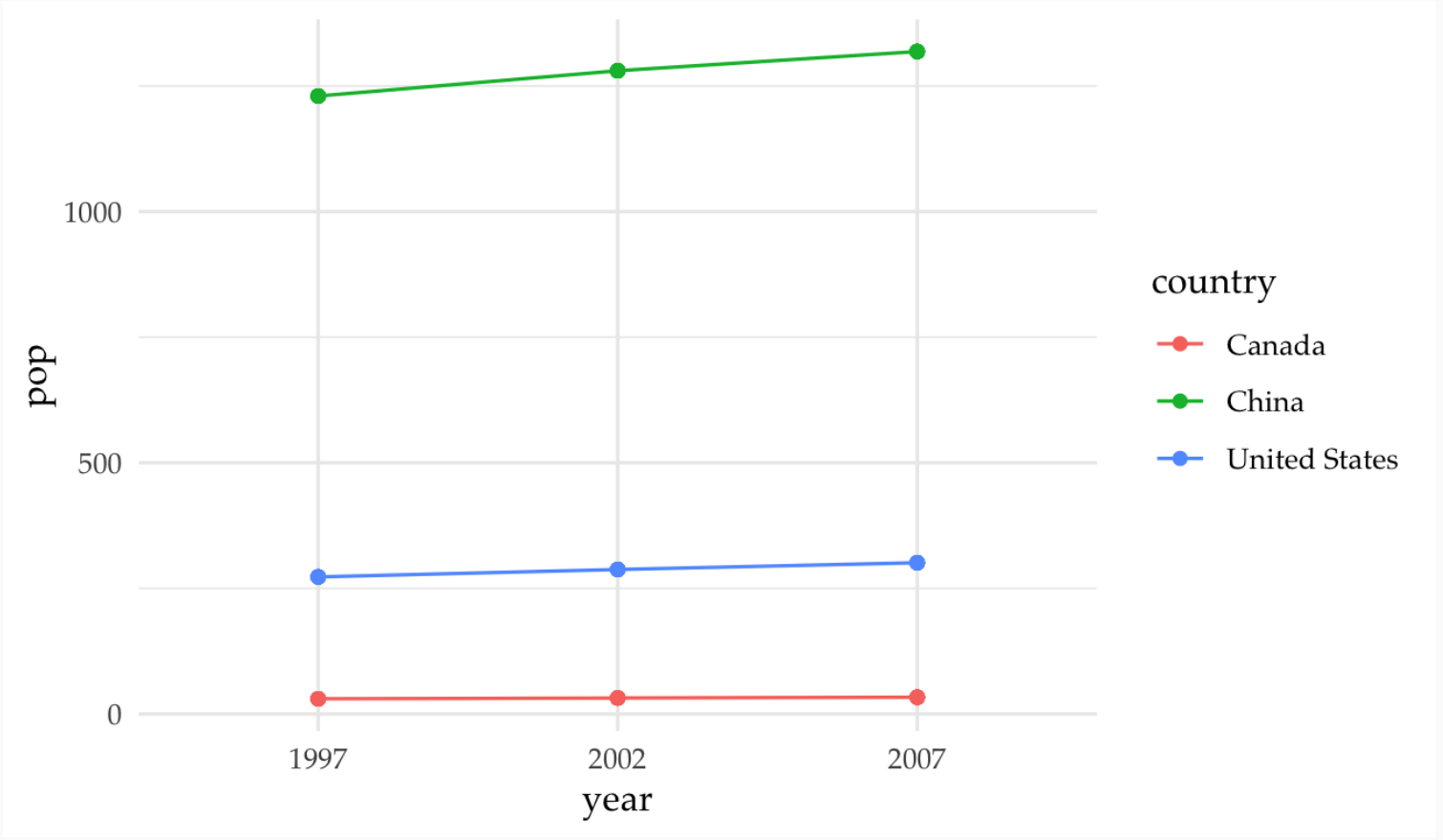
Coords

Scales

Theme

```
+ theme()
```

```
g + theme_minimal() + theme(text = element_text(family = "Palatino"))
```



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

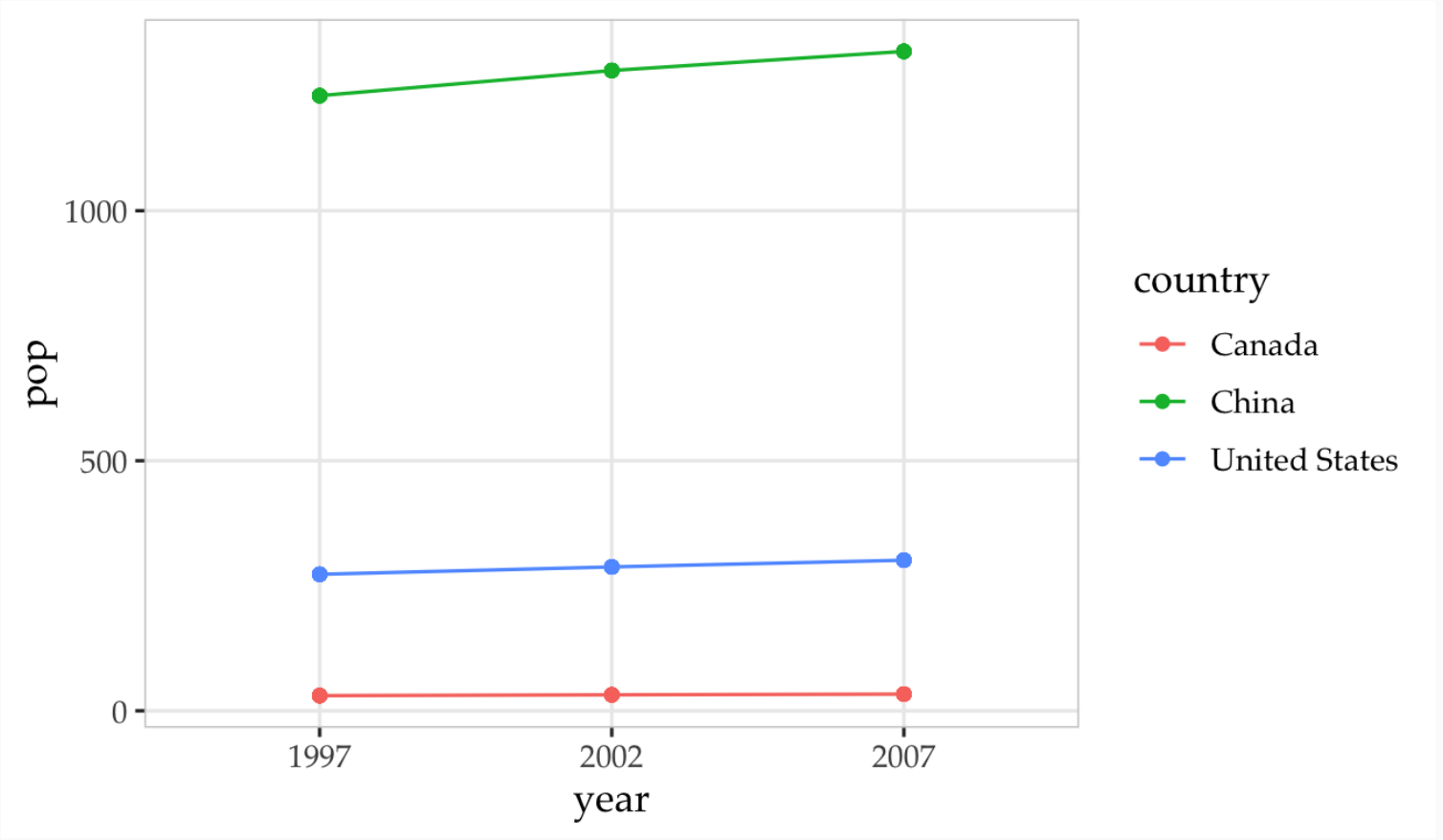
Coords

Scales

Theme

```
+ theme()
```

```
g
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

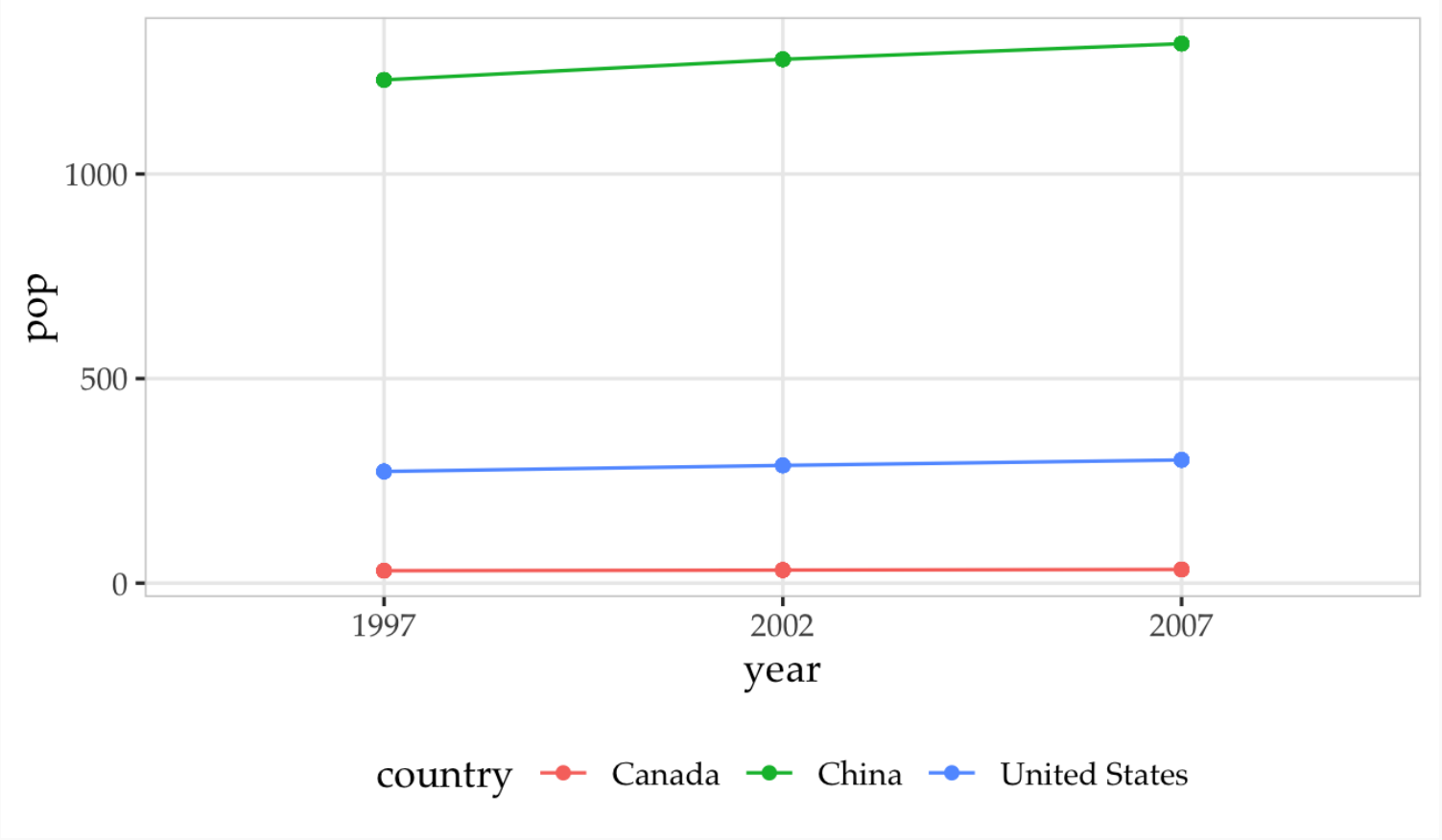
Coords

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)

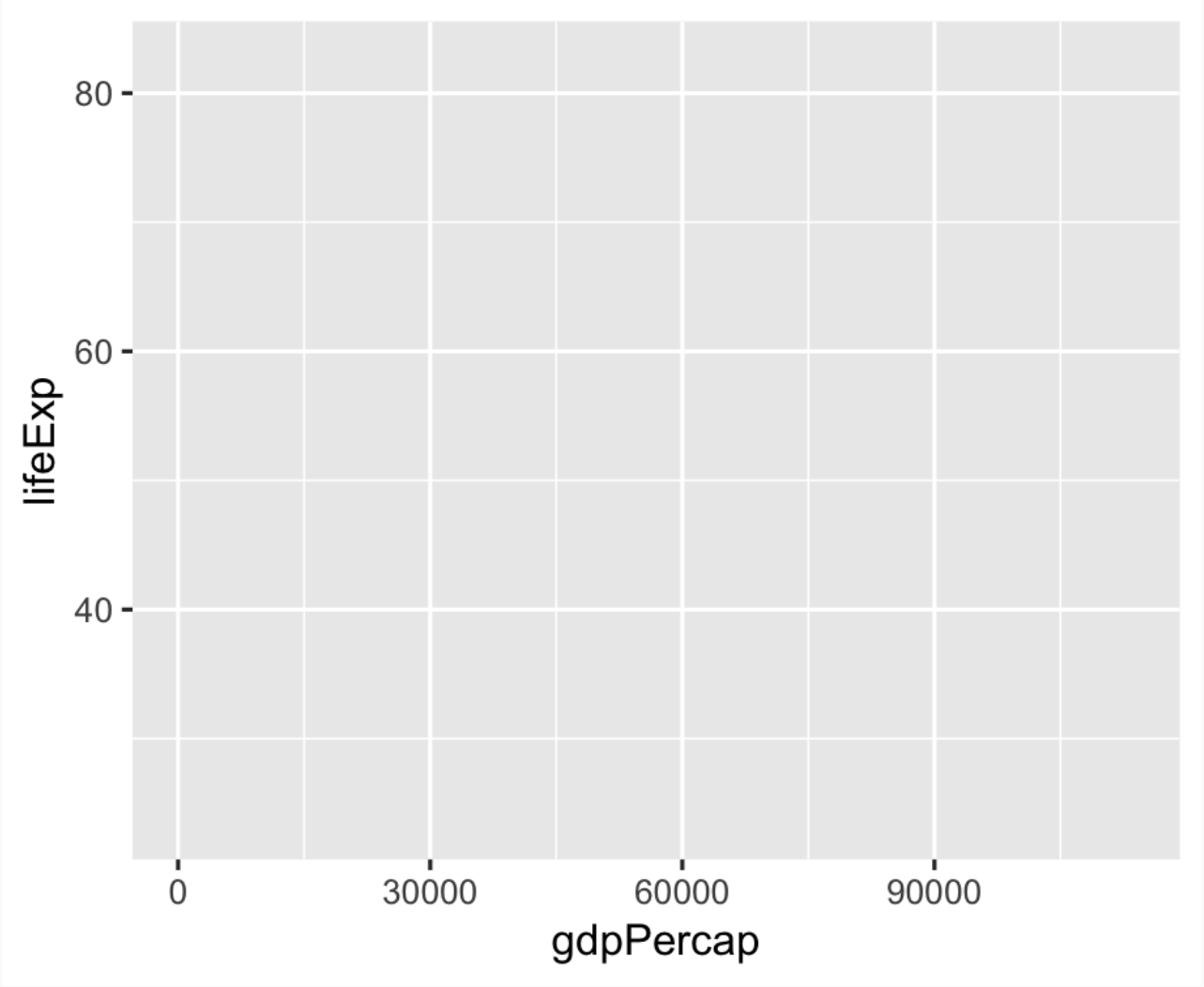
```
Observations: 1,704
Variables: 6
$ country   <fct> Afghanistan, Afghanistan, Afghanistan, Afghanistan, Afghanistan,
$ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia,
$ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, 2002,
$ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.822, 4
$ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 1288181
$ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, 978.0
```

glimpse(gapminder)

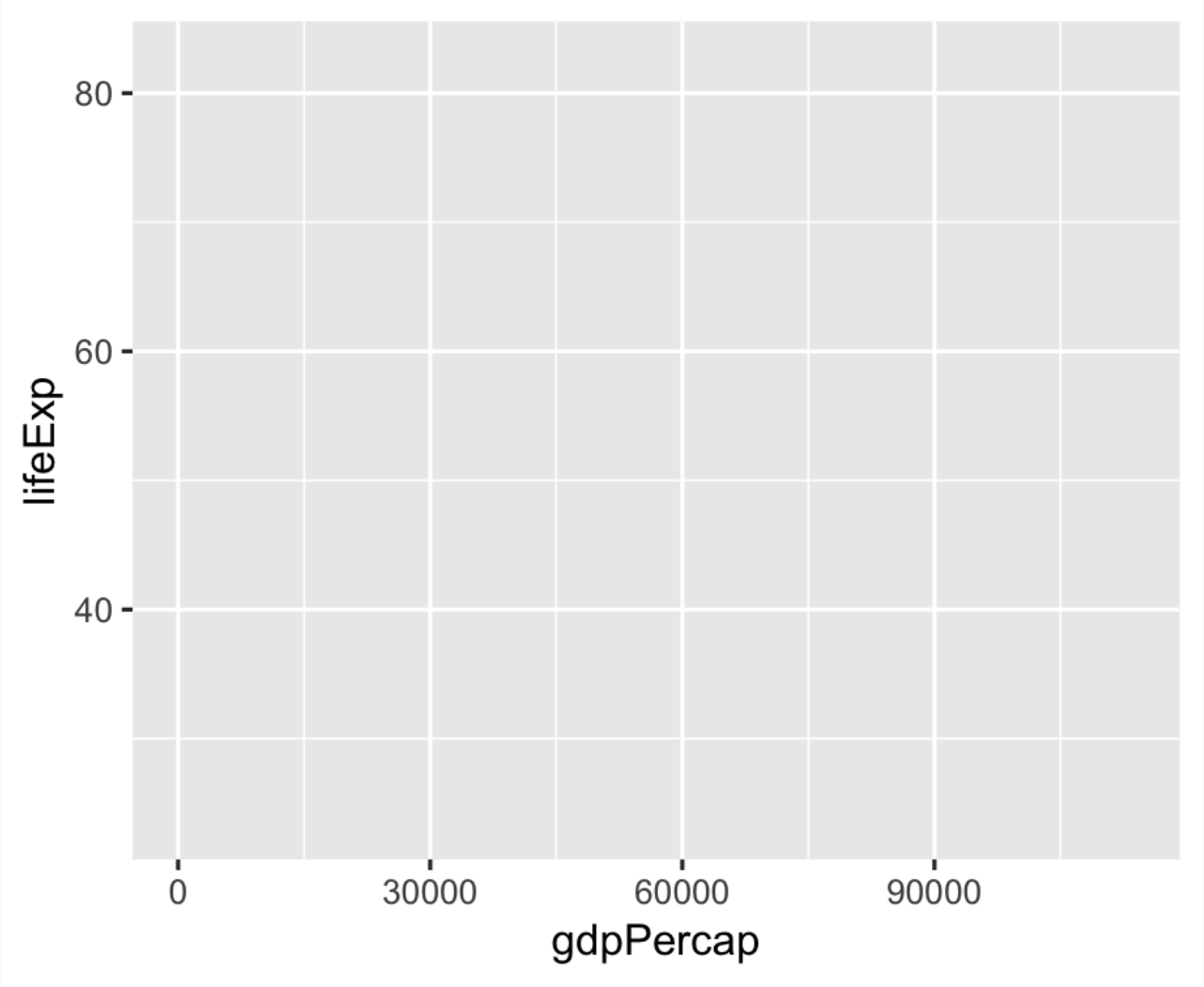
```
Observations: 1,704
Variables: 6
$ country   <fct> Afghanistan, Afghanistan, Afghanistan, Afghanistan, Afghanistan,
$ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia,
$ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, 2002,
$ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.822, 4
$ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 1288181
$ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, 978.0
```

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

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

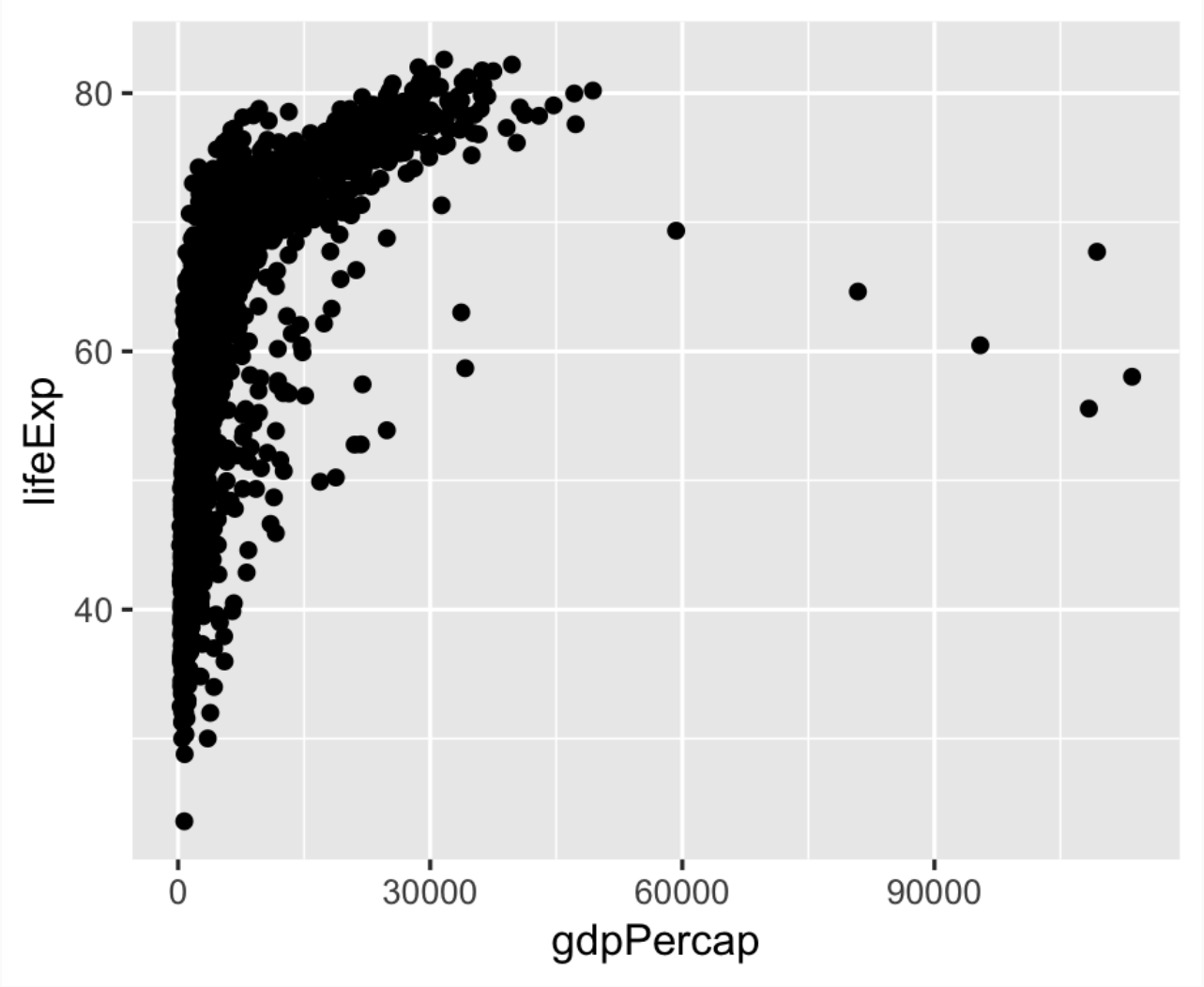


```
ggplot(gapminder) +
  aes(x = gdpPercap,
      y = lifeExp)
```

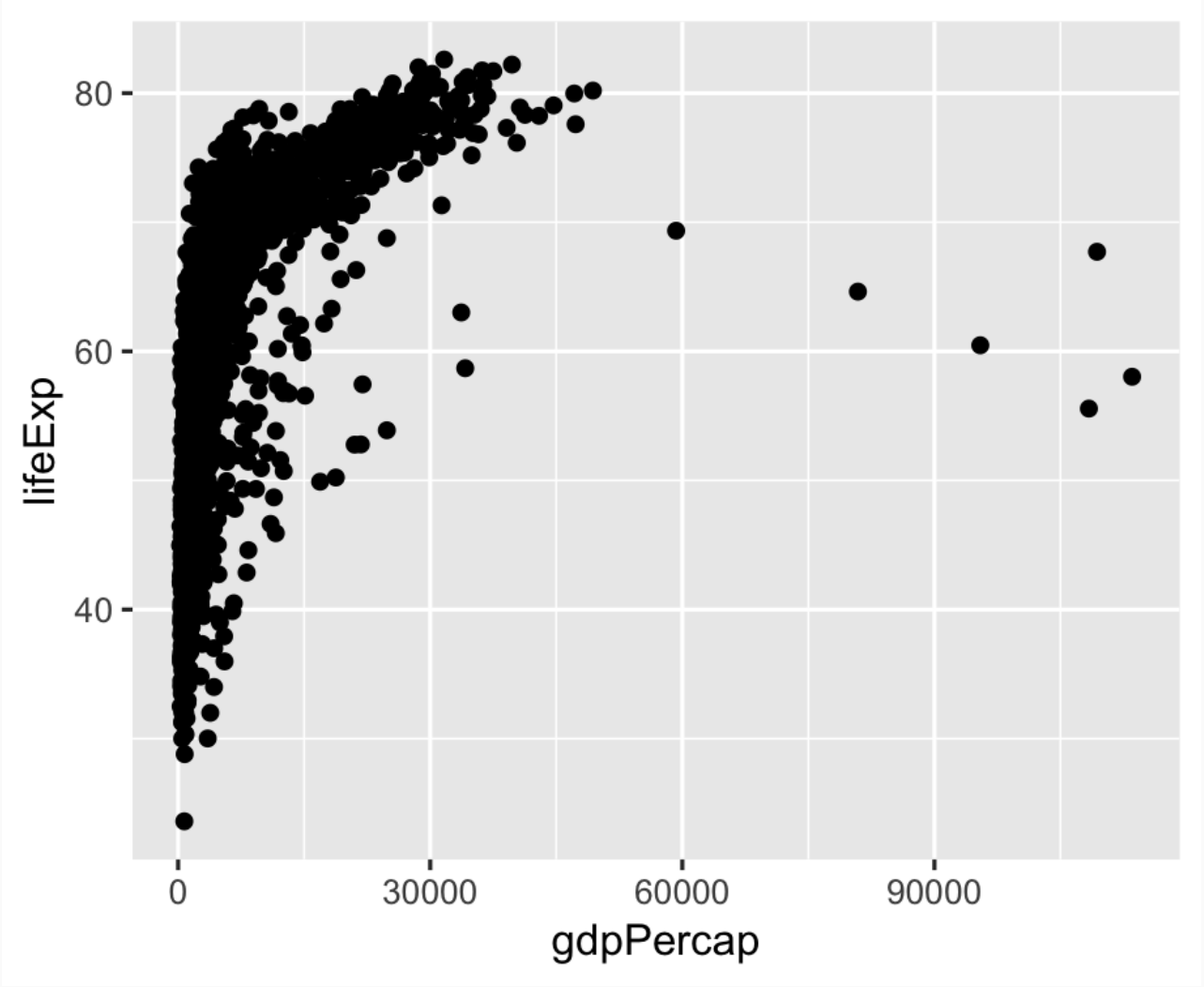


Add points...

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

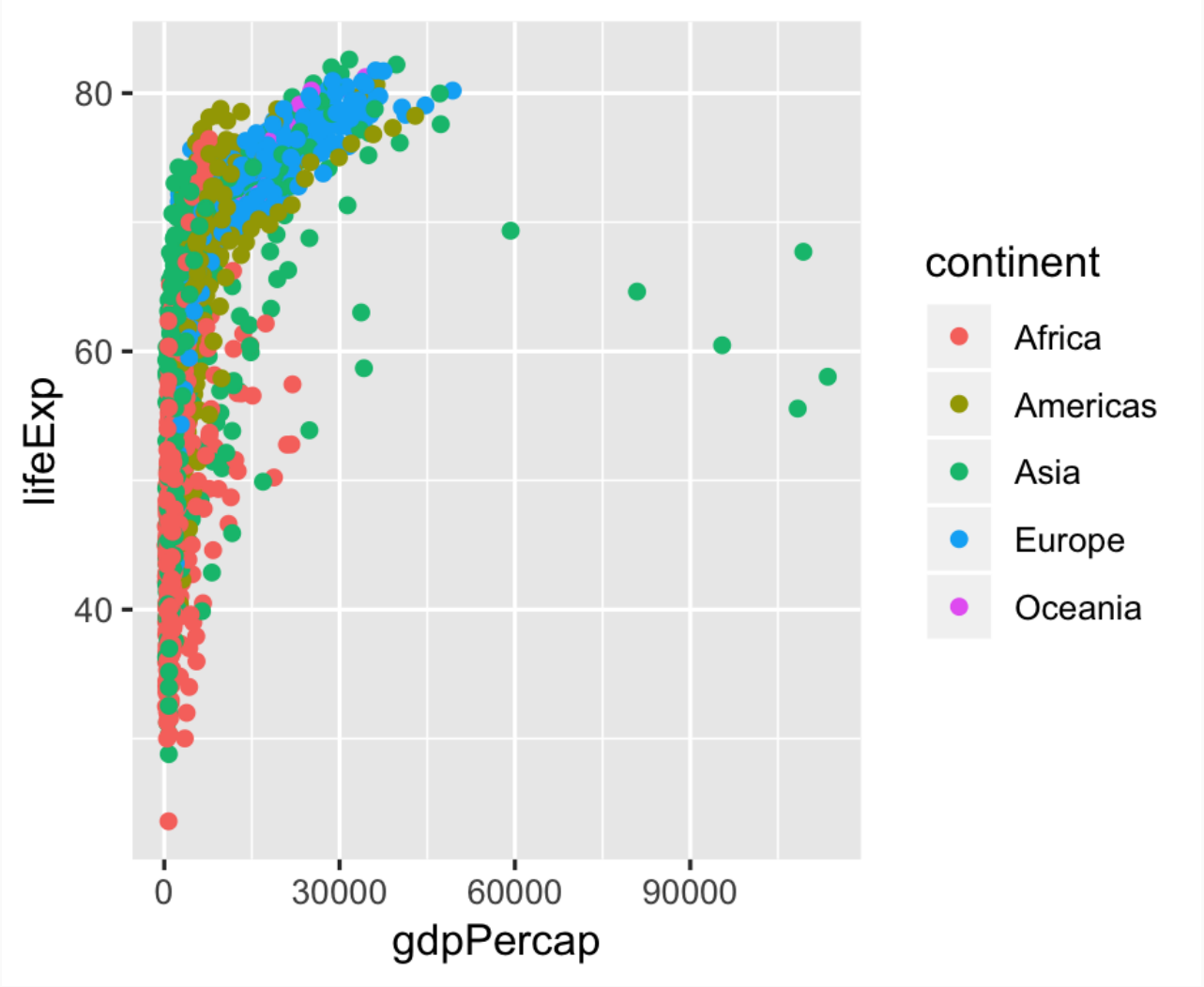


```
ggplot(gapminder) +
  aes(x = gdpPercap,
      y = lifeExp) +
  geom_point()
```

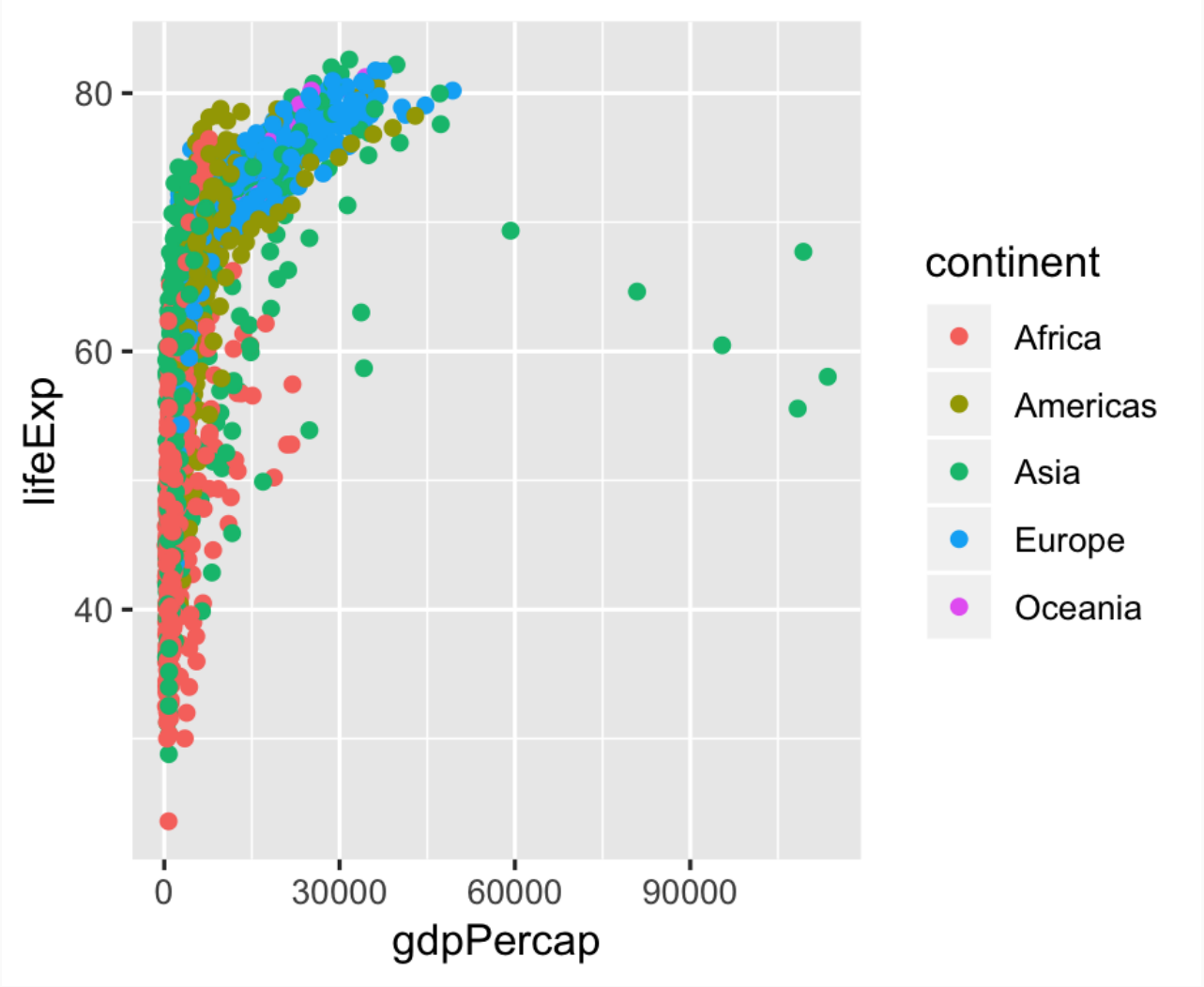


How can I tell countries apart?

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

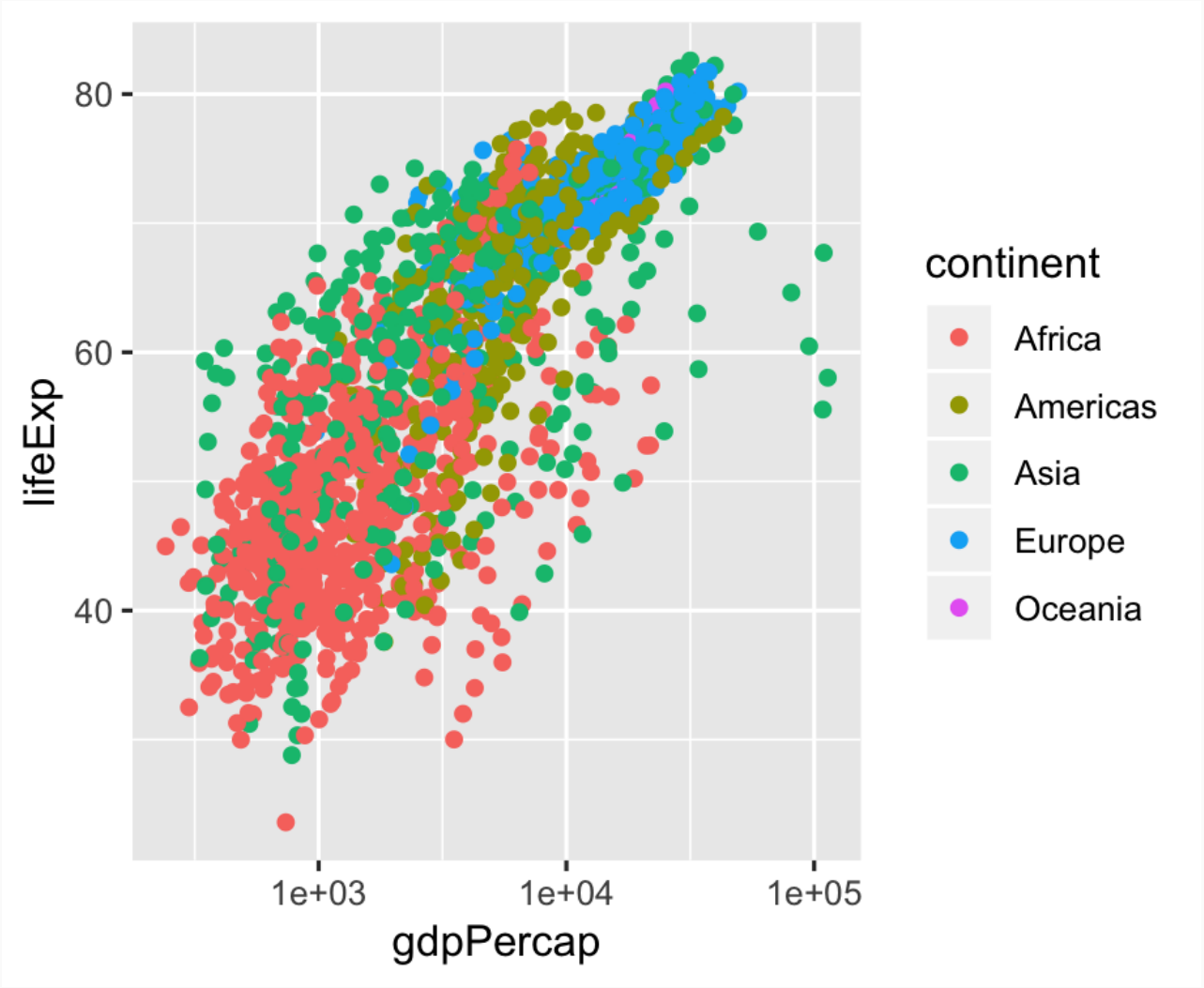


```
ggplot(gapminder) +
  aes(x = gdpPercap,
      y = lifeExp,
      color = continent) +
  geom_point()
```

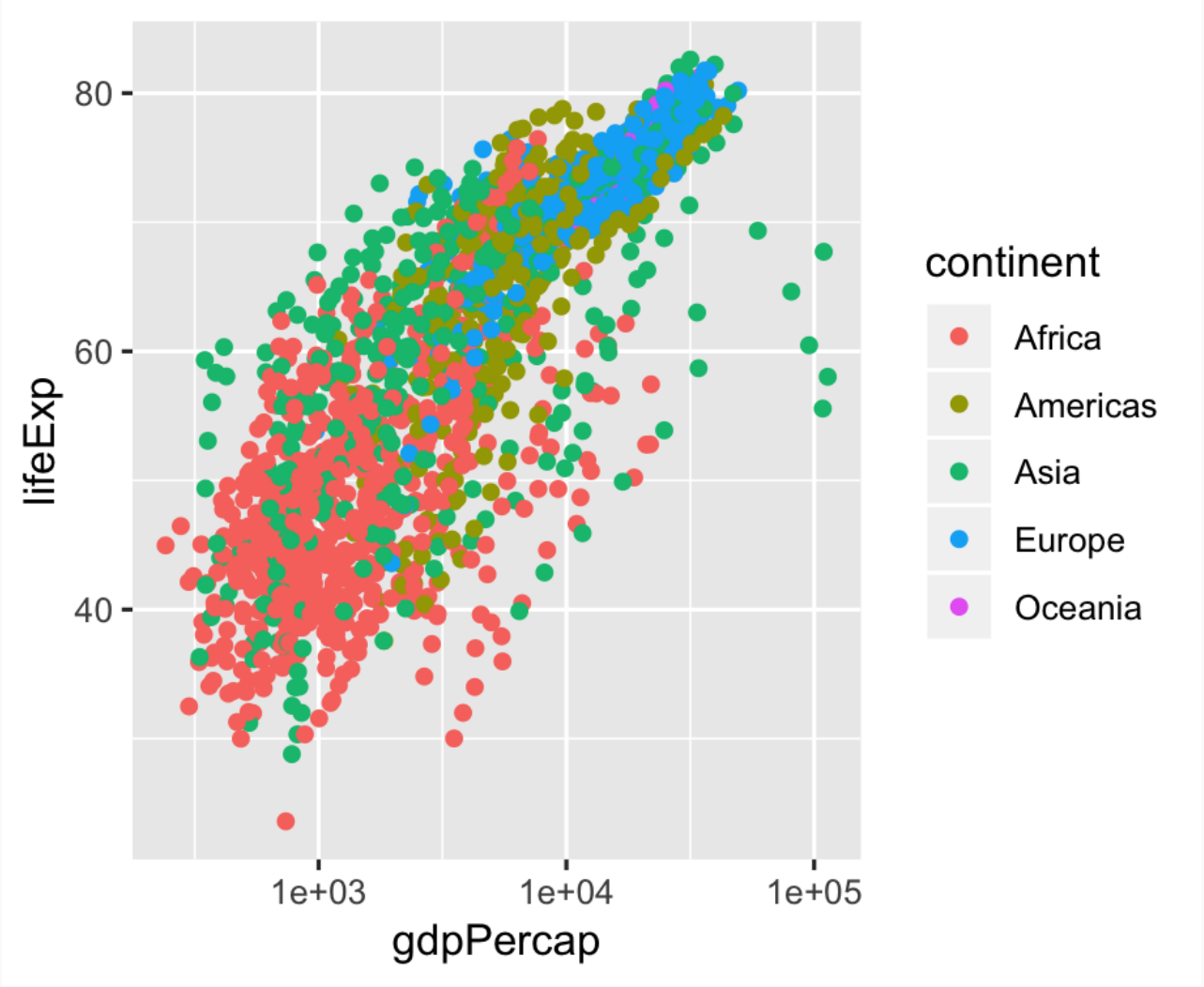


GDP is squished together on the left

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



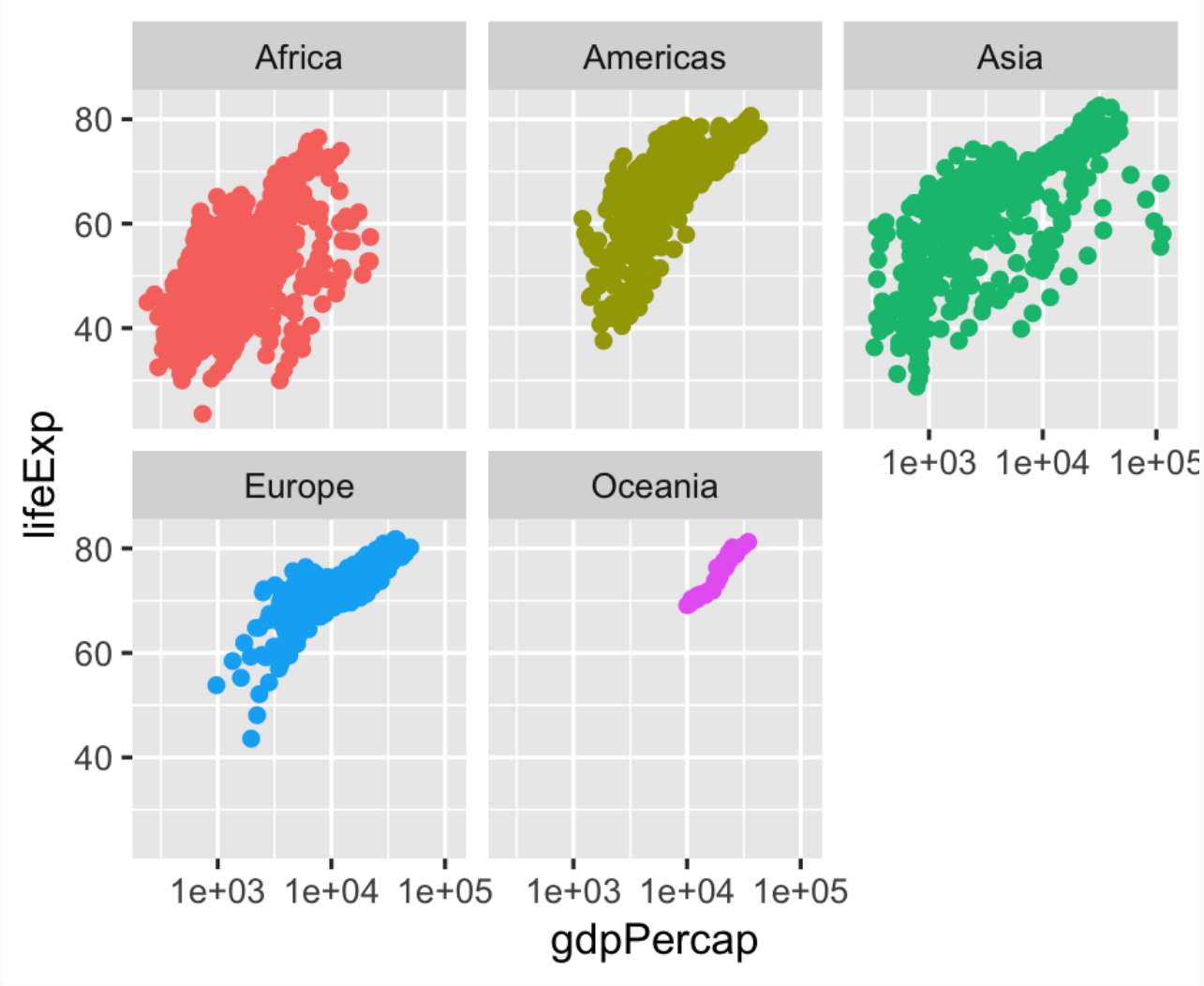
```
ggplot(gapminder) +
  aes(x = gdpPercap,
      y = lifeExp,
      color = continent) +
  geom_point() +
  scale_x_log10()
```



Still lots of overlap in the countries...

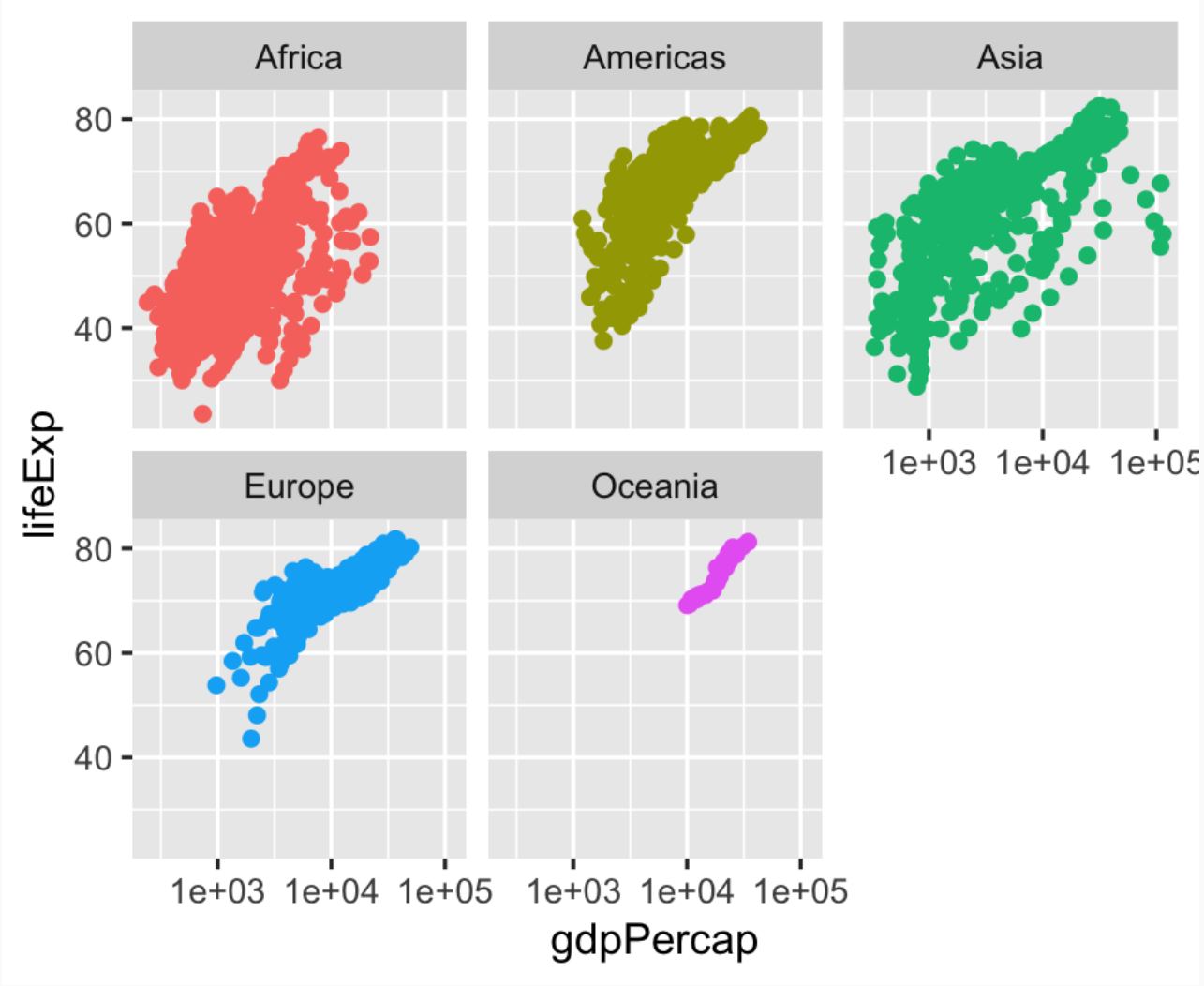
```
ggplot(gapminder) +
  aes(x = gdpPercap,
      y = lifeExp,
      color = continent) +
  geom_point() +
  scale_x_log10() +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```

No need for color legend thanks to facet titles



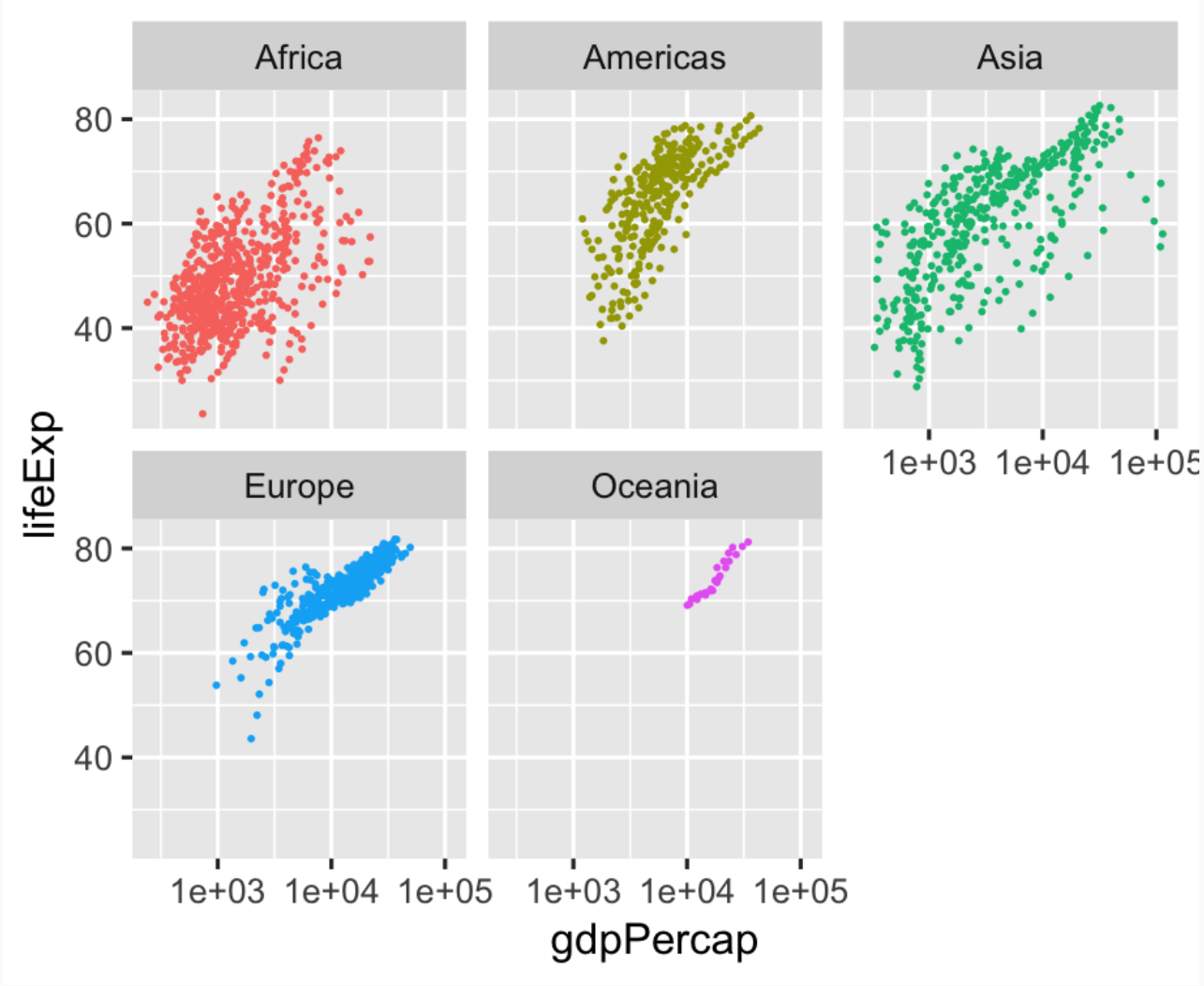
```
ggplot(gapminder) +
  aes(x = gdpPercap,
      y = lifeExp,
      color = continent) +
  geom_point() +
  scale_x_log10() +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```

No need for color legend thanks to facet titles

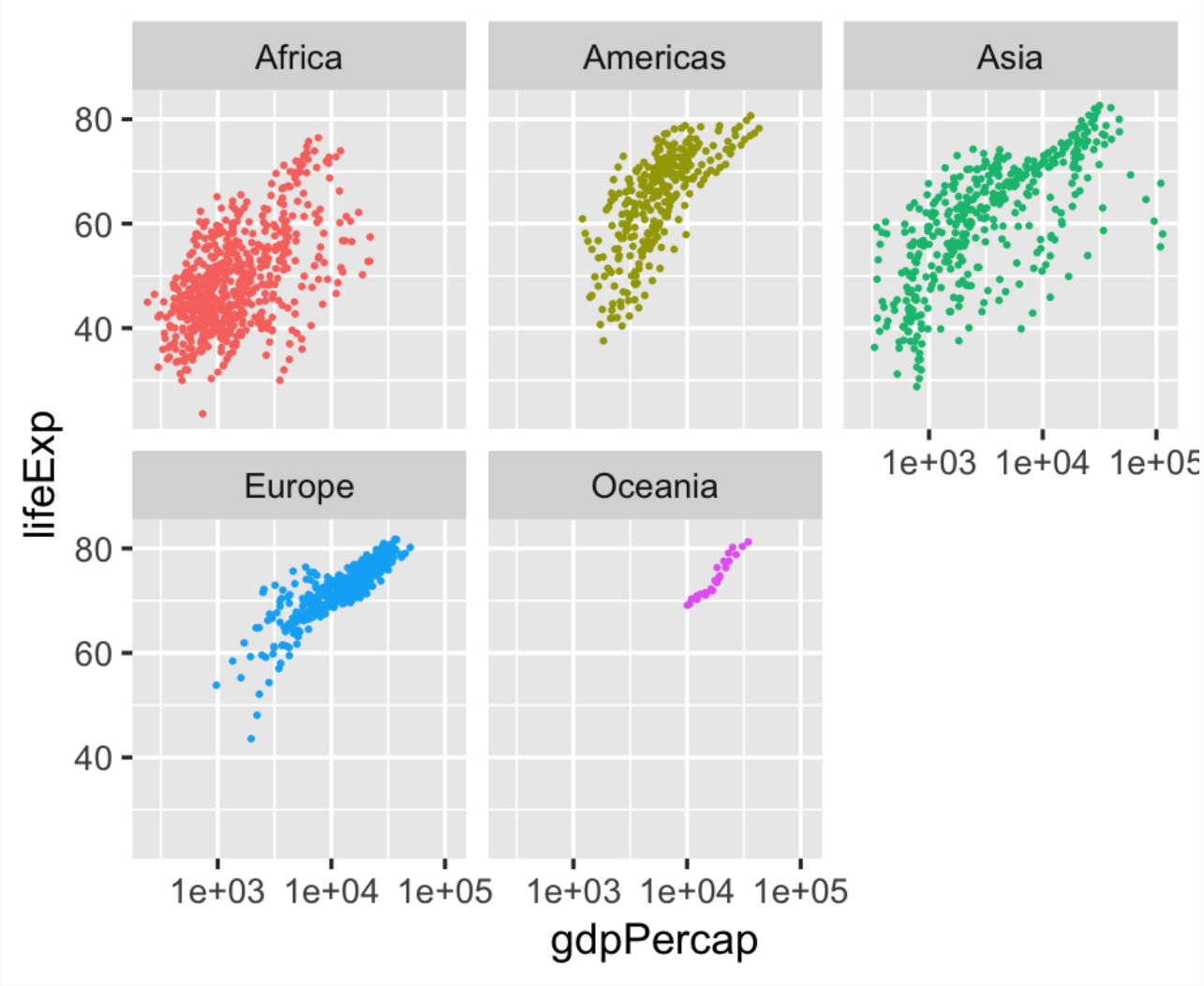


Lots of overplotting due to point size

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

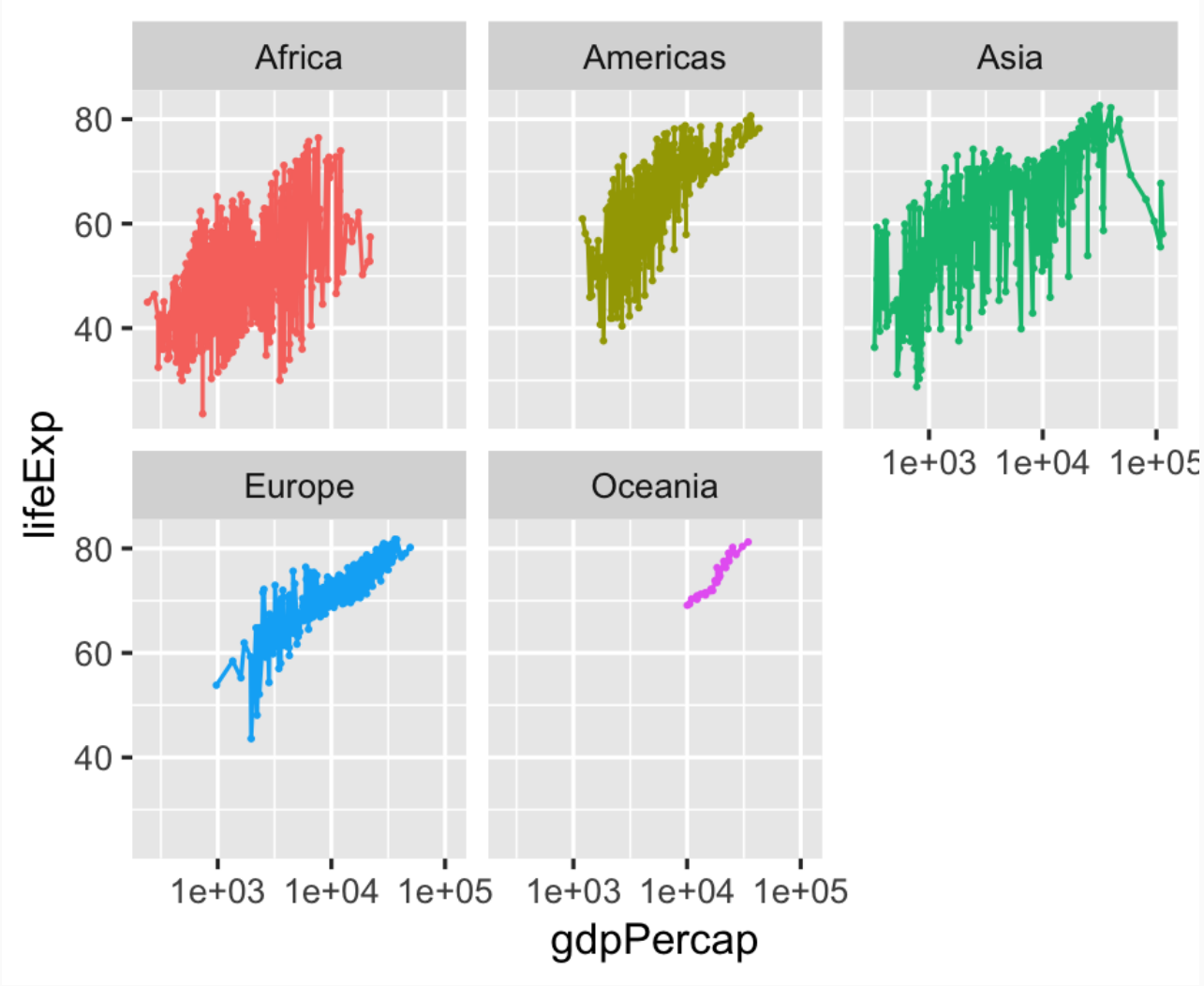


```
ggplot(gapminder) +
  aes(x = gdpPercap,
      y = lifeExp,
      color = continent) +
  geom_point(size = 0.25) +
  scale_x_log10() +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```

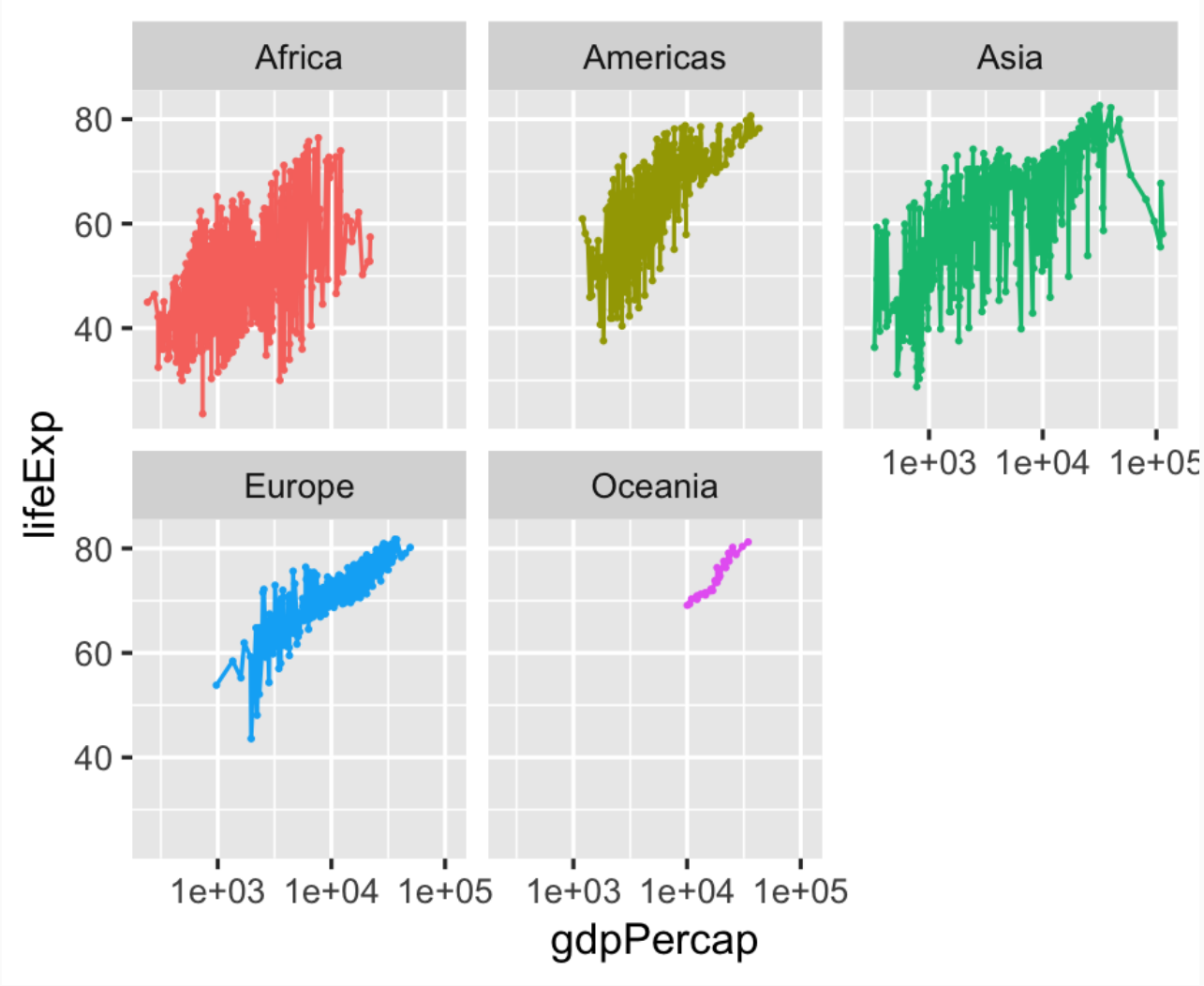


Is there a trend?

```
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)
```

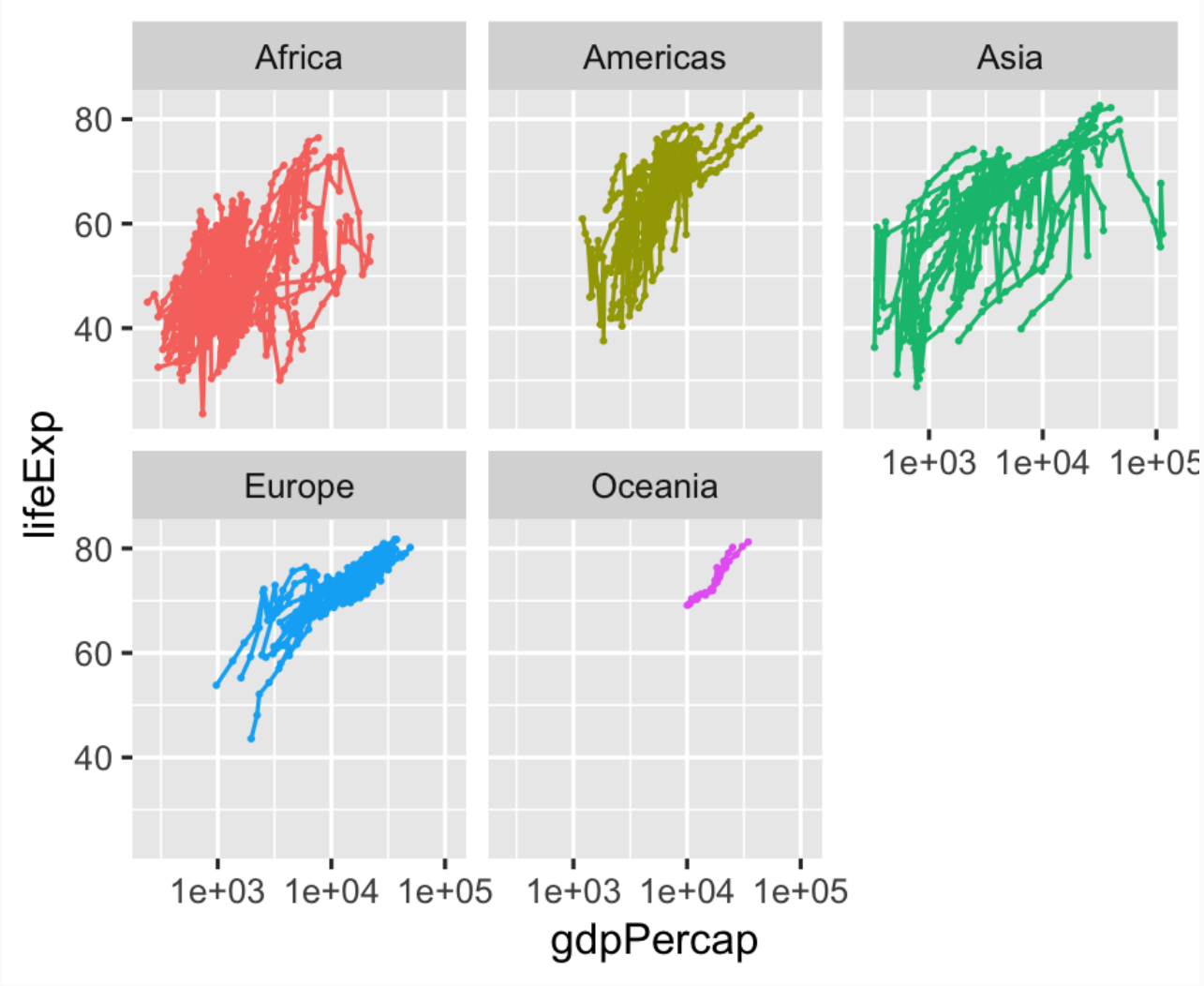


```
ggplot(gapminder) +
  aes(x = gdpPercap,
      y = lifeExp,
      color = continent) +
  geom_line() +
  geom_point(size = 0.25) +
  scale_x_log10() +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```

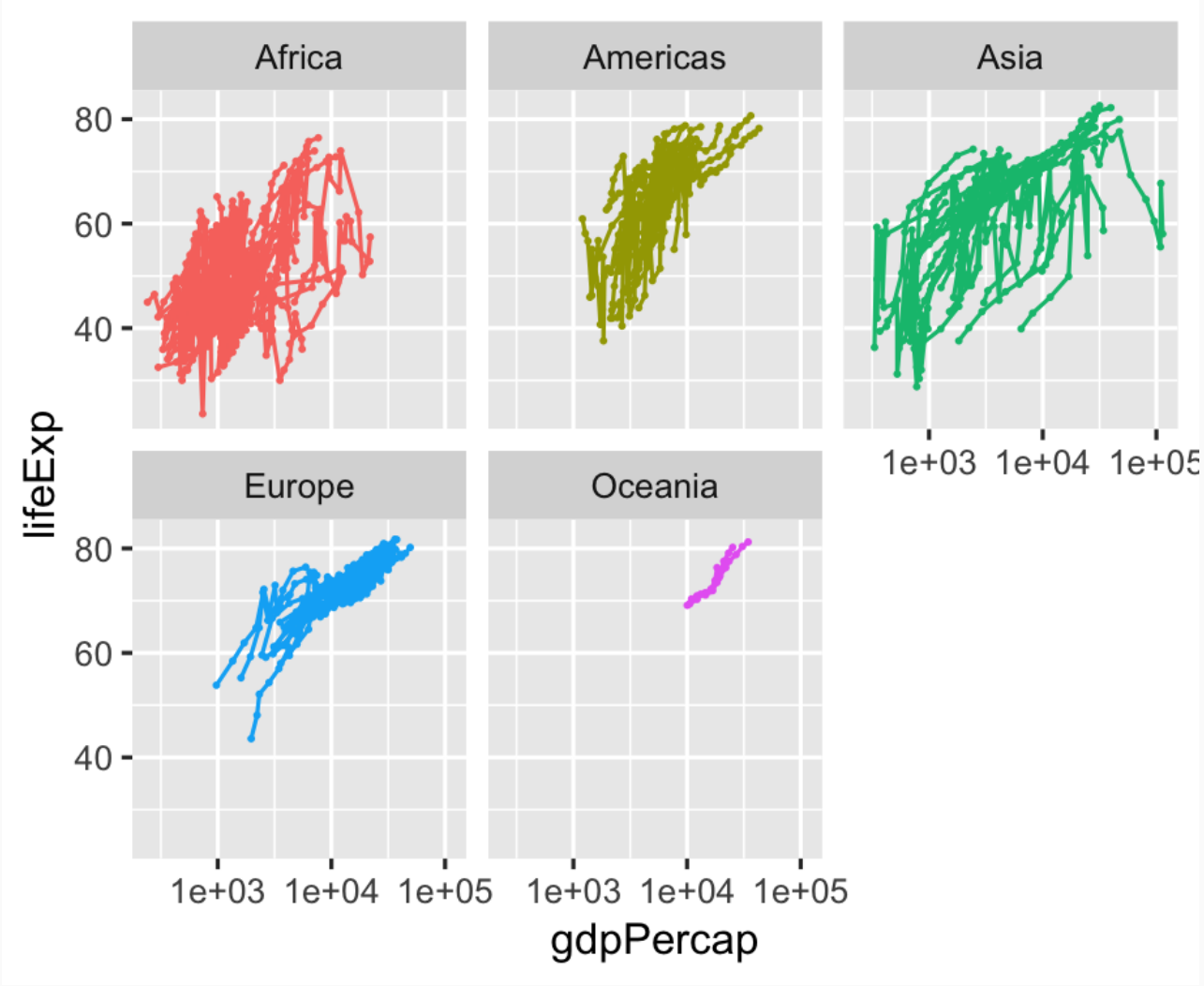


Okay, that line just connected all of the points sequentially...

```
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)
```

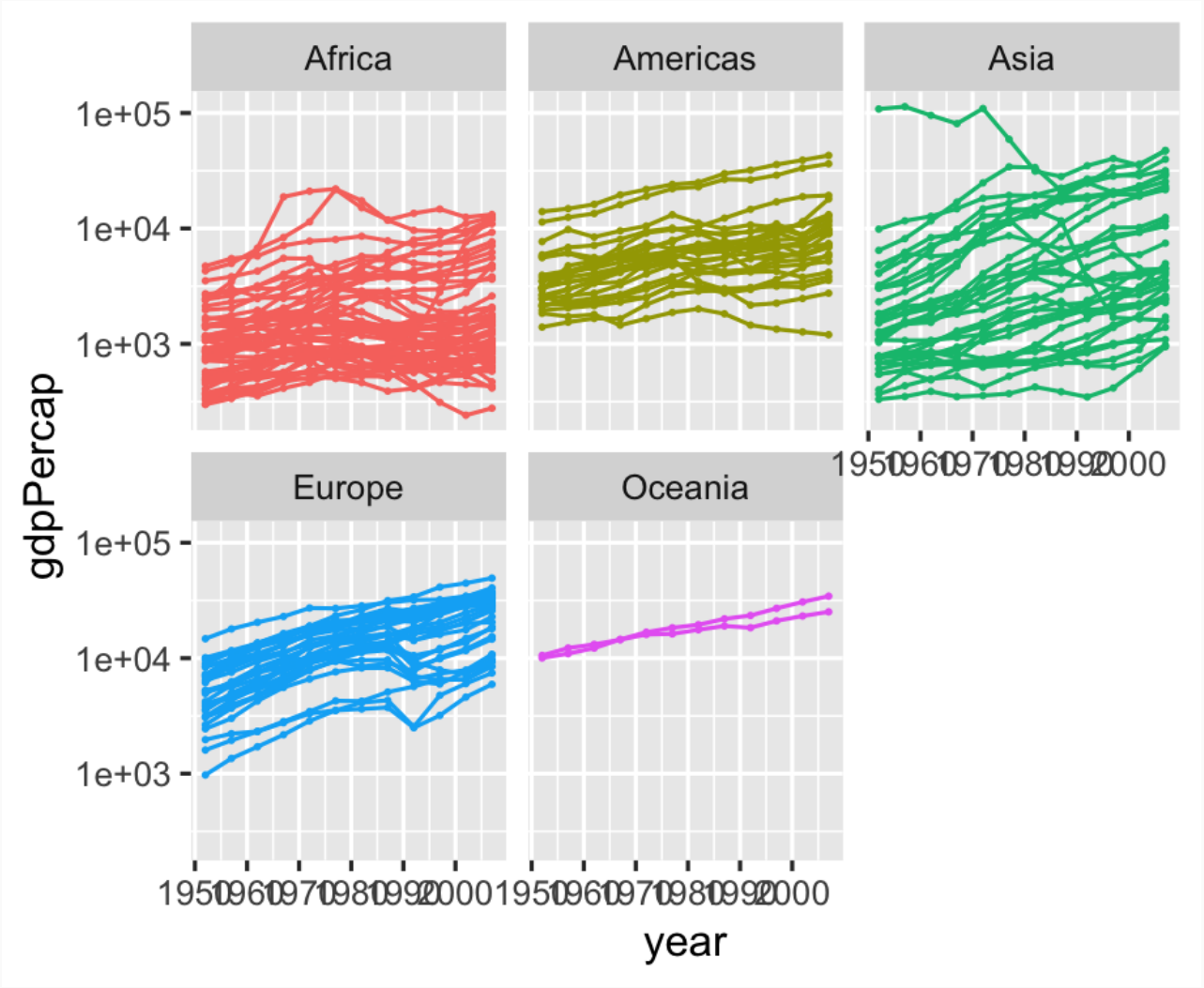


```
ggplot(gapminder) +
  aes(x = gdpPercap,
      y = lifeExp,
      color = continent) +
  geom_line(
    aes(group = country)
  ) +
  geom_point(size = 0.25) +
  scale_x_log10() +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```

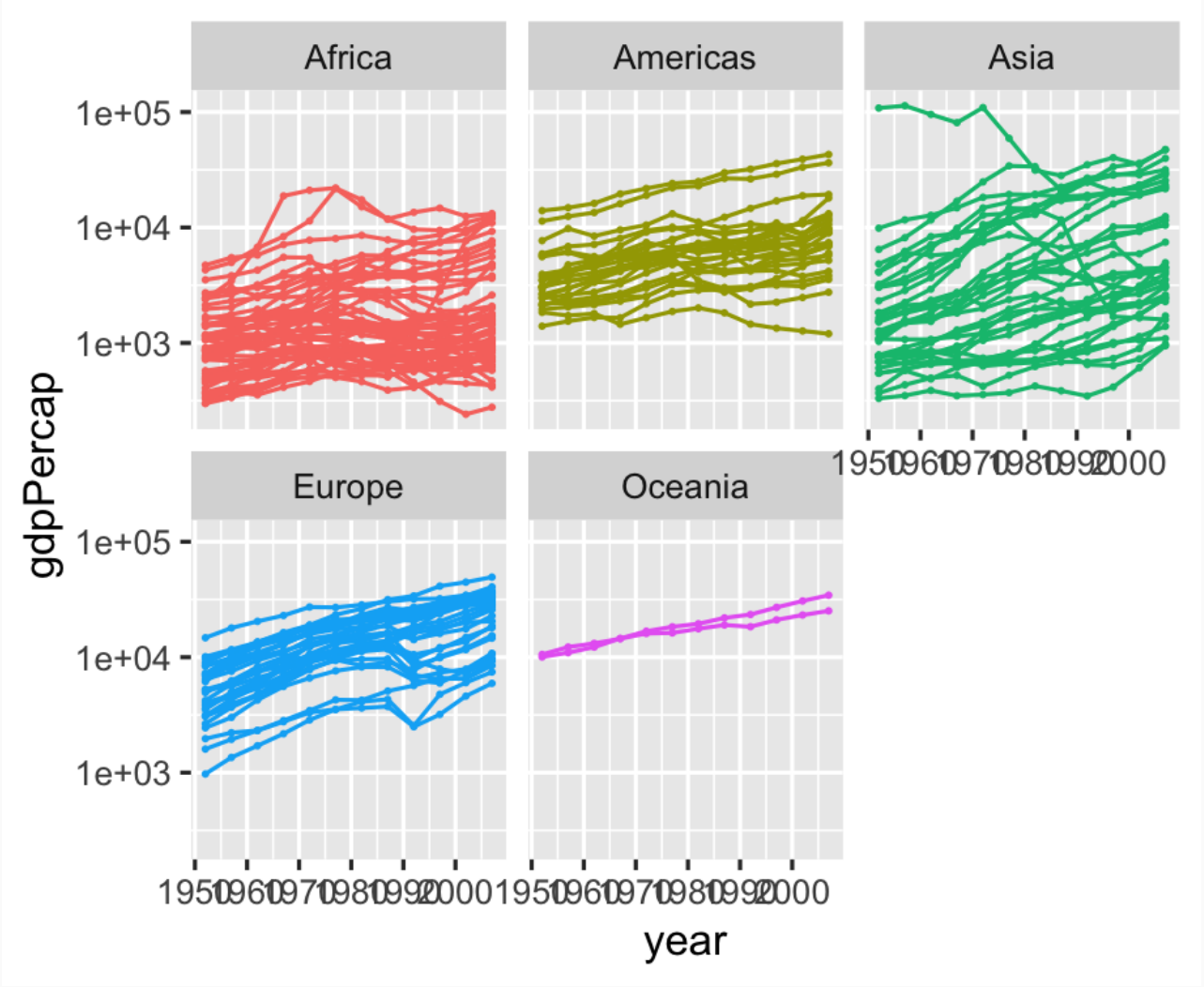


We need time on x-axis!

```
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)
```

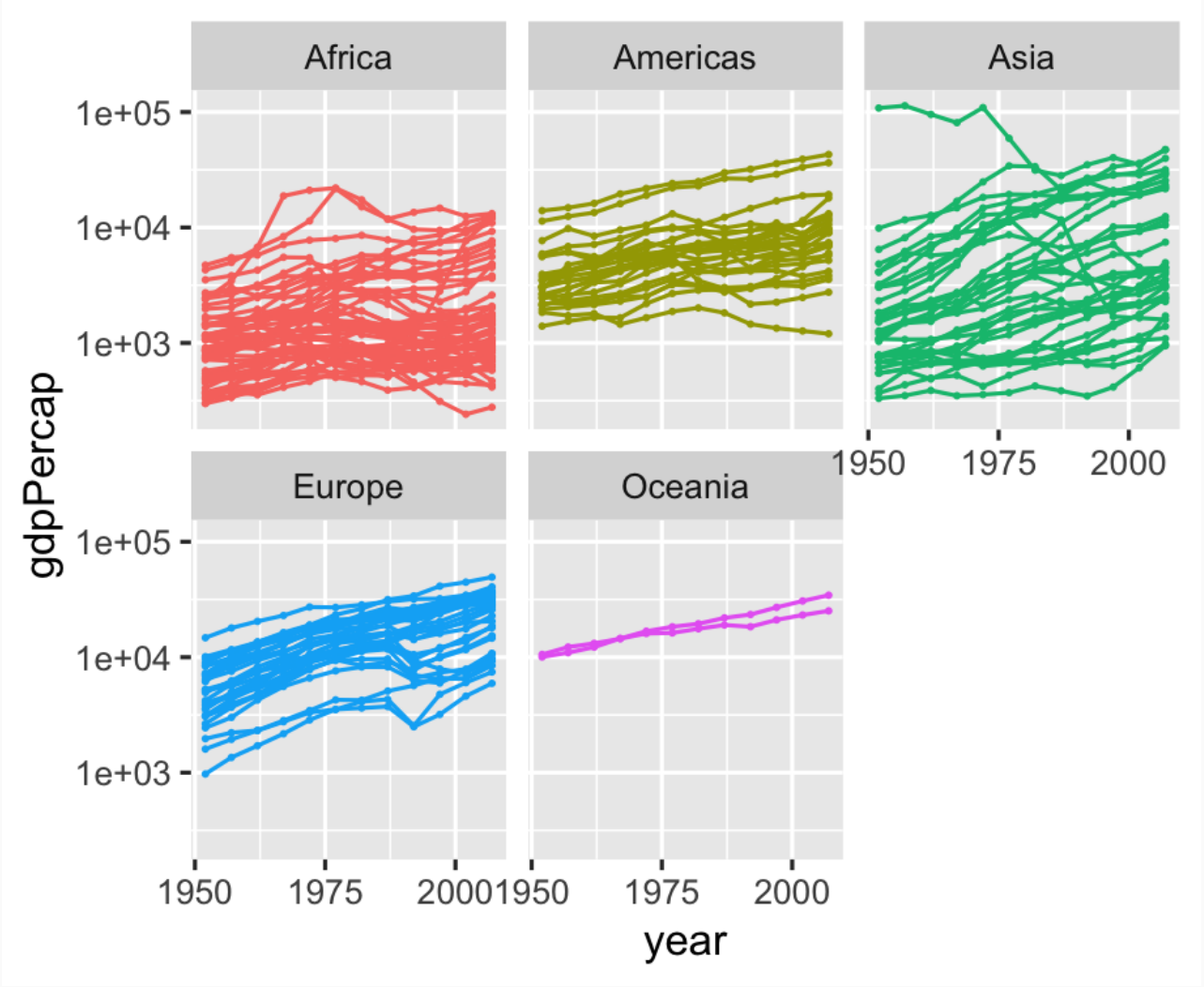


```
ggplot(gapminder) +
  aes(x = year,
      y = gdpPercap,
      color = continent) +
  geom_line(
    aes(group = country)
  ) +
  geom_point(size = 0.25) +
  scale_y_log10() +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```

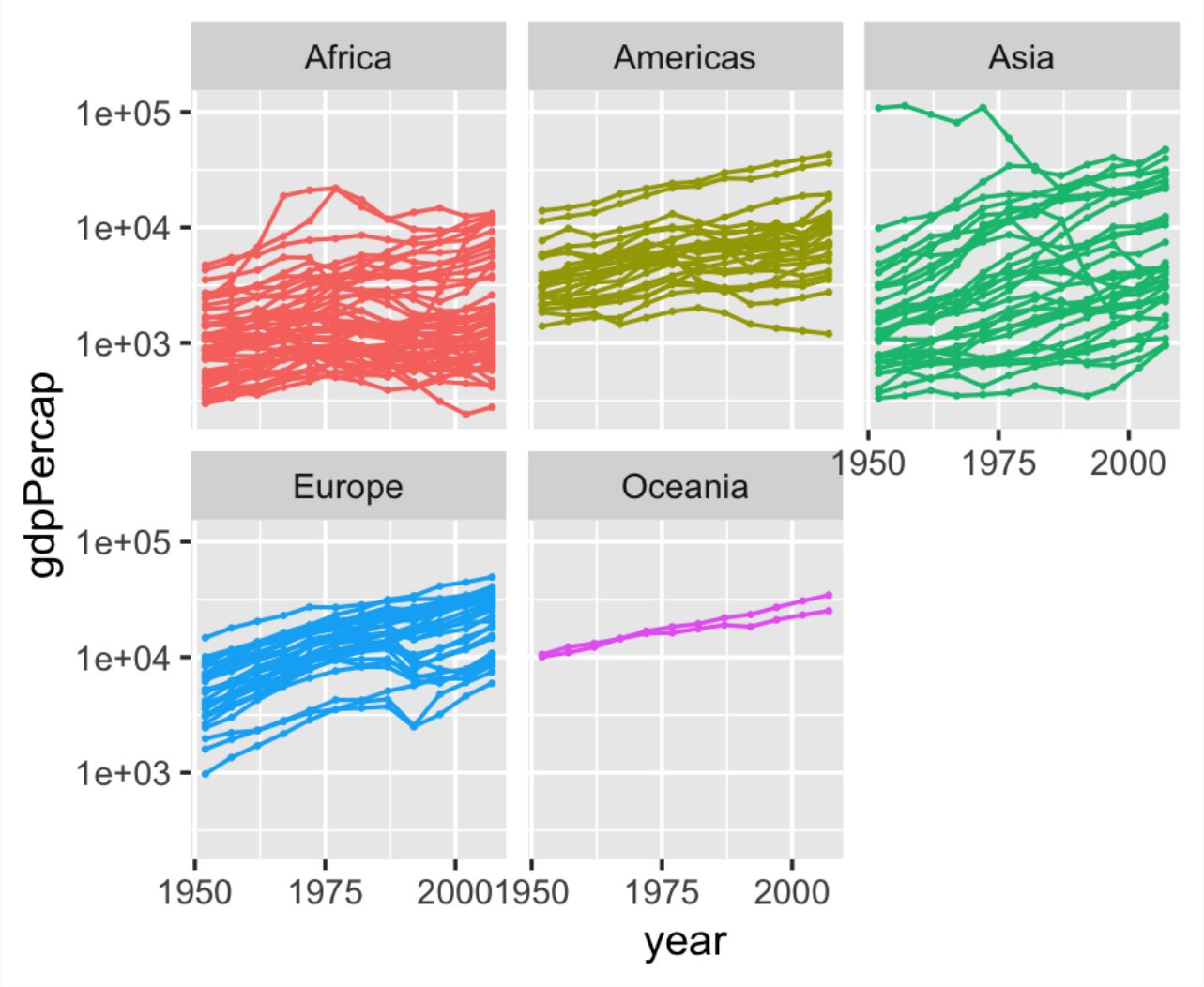


Can't see x-axis labels, though

```
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)
```

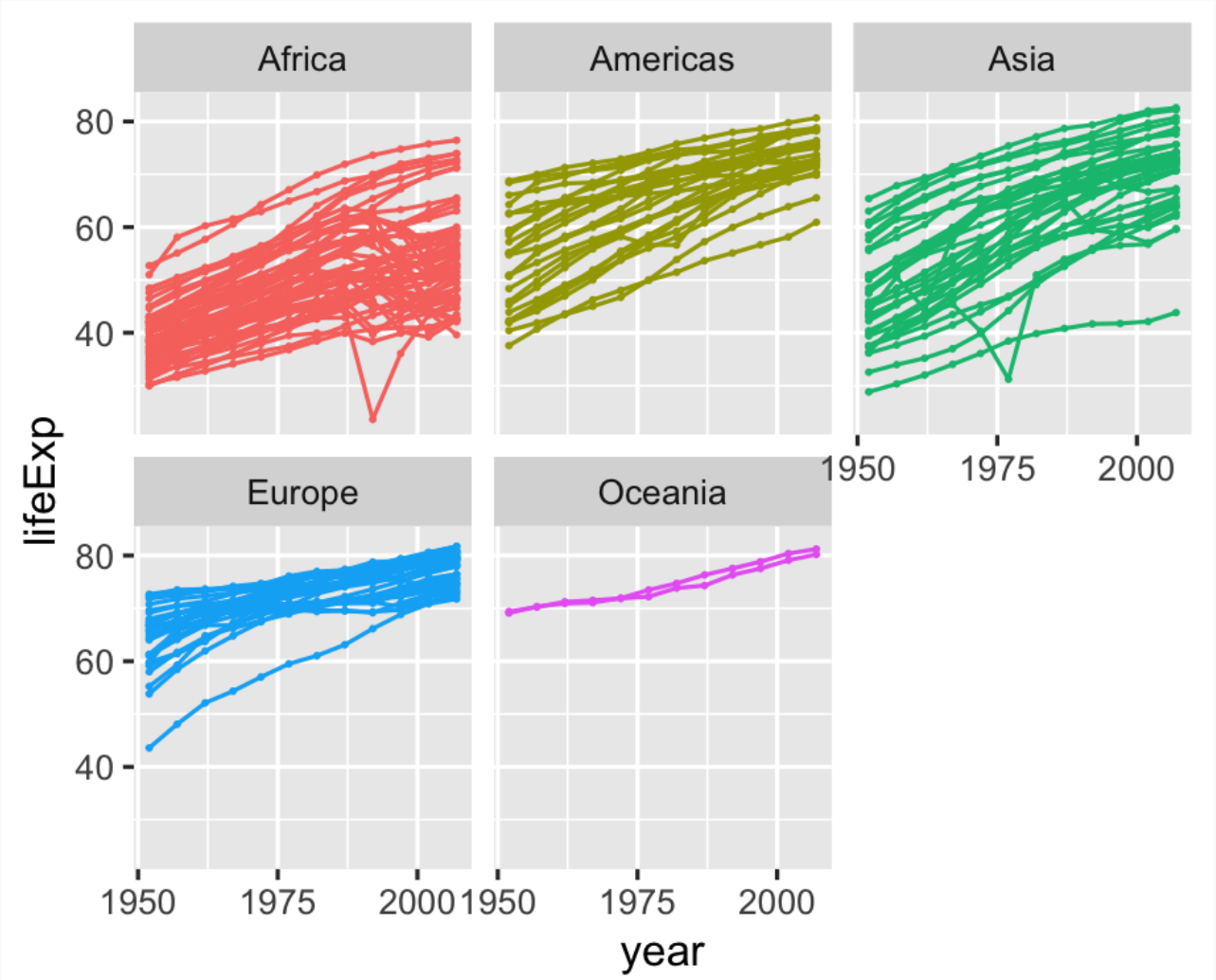


```
ggplot(gapminder) +
  aes(x = year,
      y = gdpPercap,
      color = continent) +
  geom_line(
    aes(group = country)
  ) +
  geom_point(size = 0.25) +
  scale_y_log10() +
  scale_x_continuous(breaks =
    seq(1950, 2000, 25)
  ) +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```

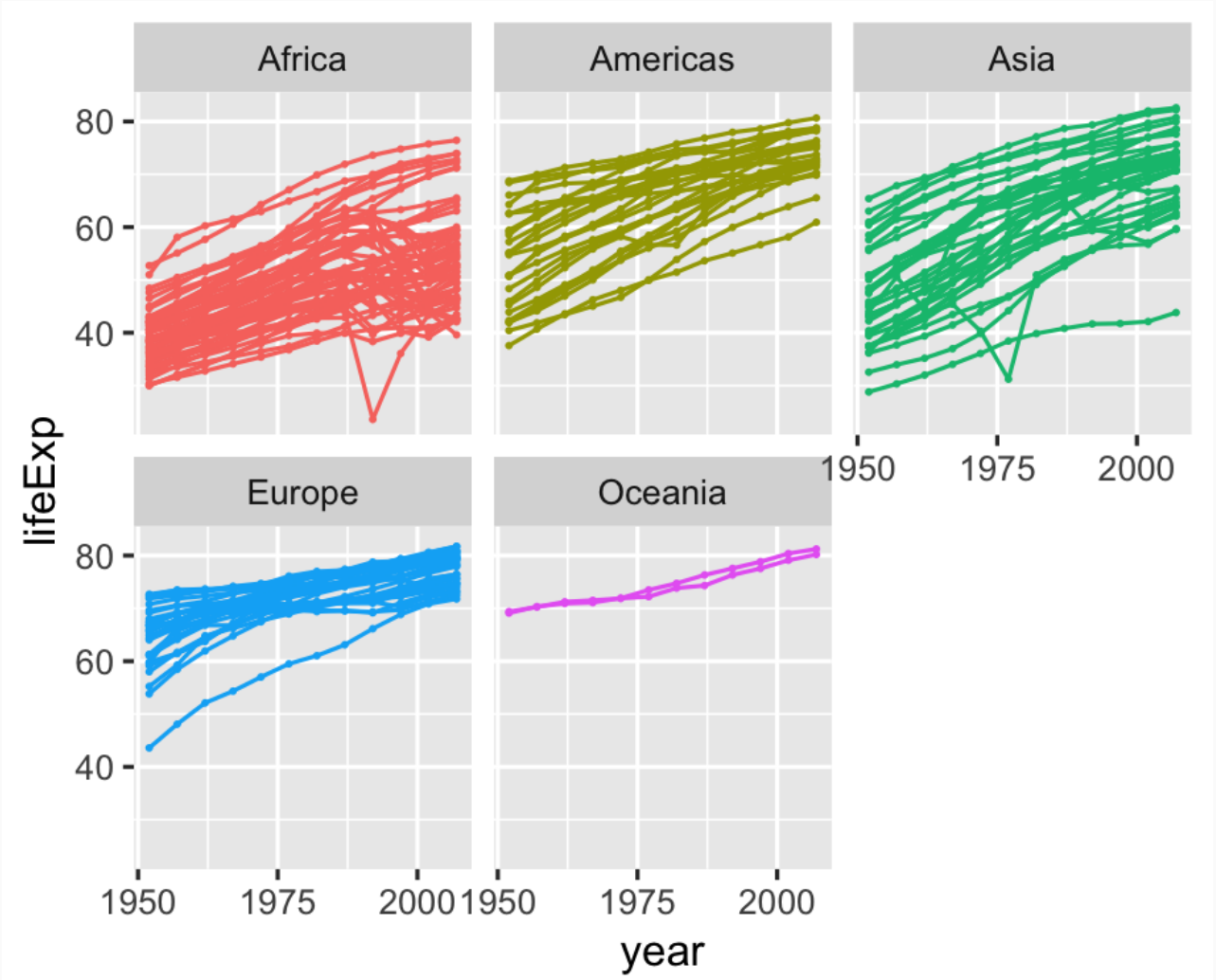


What about life expectancy?

```
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)
```

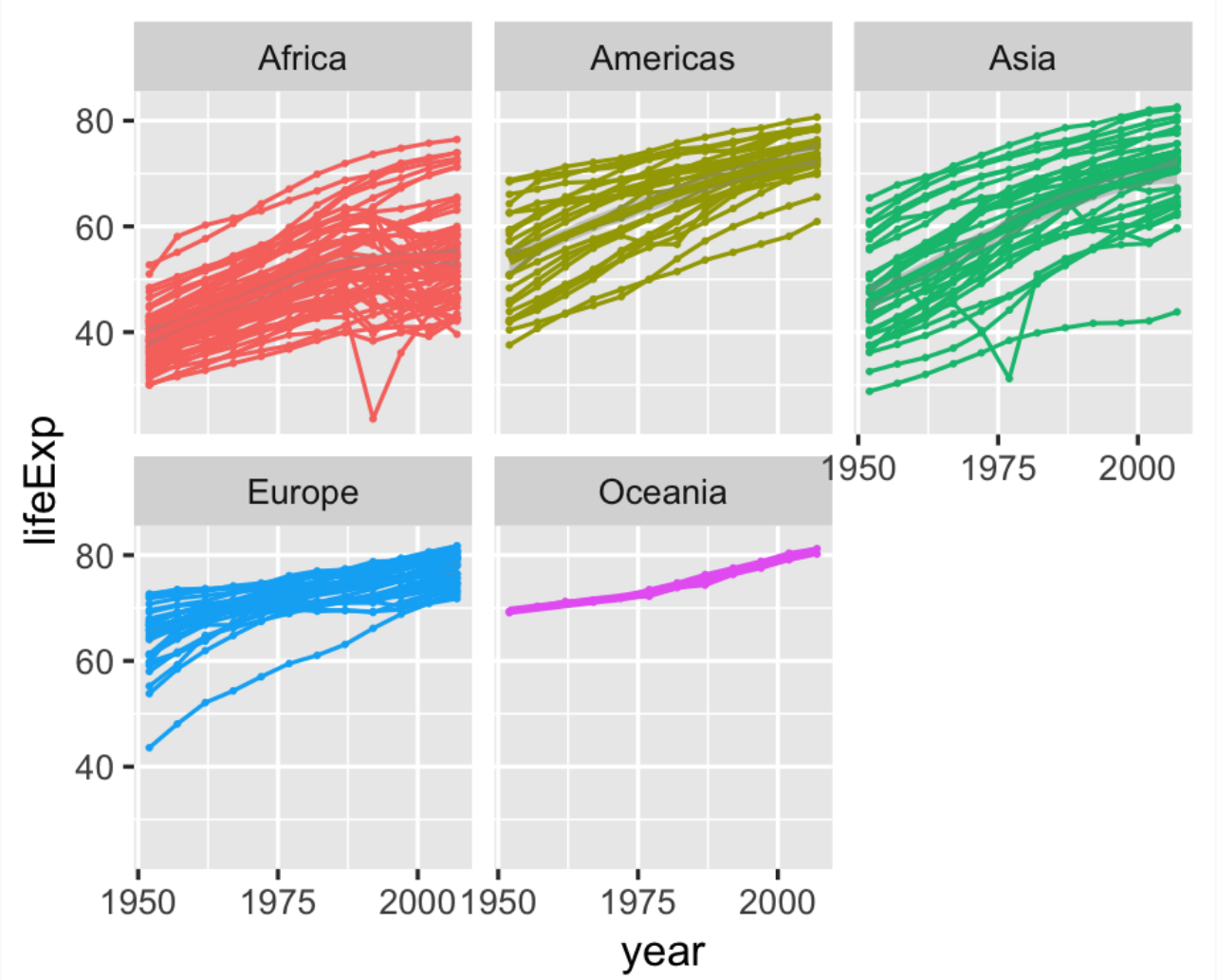


```
ggplot(gapminder) +
  aes(x = year,
      y = lifeExp,
      color = continent) +
  geom_line(
    aes(group = country)
  ) +
  geom_point(size = 0.25) +
  #scale_y_log10() +
  scale_x_continuous(breaks =
    seq(1950, 2000, 25)
  ) +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```

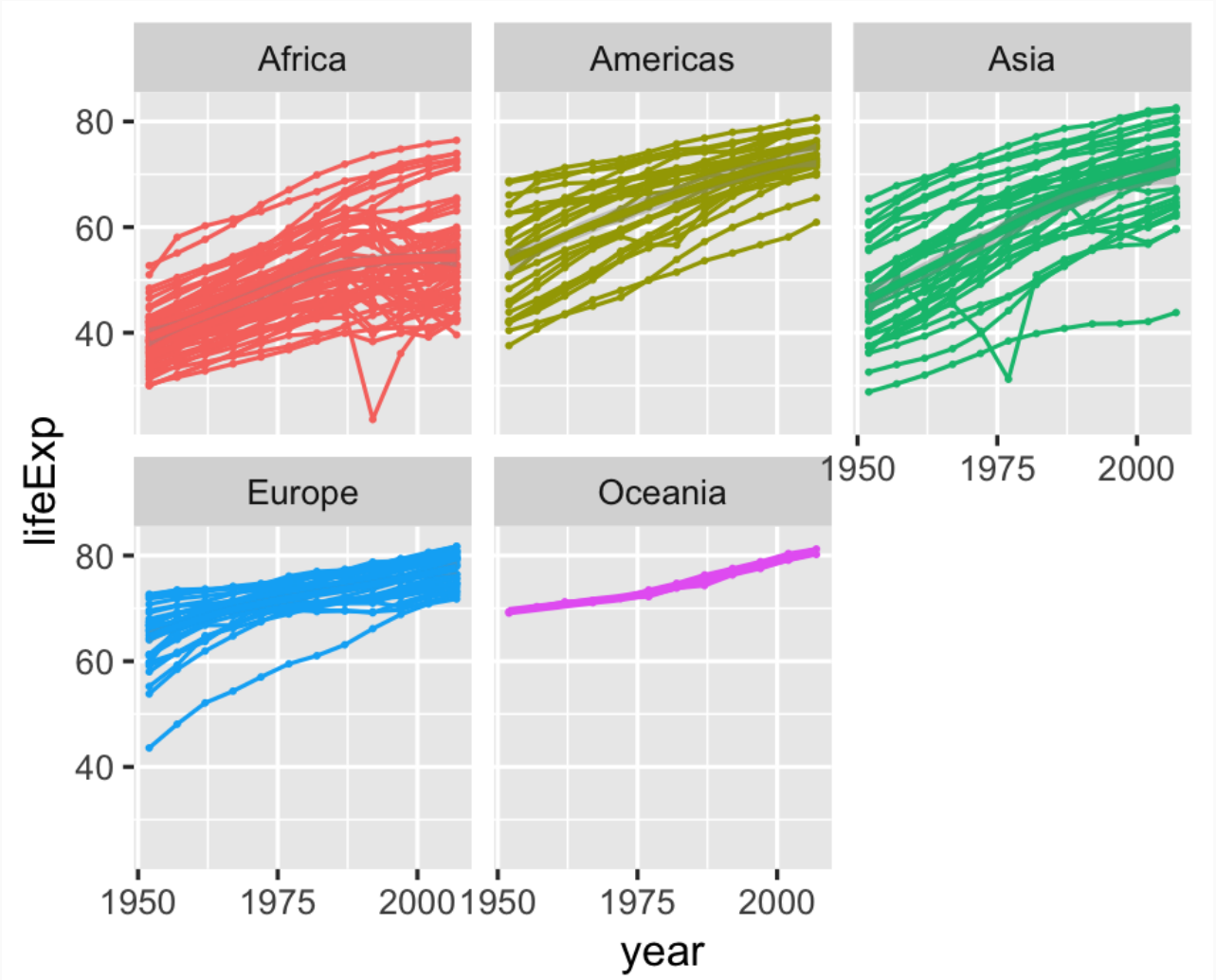


Okay, let's add a trend line

```
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)
```

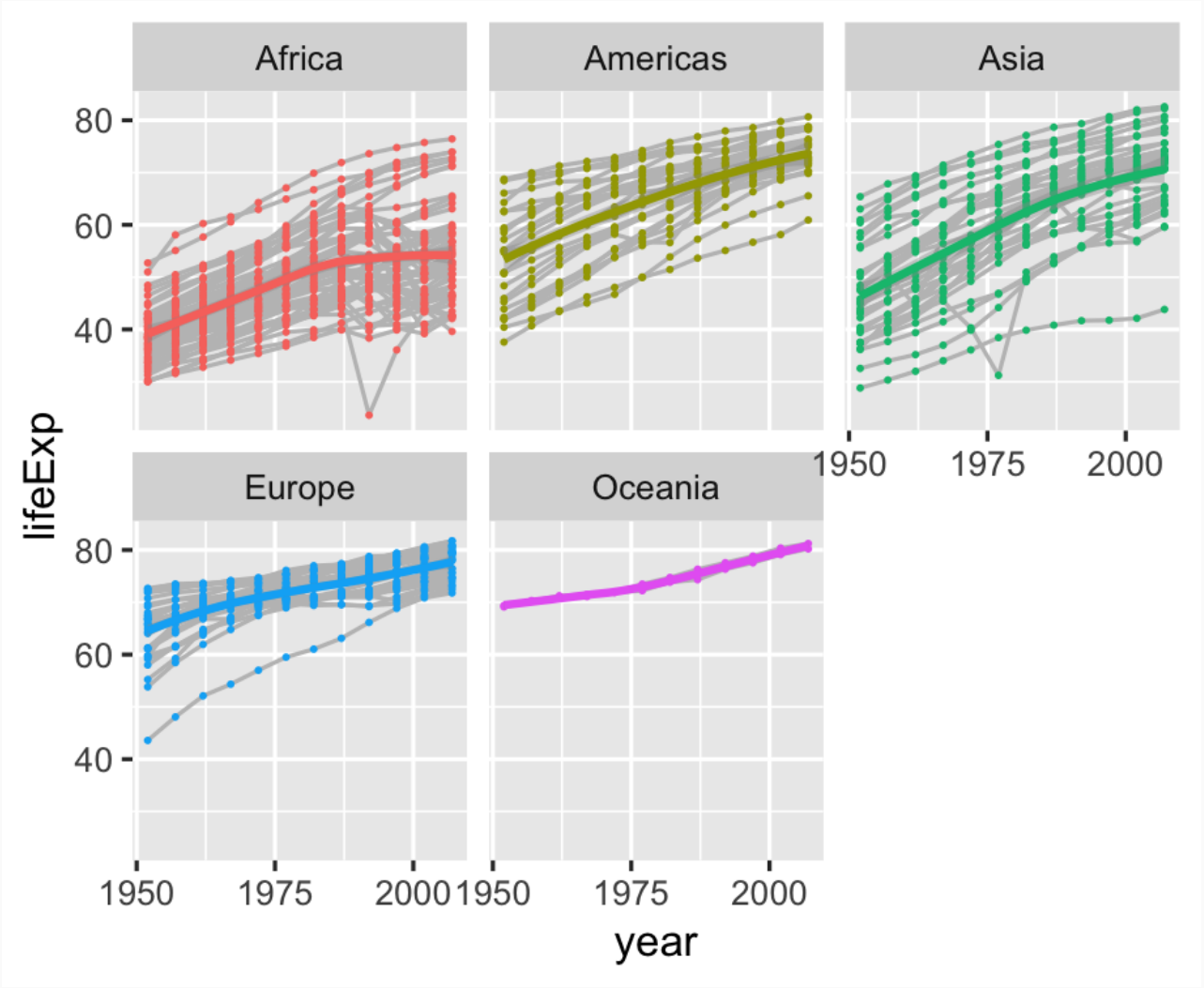


```
ggplot(gapminder) +
  aes(x = year,
      y = lifeExp,
      color = continent) +
  geom_line(
    aes(group = country)
  ) +
  geom_point(size = 0.25) +
  geom_smooth() +
  scale_x_continuous(breaks =
    seq(1950, 2000, 25)
  ) +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```

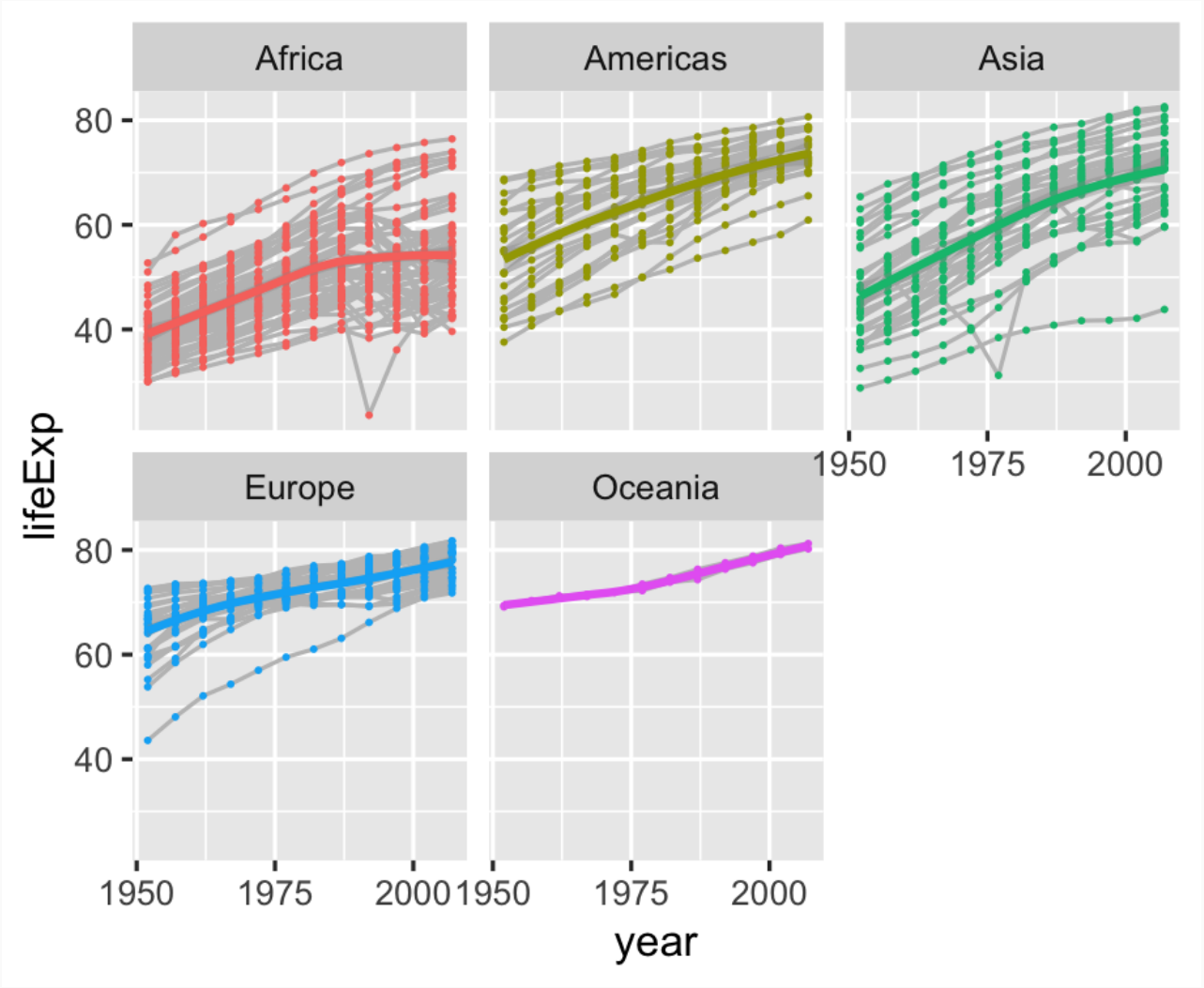


De-emphasize individual countries

```
ggplot(gapminder) +
  aes(x = year,
      y = lifeExp,
      color = continent) +
  geom_line(
    aes(group = country),
    color = "grey75"
  ) +
  geom_point(size = 0.25) +
  geom_smooth() +
  scale_x_continuous(breaks =
    seq(1950, 2000, 25)
  ) +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```

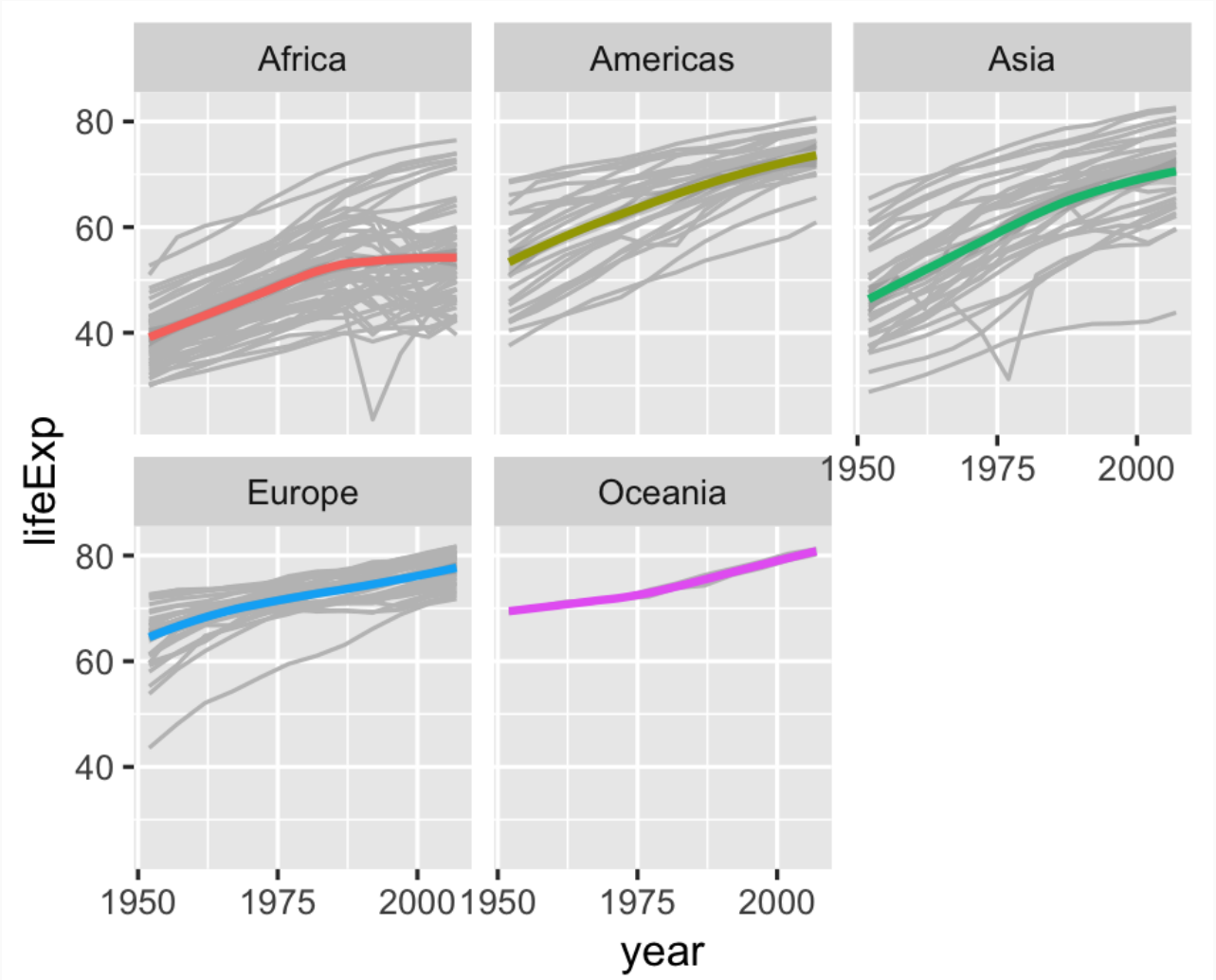


```
ggplot(gapminder) +
  aes(x = year,
      y = lifeExp,
      color = continent) +
  geom_line(
    aes(group = country),
    color = "grey75"
  ) +
  geom_point(size = 0.25) +
  geom_smooth() +
  scale_x_continuous(breaks =
    seq(1950, 2000, 25)
  ) +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```

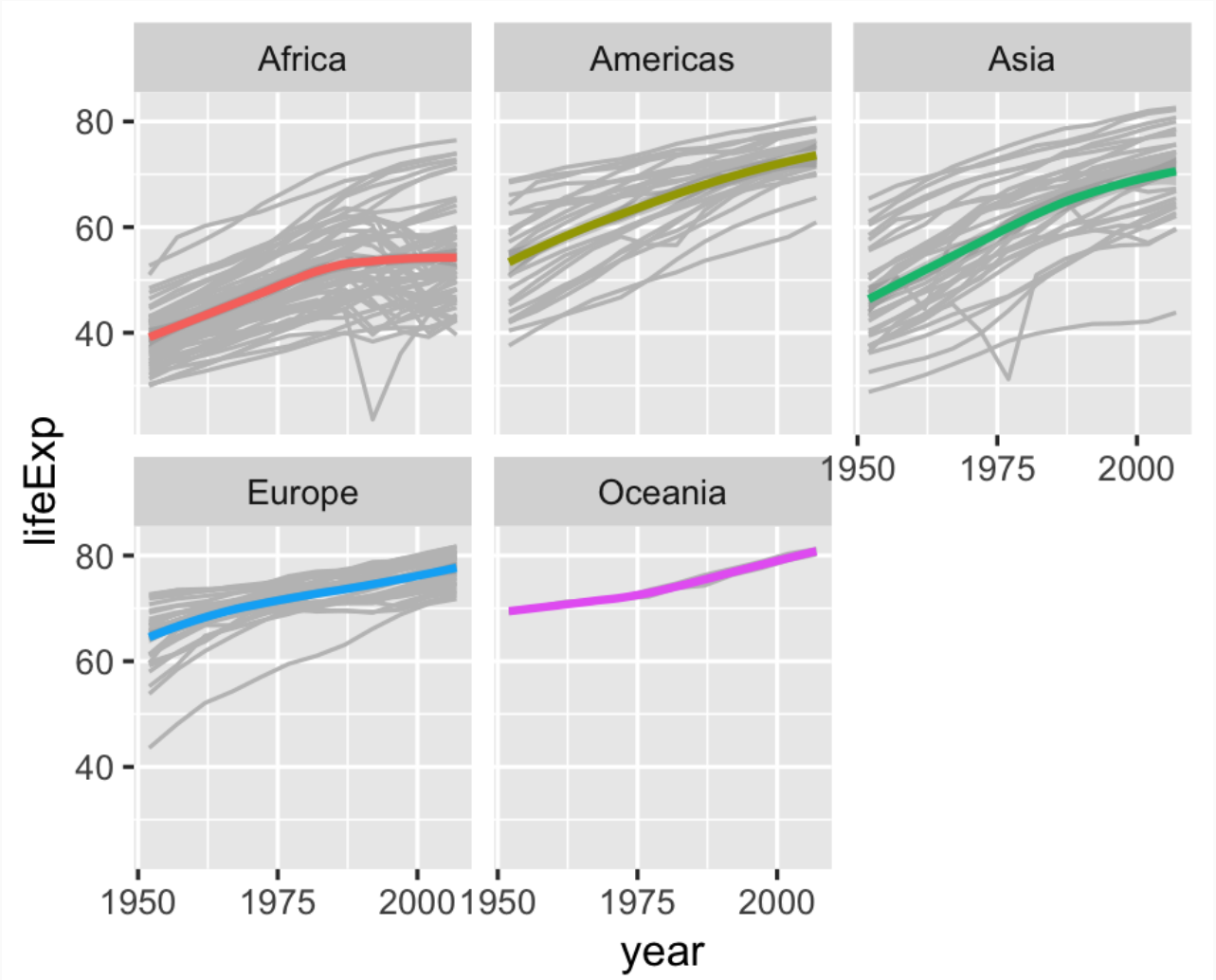


Points are still in the way

```
ggplot(gapminder) +
  aes(x = year,
      y = lifeExp,
      color = continent) +
  geom_line(
    aes(group = country),
    color = "grey75"
  ) +
  #geom_point(size = 0.25) +
  geom_smooth() +
  scale_x_continuous(breaks =
    seq(1950, 2000, 25)
  ) +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```

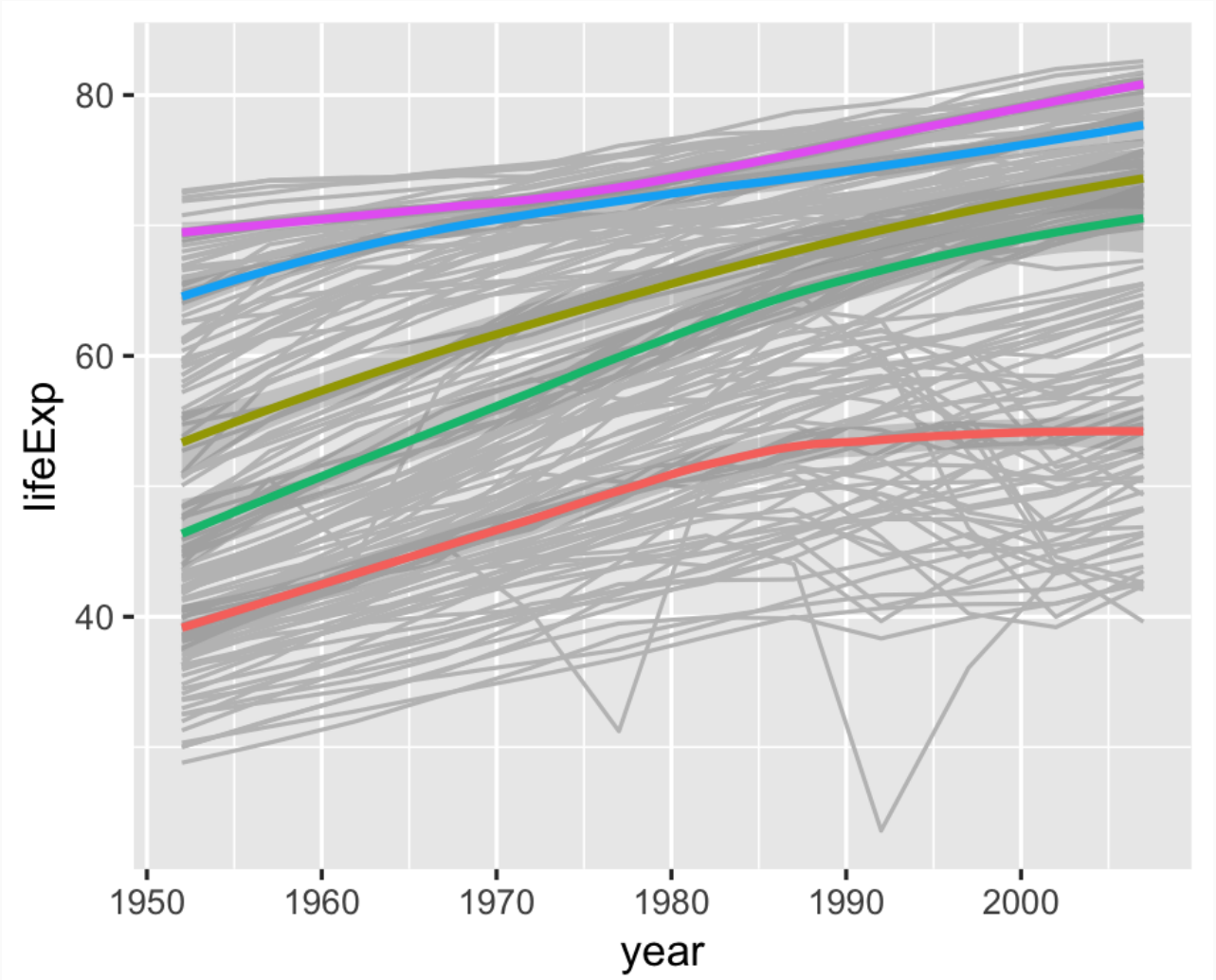


```
ggplot(gapminder) +
  aes(x = year,
      y = lifeExp,
      color = continent) +
  geom_line(
    aes(group = country),
    color = "grey75"
  ) +
  #geom_point(size = 0.25) +
  geom_smooth() +
  scale_x_continuous(breaks =
    seq(1950, 2000, 25)
  ) +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```

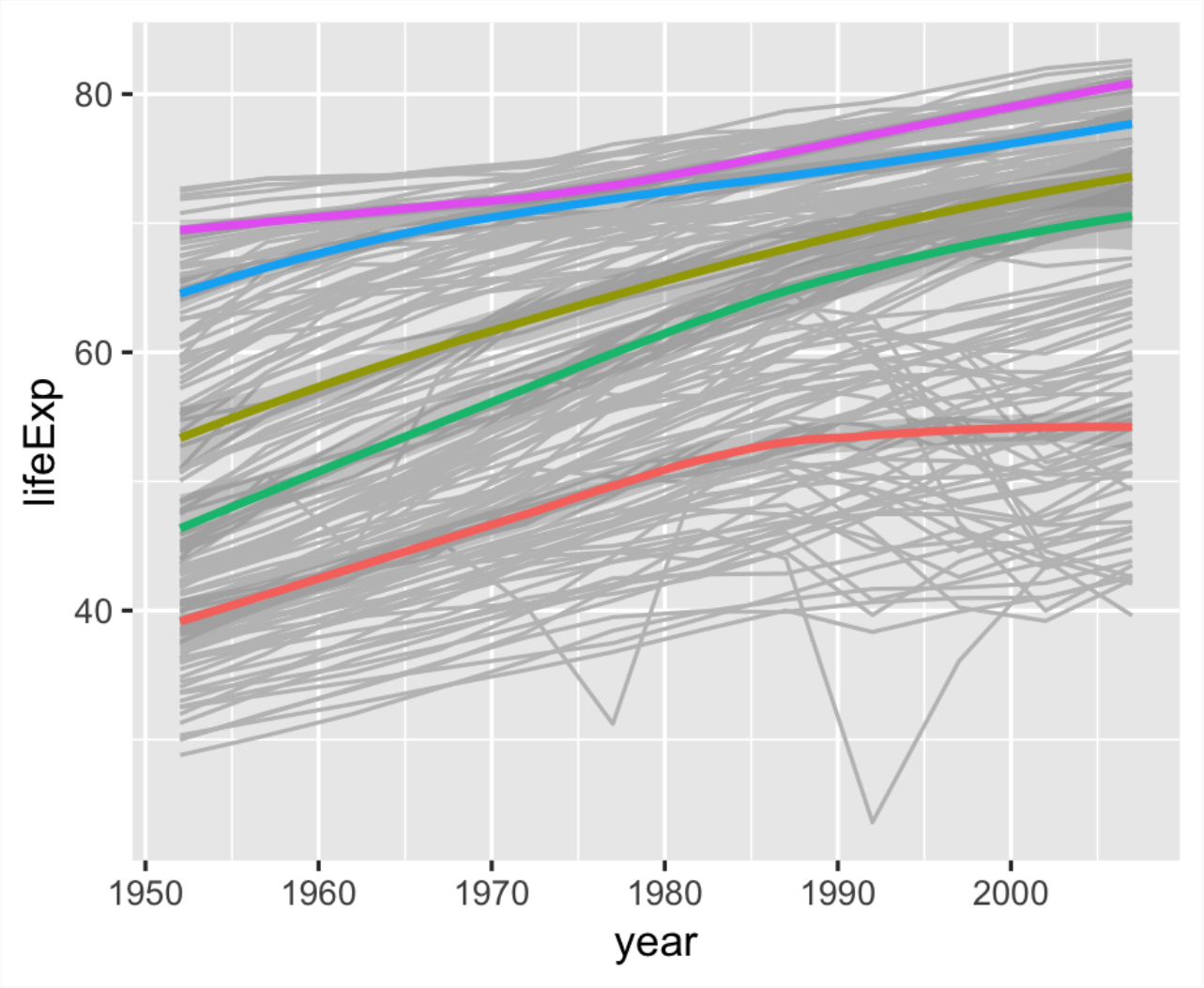


Let's compare continents

```
ggplot(gapminder) +
  aes(x = year,
      y = lifeExp,
      color = continent) +
  geom_line(
    aes(group = country),
    color = "grey75"
  ) +
  geom_smooth() +
  # scale_x_continuous(
  #   breaks =
  #     seq(1950, 2000, 25)
  # ) +
  # facet_wrap(~ continent) +
  guides(color = FALSE)
```

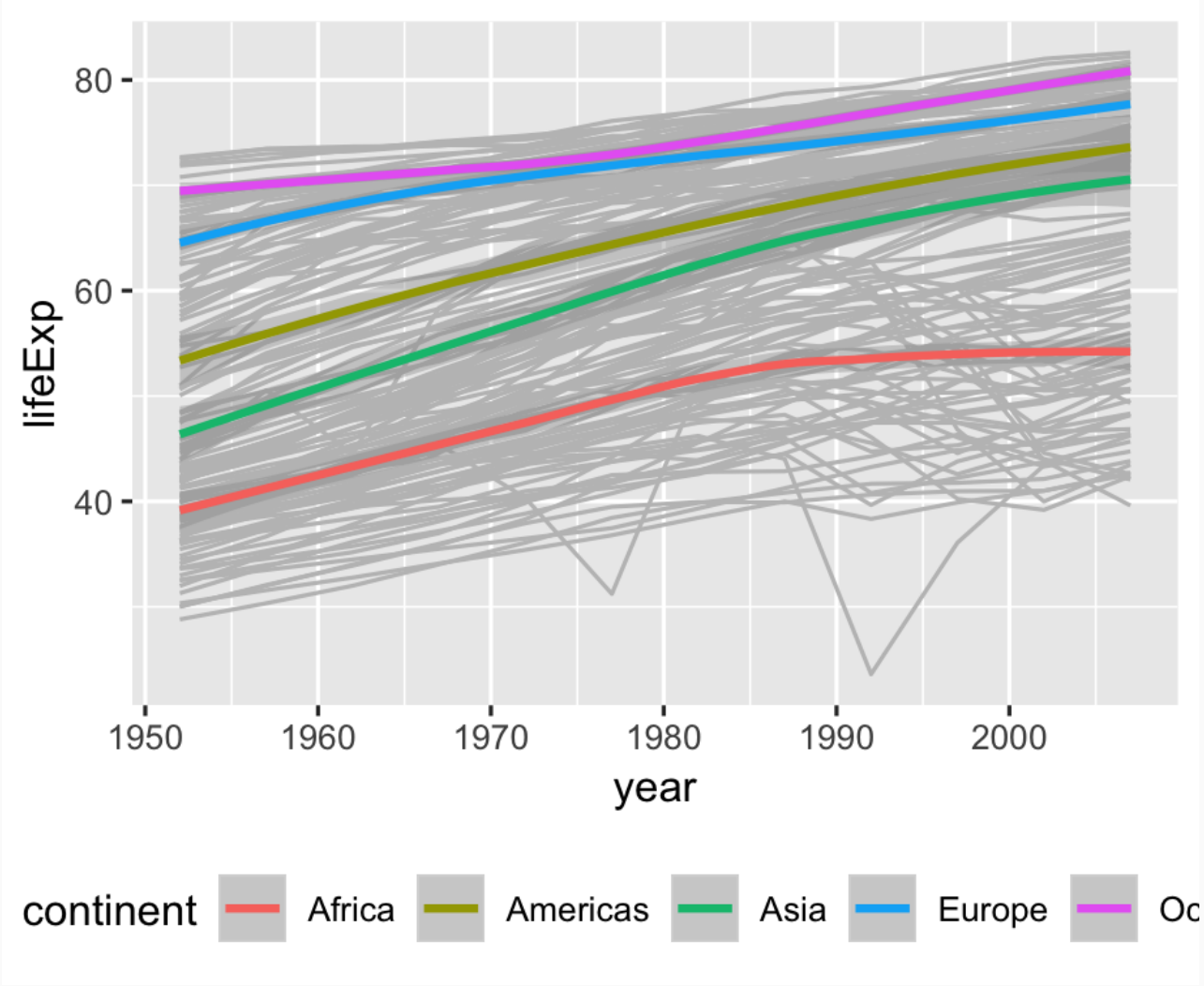


```
ggplot(gapminder) +
  aes(x = year,
      y = lifeExp,
      color = continent) +
  geom_line(
    aes(group = country),
    color = "grey75"
  ) +
  geom_smooth() +
  # scale_x_continuous(
  #   breaks =
  #     seq(1950, 2000, 25)
  # ) +
  # facet_wrap(~ continent) +
  guides(color = FALSE)
```

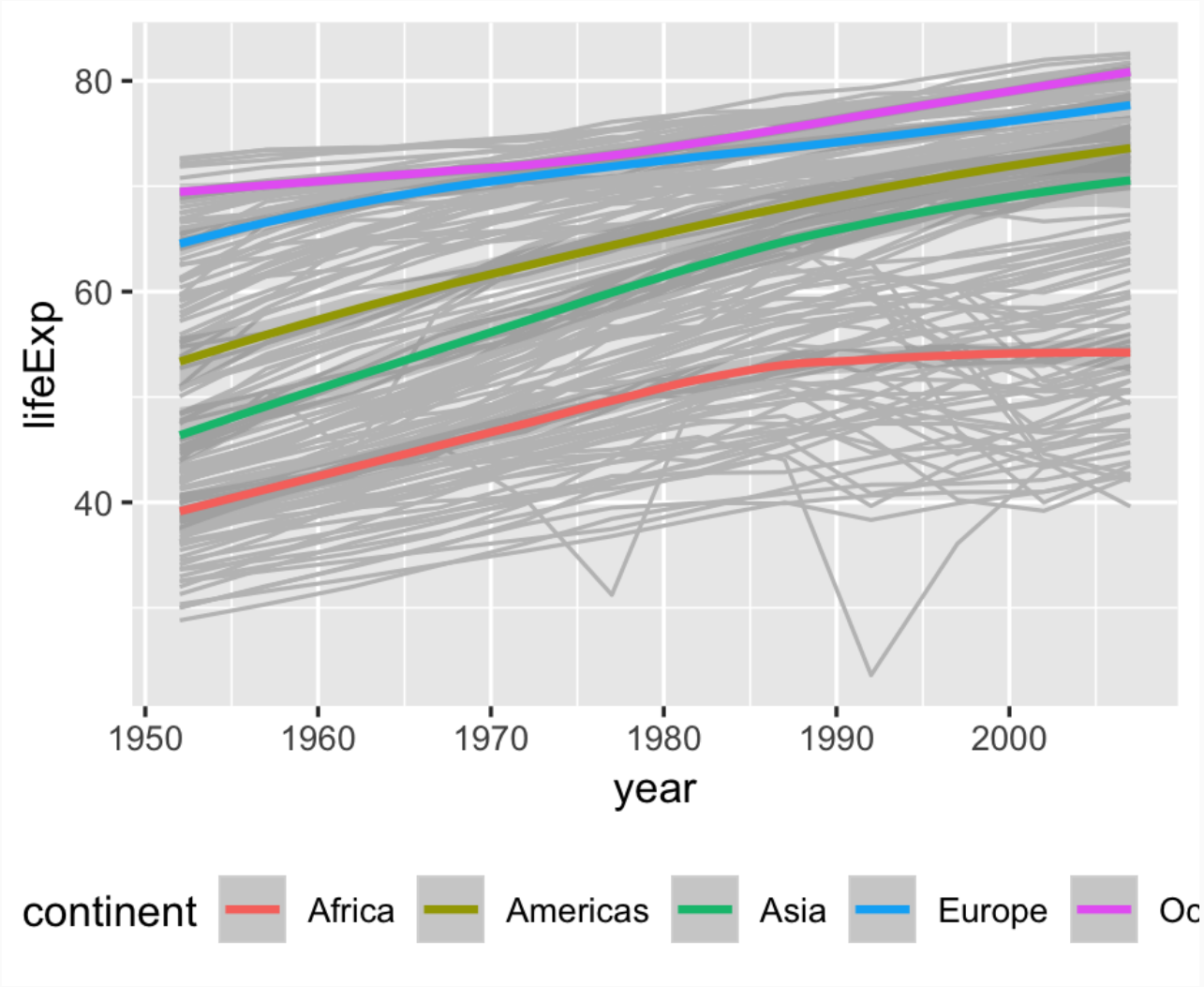


Wait, what color is each continent?

```
ggplot(gapminder) +
  aes(x = year,
      y = lifeExp,
      color = continent) +
  geom_line(
    aes(group = country),
    color = "grey75"
  ) +
  geom_smooth() +
  theme(
  legend.position = "bottom"
  )
```

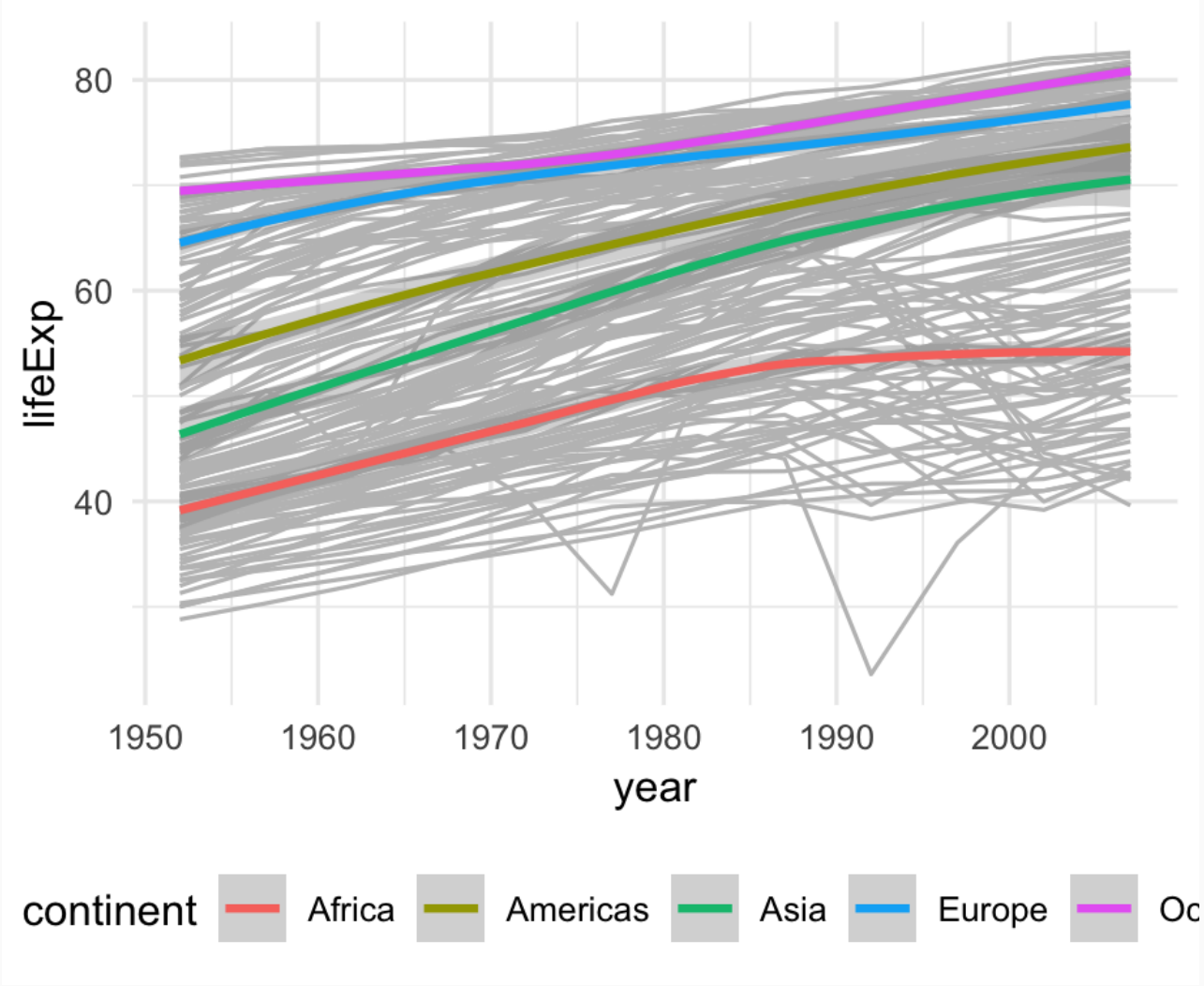


```
ggplot(gapminder) +
  aes(x = year,
      y = lifeExp,
      color = continent) +
  geom_line(
    aes(group = country),
    color = "grey75"
  ) +
  geom_smooth() +
  theme(
  legend.position = "bottom"
  )
```

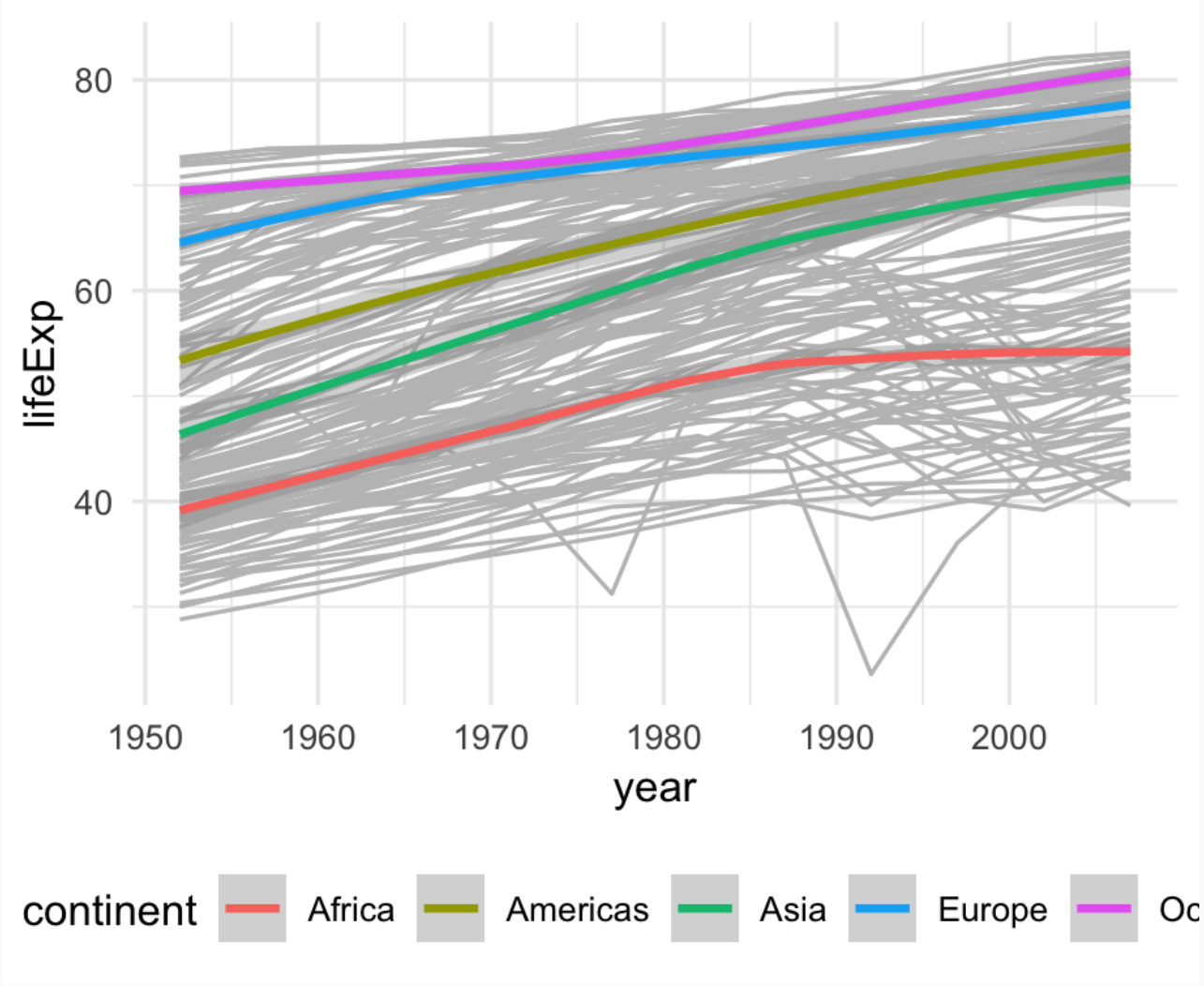


Let's try the minimal theme

```
ggplot(gapminder) +
  aes(x = year,
      y = lifeExp,
      color = continent) +
  geom_line(
    aes(group = country),
    color = "grey75"
  ) +
  geom_smooth() +
  theme_minimal() +
  theme(
  legend.position = "bottom"
  )
```

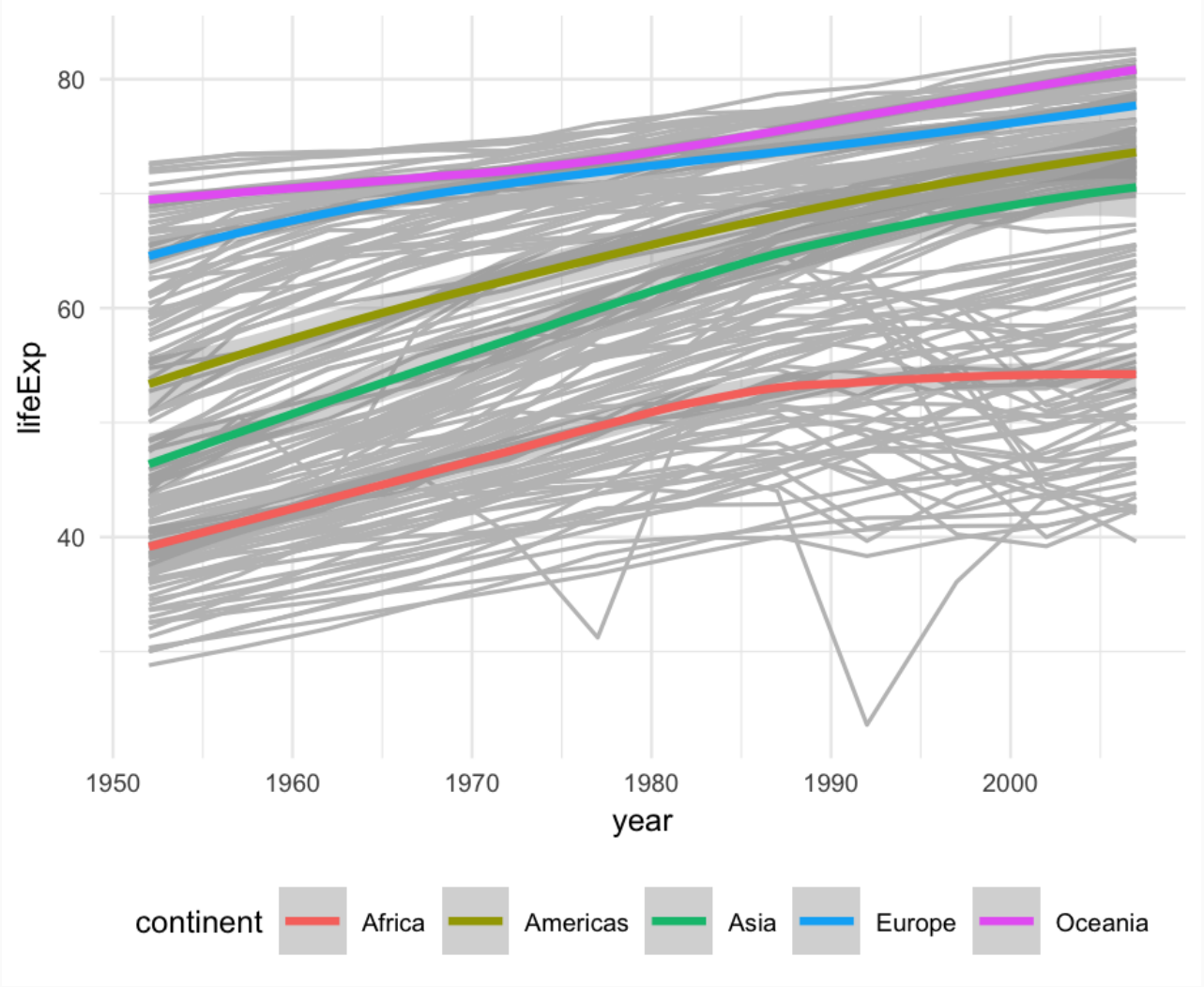


```
ggplot(gapminder) +
  aes(x = year,
      y = lifeExp,
      color = continent) +
  geom_line(
    aes(group = country),
    color = "grey75"
  ) +
  geom_smooth() +
  theme_minimal() +
  theme(
  legend.position = "bottom"
  )
```

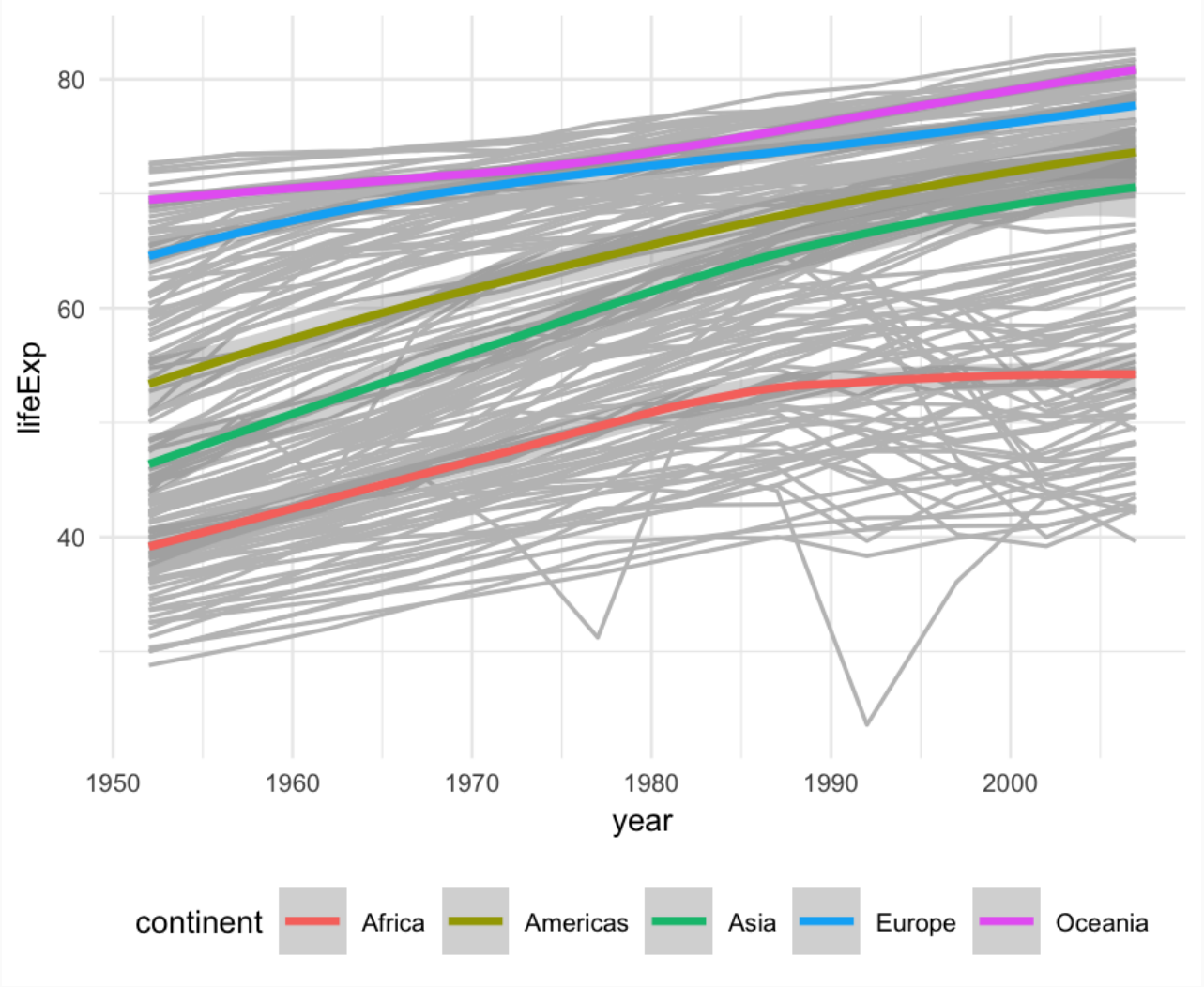


Fonts are kind of big

```
ggplot(gapminder) +
  aes(x = year,
      y = lifeExp,
      color = continent) +
  geom_line(
    aes(group = country),
    color = "grey75"
  ) +
  geom_smooth() +
  theme_minimal(
    base_size = 8) +
  theme(
  legend.position = "bottom"
  )
```



```
ggplot(gapminder) +
  aes(x = year,
      y = lifeExp,
      color = continent) +
  geom_line(
    aes(group = country),
    color = "grey75"
  ) +
  geom_smooth() +
  theme_minimal(
    base_size = 8) +
  theme(
  legend.position = "bottom"
  )
```



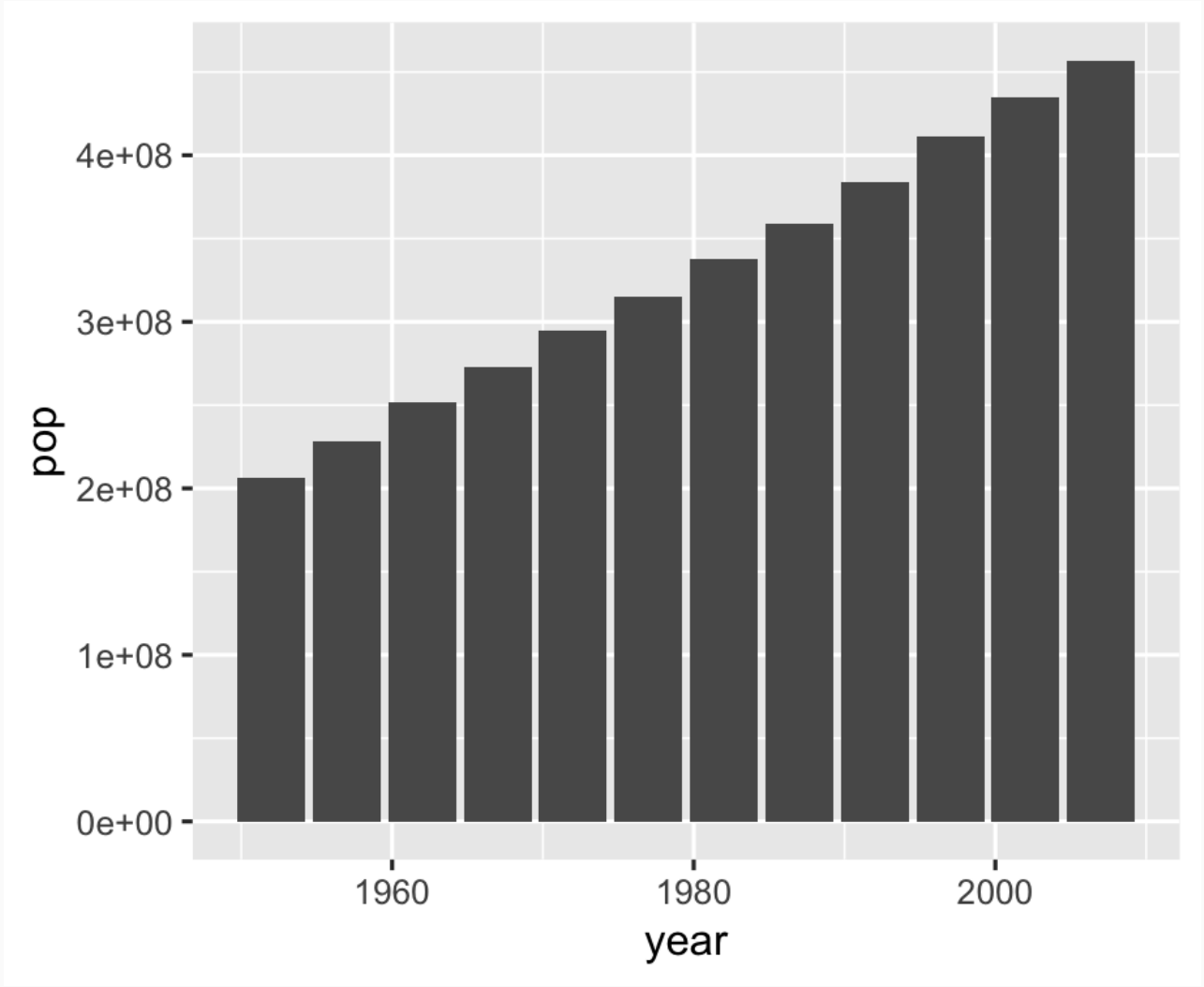
Cool, let's switch gears

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

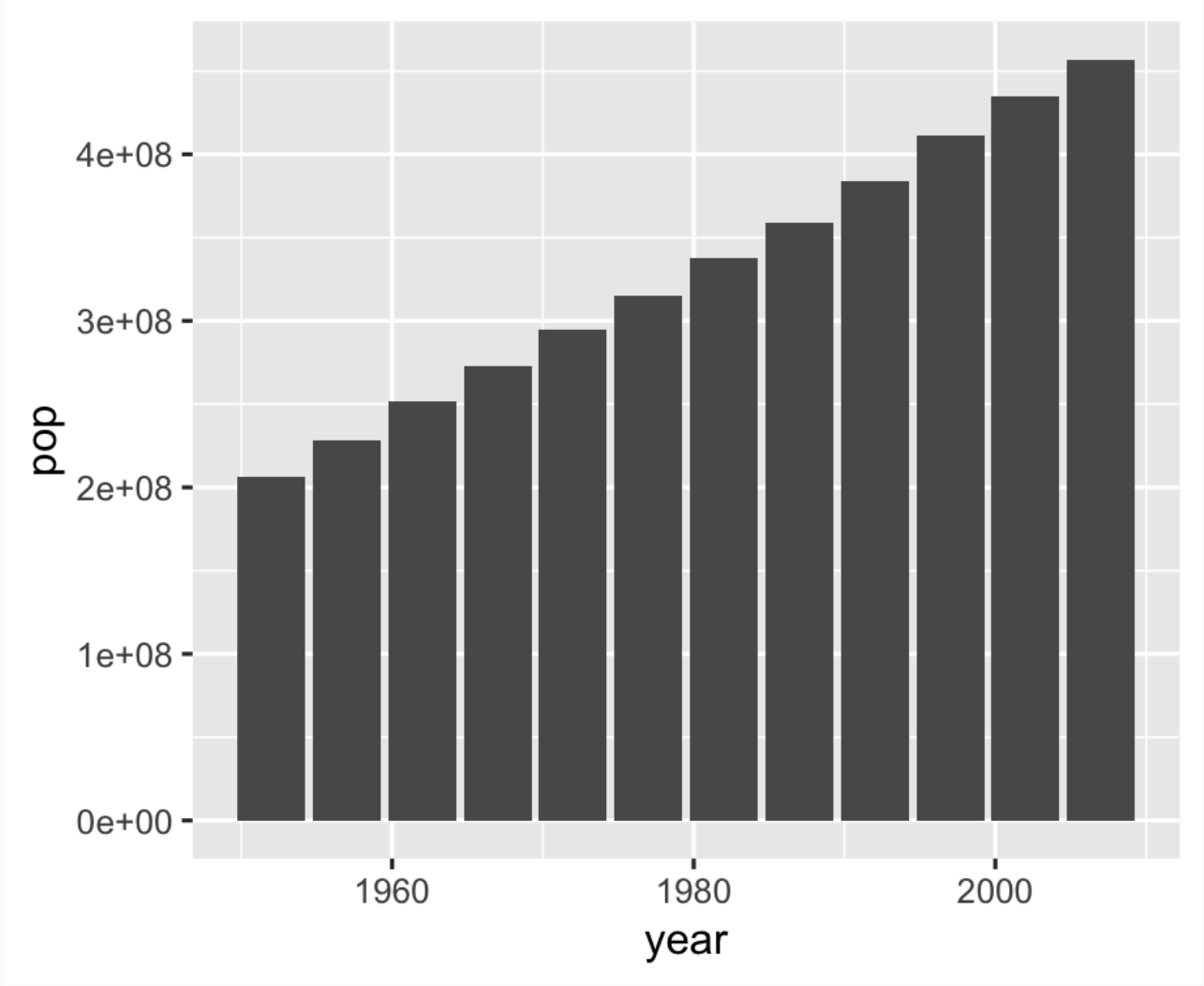
Let's look at four countries in more detail. How do their populations compare to each other?

country	continent	year	lifeExp	pop	gdpPercap
Canada	Americas	1952	68.750	14785584	11367.161
Canada	Americas	1957	69.960	17010154	12489.950
Canada	Americas	1962	71.300	18985849	13462.486
Canada	Americas	1967	72.130	20819767	16076.588
Canada	Americas	1972	72.880	22284500	18970.571
Canada	Americas	1977	74.210	23796400	22090.883
Canada	Americas	1982	75.760	25201900	22898.792
Canada	Americas	1987	76.860	26549700	26626.515
Canada	Americas	1992	77.950	28523502	26342.884
Canada	Americas	1997	78.610	30305843	28954.926
Canada	Americas	2002	79.770	31902268	33328.965
Canada	Americas	2007	80.653	33390141	36319.235

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

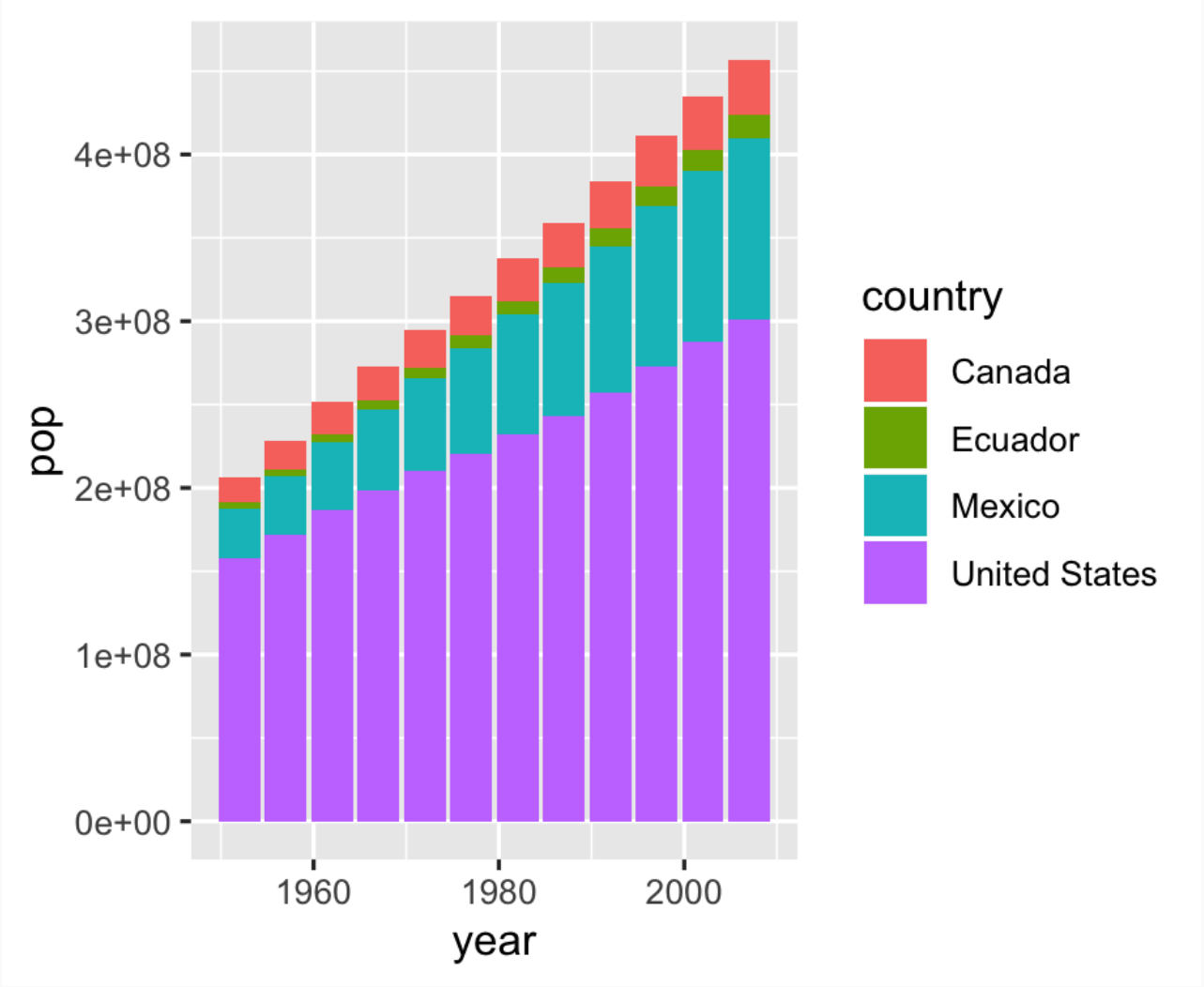


```
ggplot(americas) +
  aes(
    x = year,
    y = pop
  ) +
  geom_col()
```

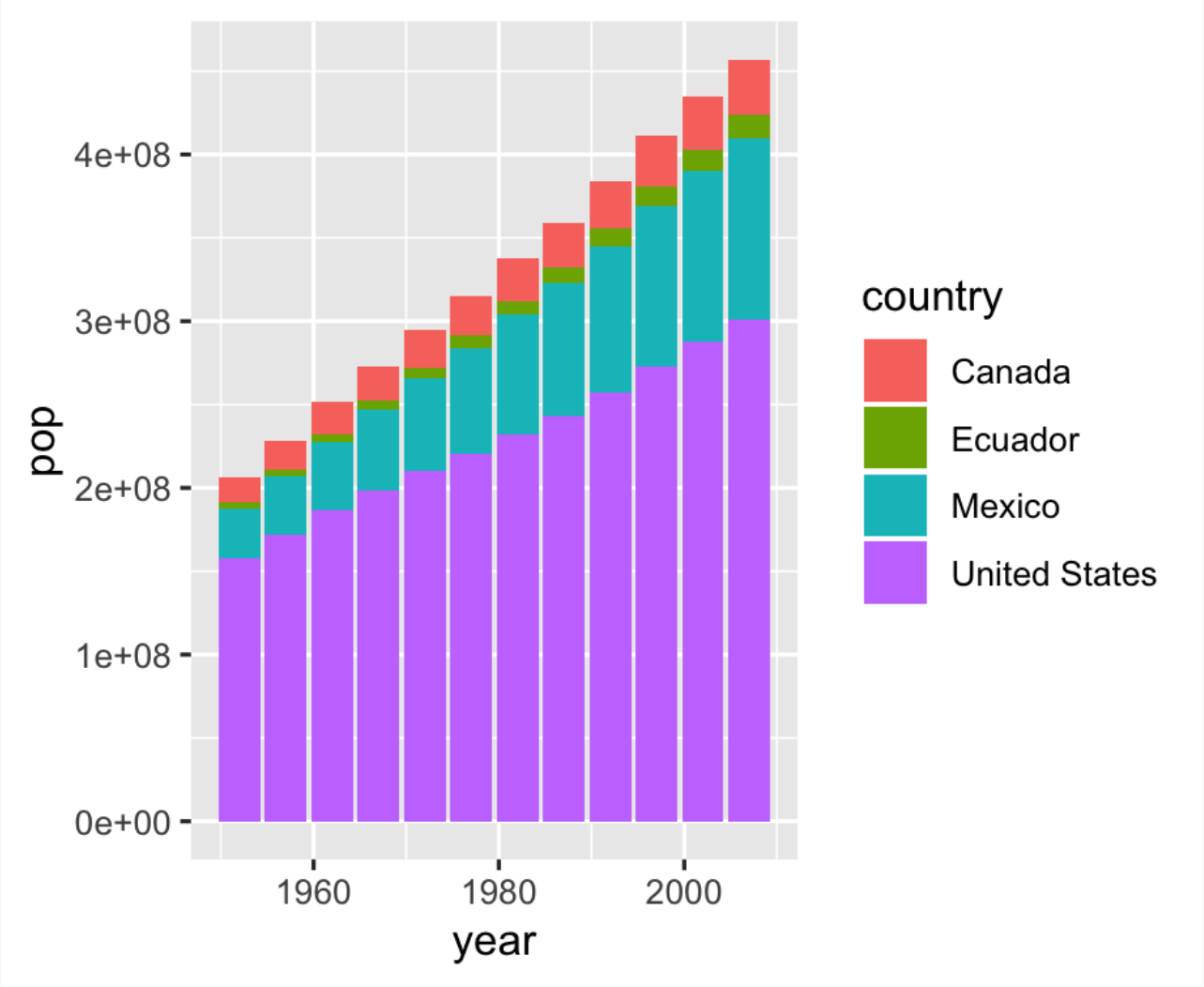


Yeah, but how many people are in each country?

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



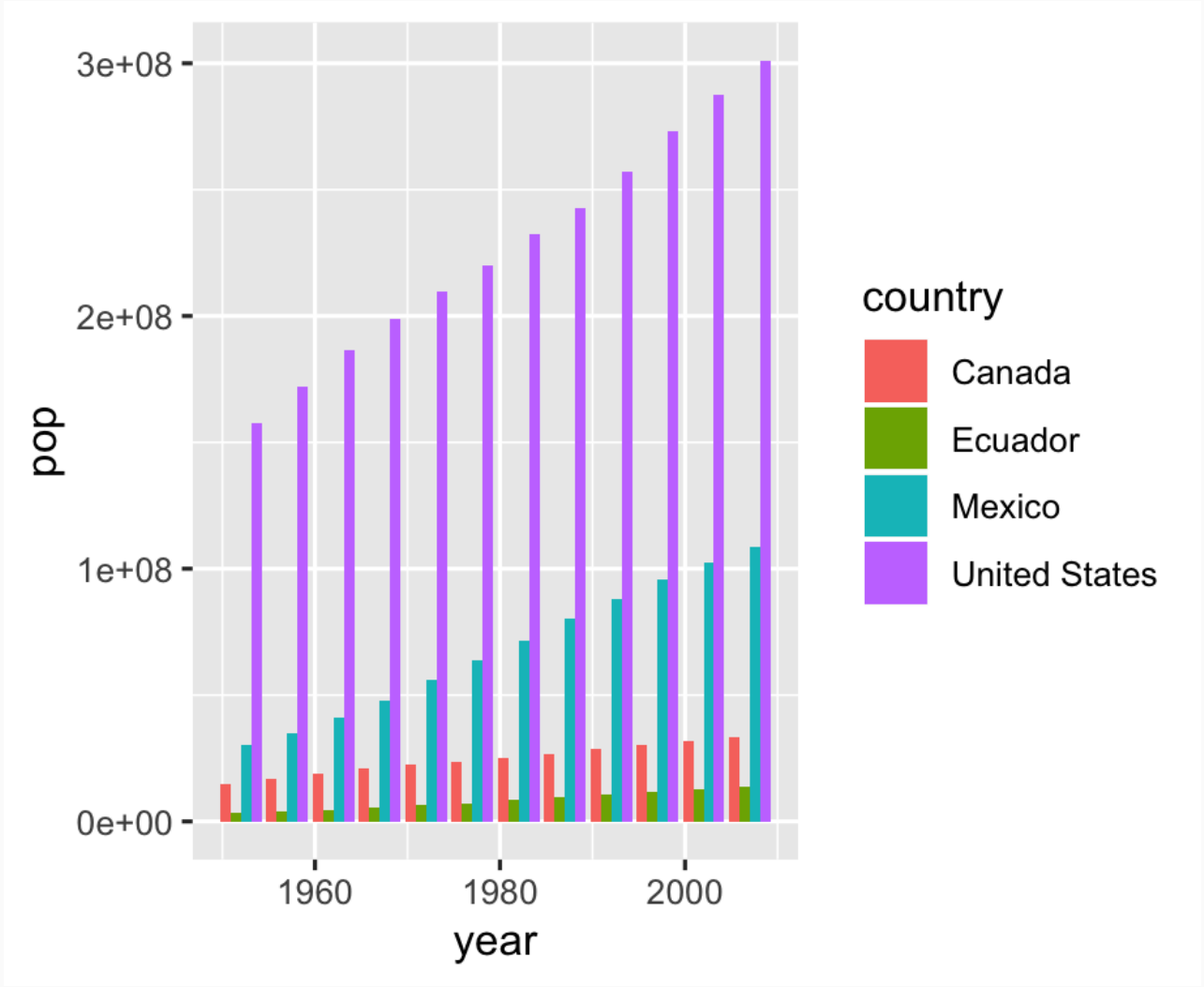
```
ggplot(americas) +
  aes(
    x = year,
    y = pop,
    fill = country
  ) +
  geom_col()
```



Bars are "stacked", can we separate?

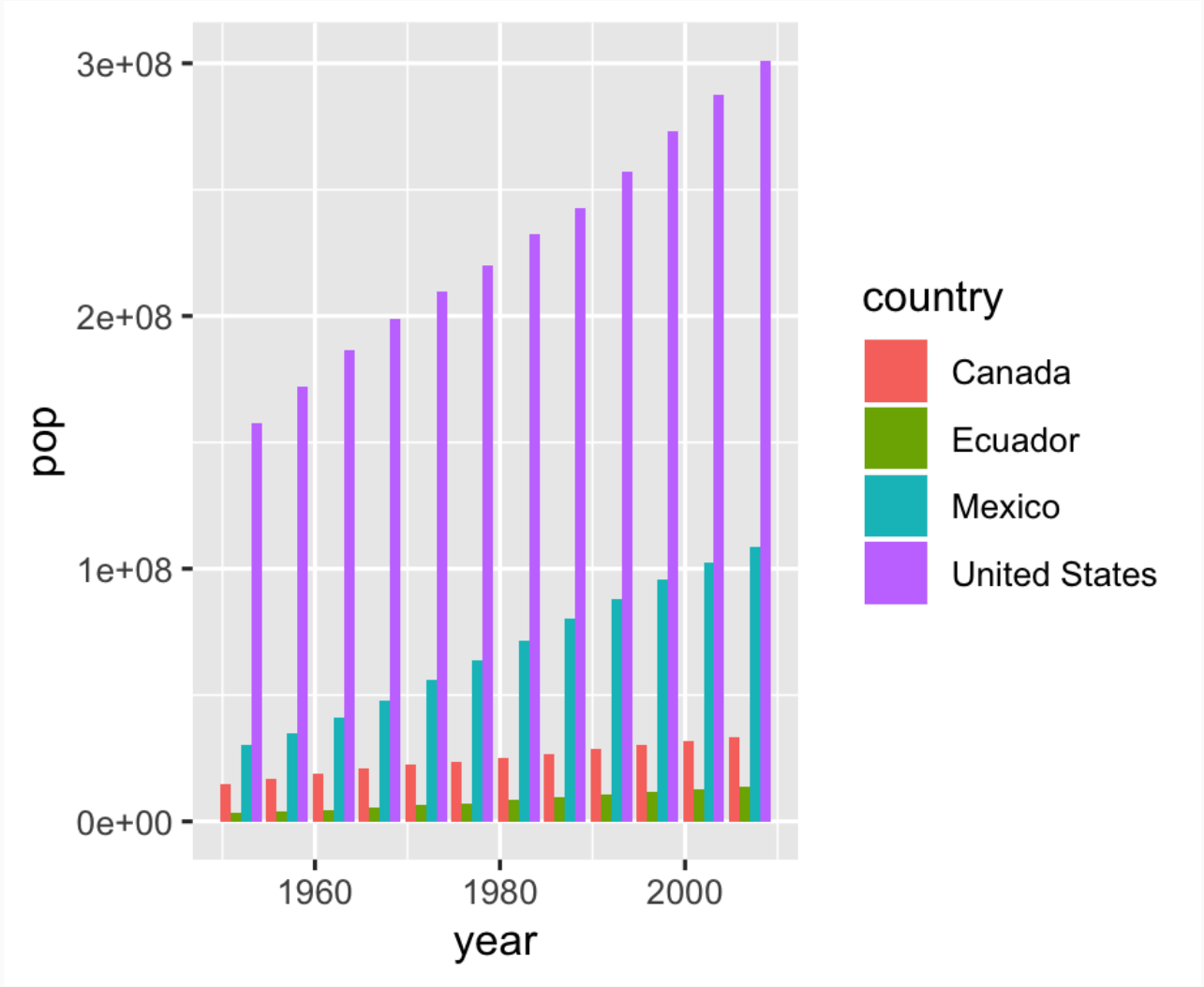
```
ggplot(americas) +
  aes(
    x = year,
    y = pop,
    fill = country
  ) +
  geom_col(
    position = "dodge"
  )
```

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



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

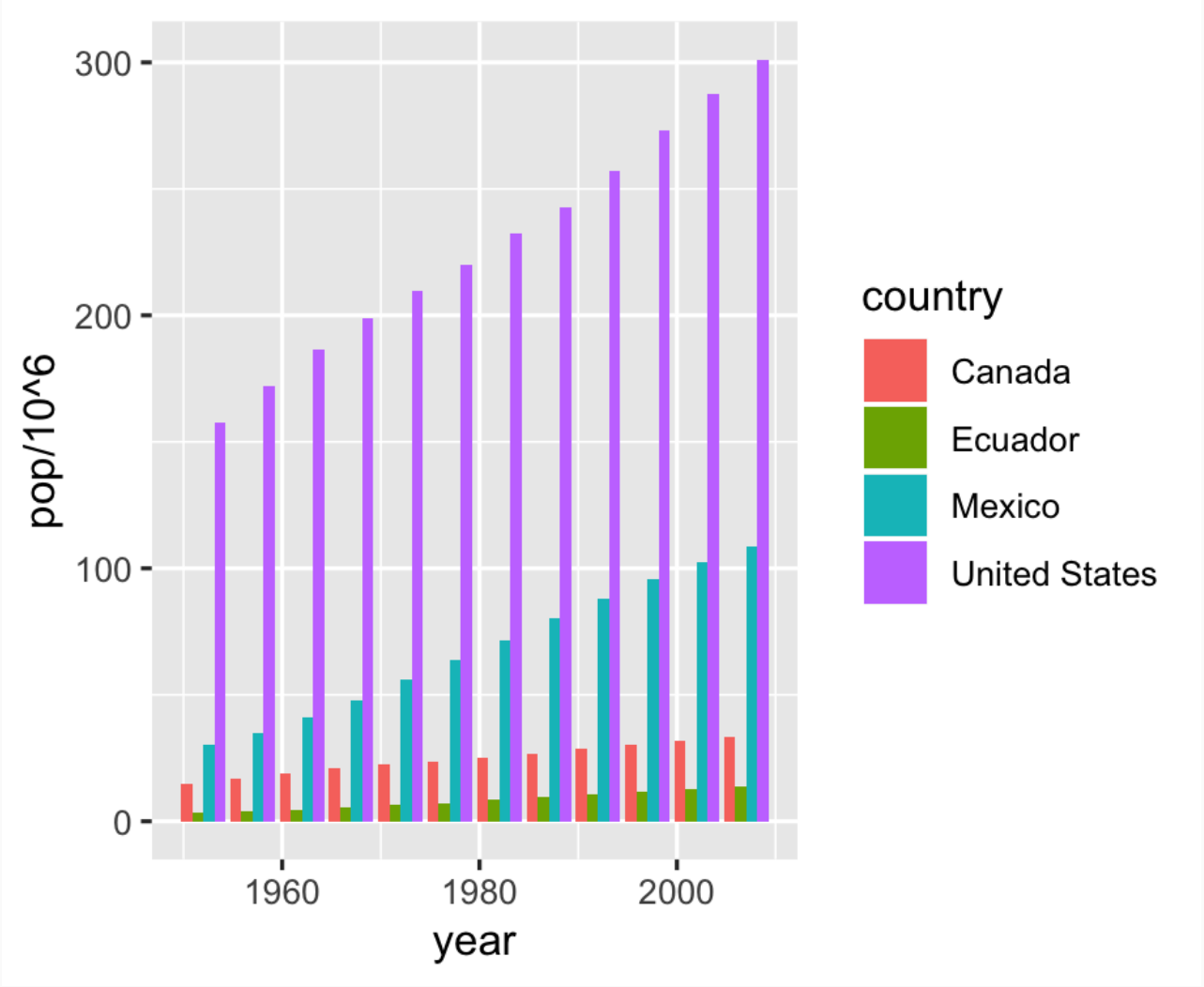
`position = "dodge"` places objects *next to each other* instead of overlapping



What is scientific notation anyway?

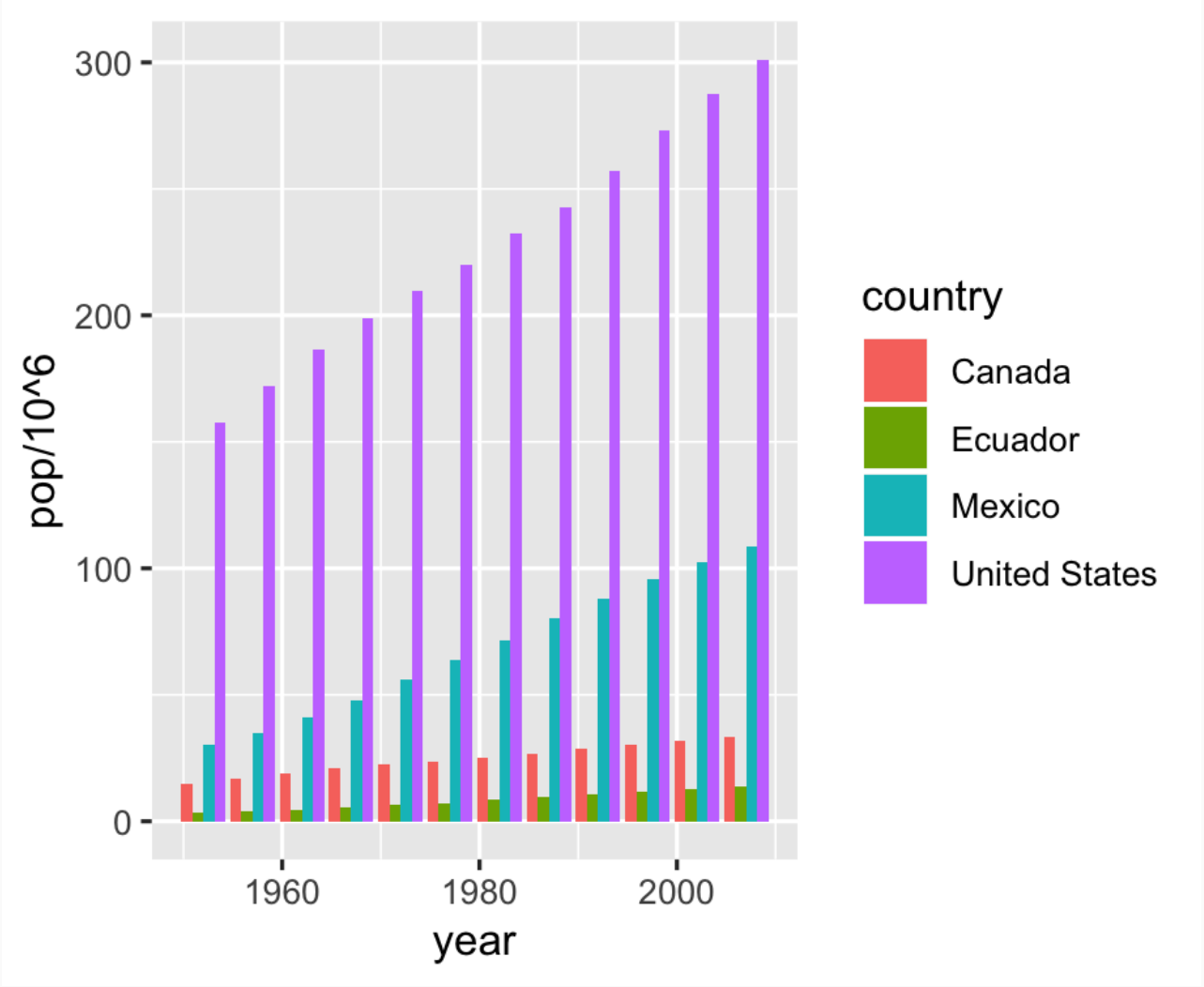
```
ggplot(americas) +
  aes(
    x = year,
    y = pop / 10^6,
    fill = country
  ) +
  geom_col(
    position = "dodge"
  )
```

ggplot aesthetics can take expressions!



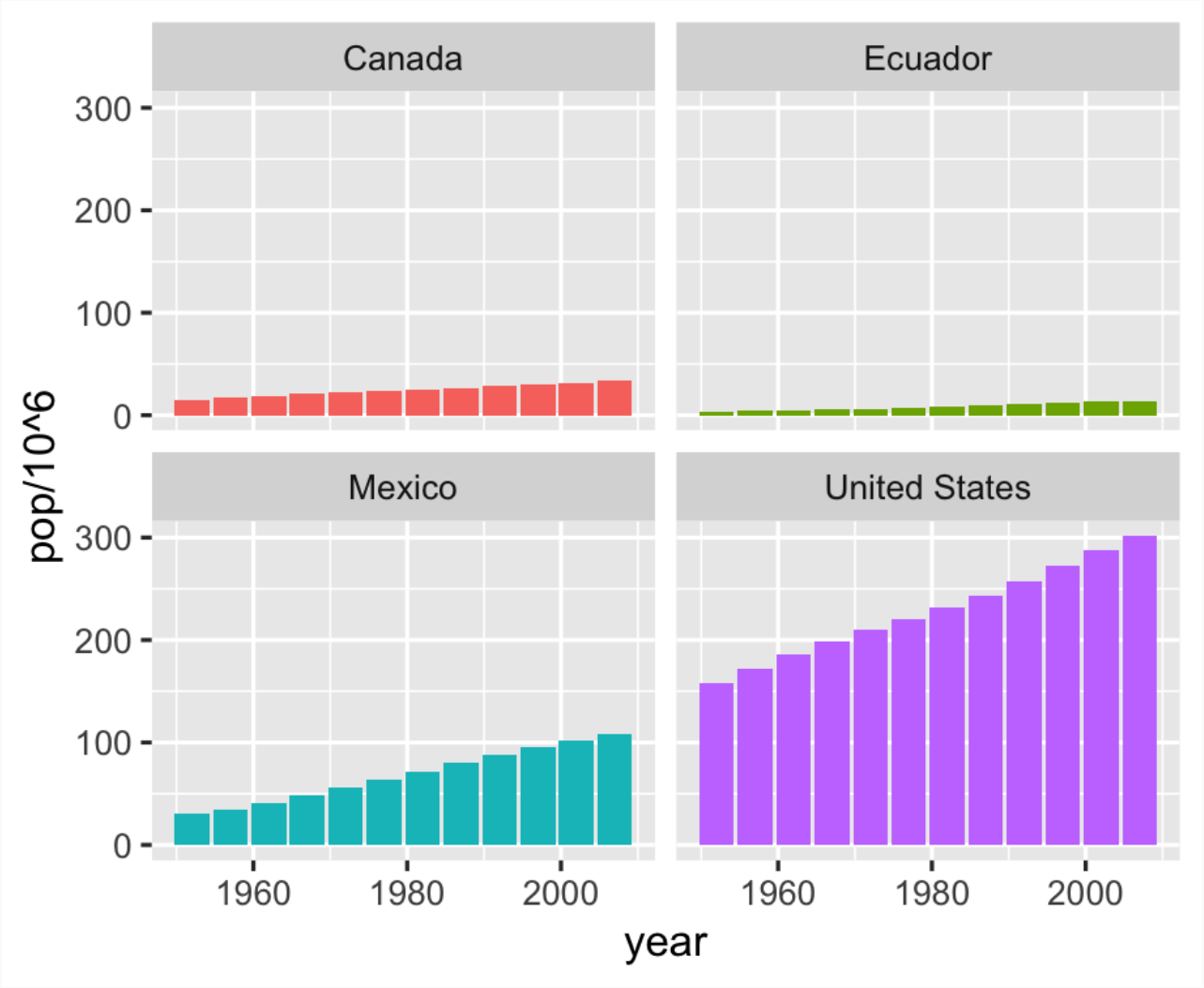
```
ggplot(americas) +
  aes(
    x = year,
    y = pop / 10^6,
    fill = country
  ) +
  geom_col(
    position = "dodge"
  )
```

ggplot aesthetics can take expressions!

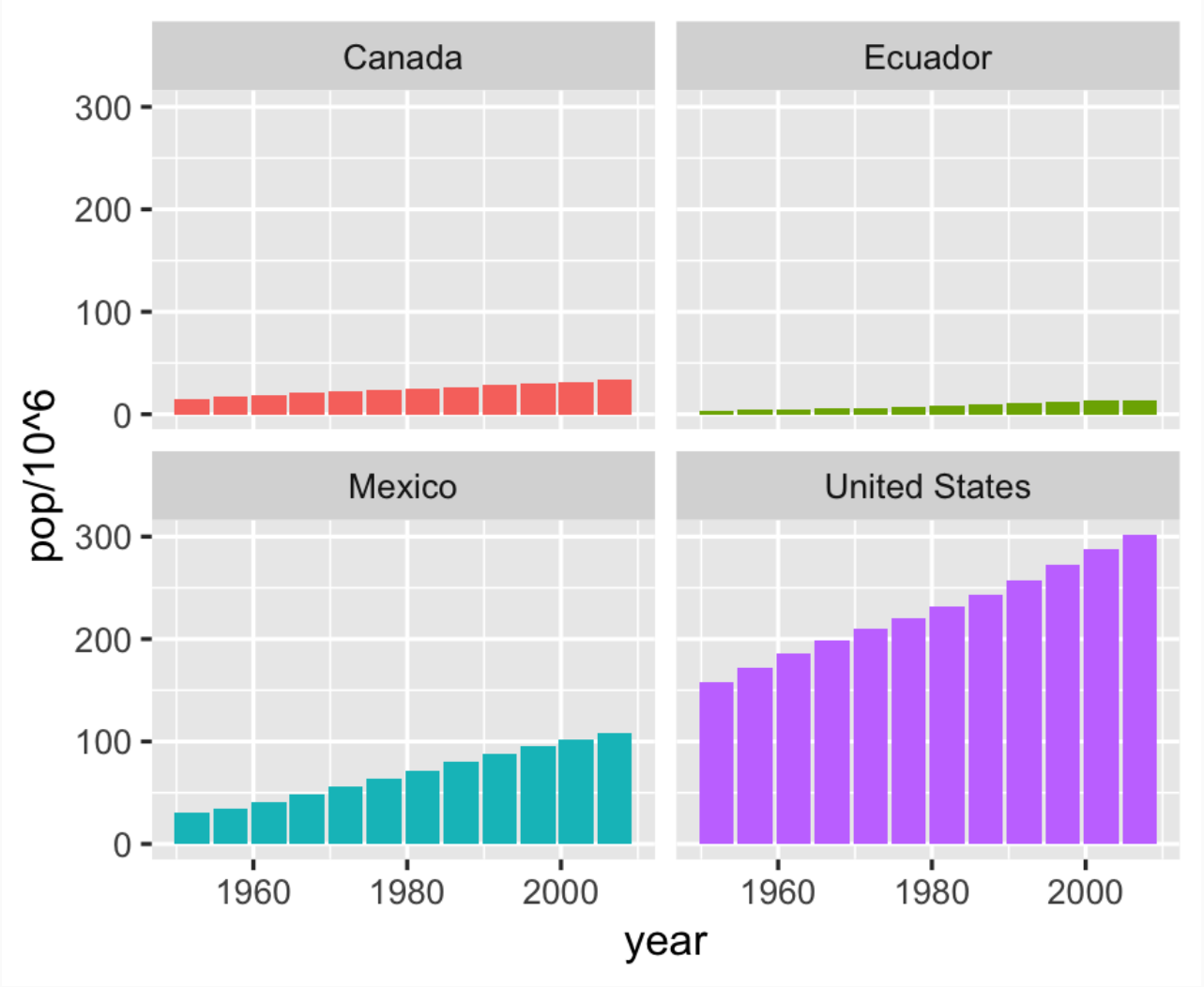


Might be easier to see countries individually

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

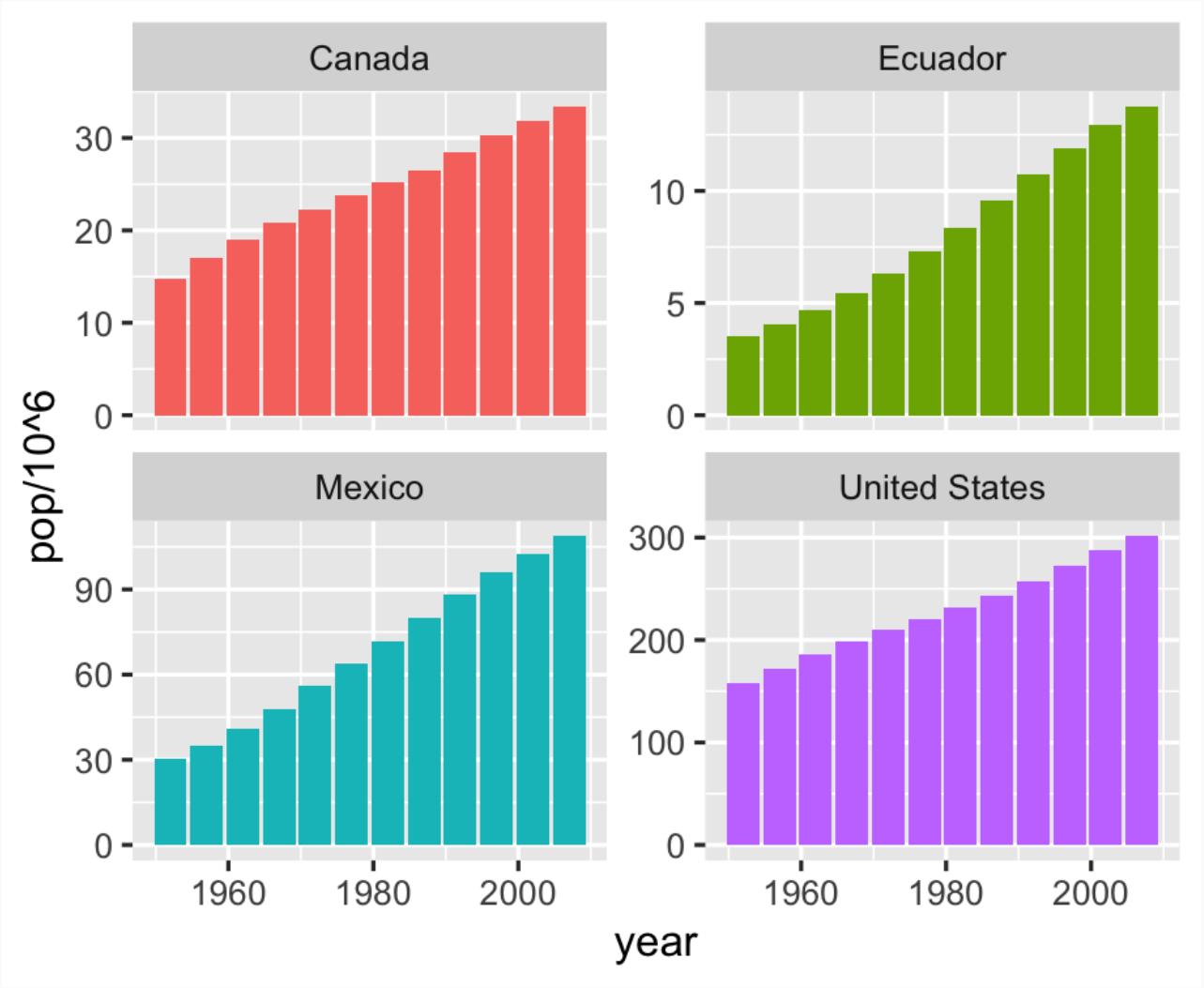


```
ggplot(americas) +
  aes(
    x = year,
    y = pop / 10^6,
    fill = country
  ) +
  geom_col(
    position = "dodge"
  ) +
  facet_wrap(~ country) +
  guides(fill = FALSE)
```

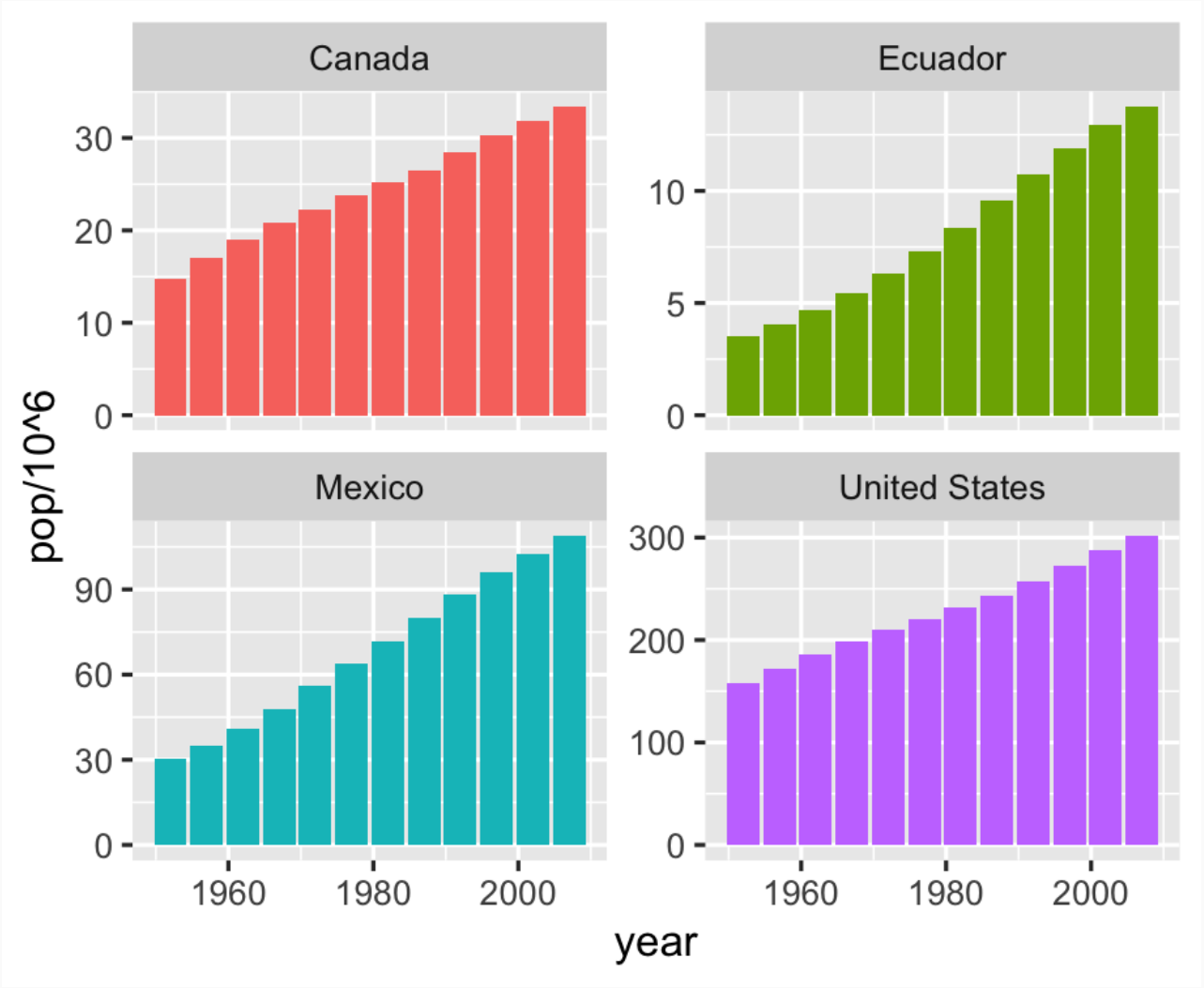


Let range of y-axis vary in each plot

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

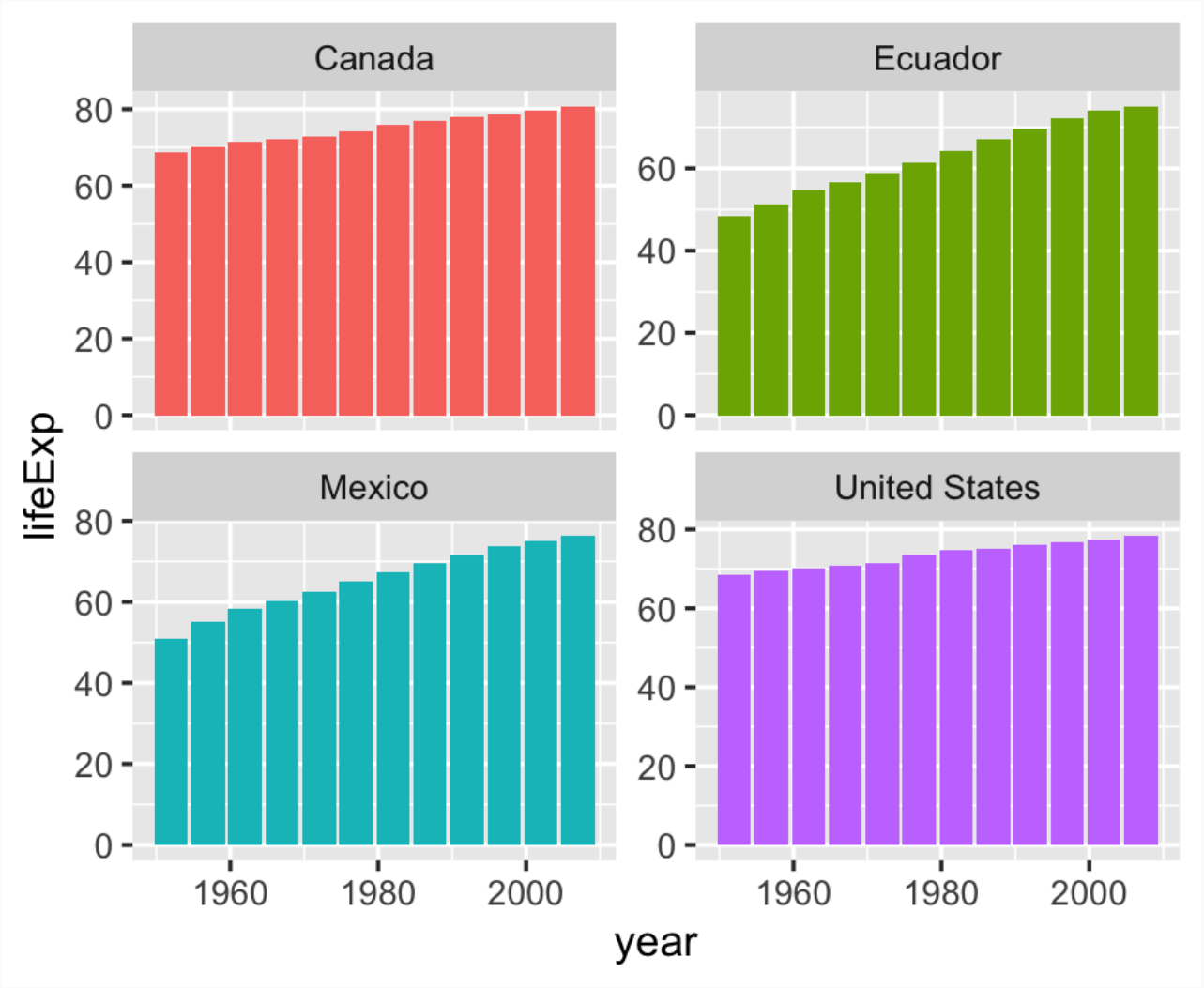


```
ggplot(americas) +
  aes(
    x = year,
    y = pop / 10^6,
    fill = country
  ) +
  geom_col(
    position = "dodge"
  ) +
  facet_wrap(~ country,
    scales = "free_y") +
  guides(fill = FALSE)
```

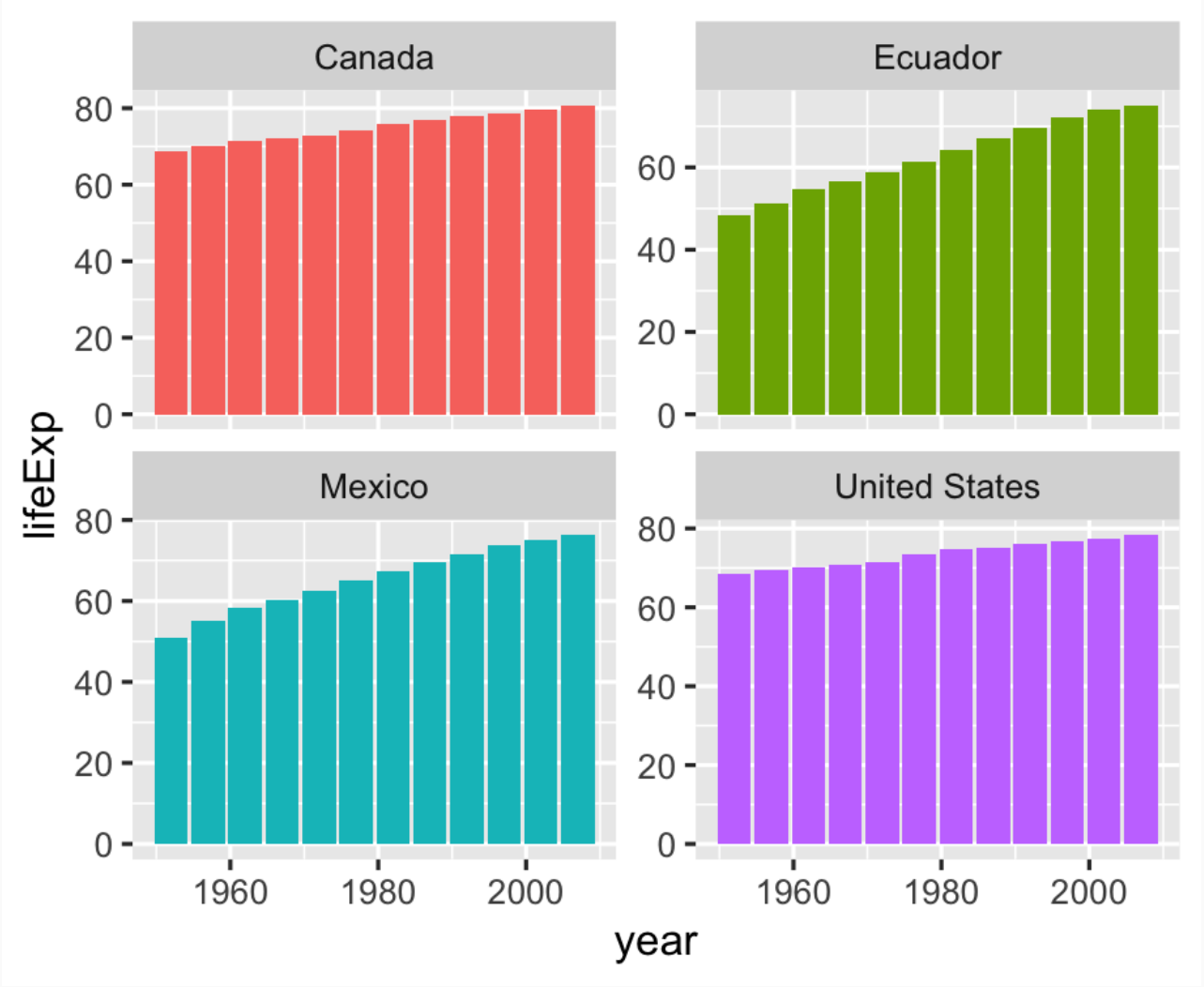


What about life expectancy again?

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

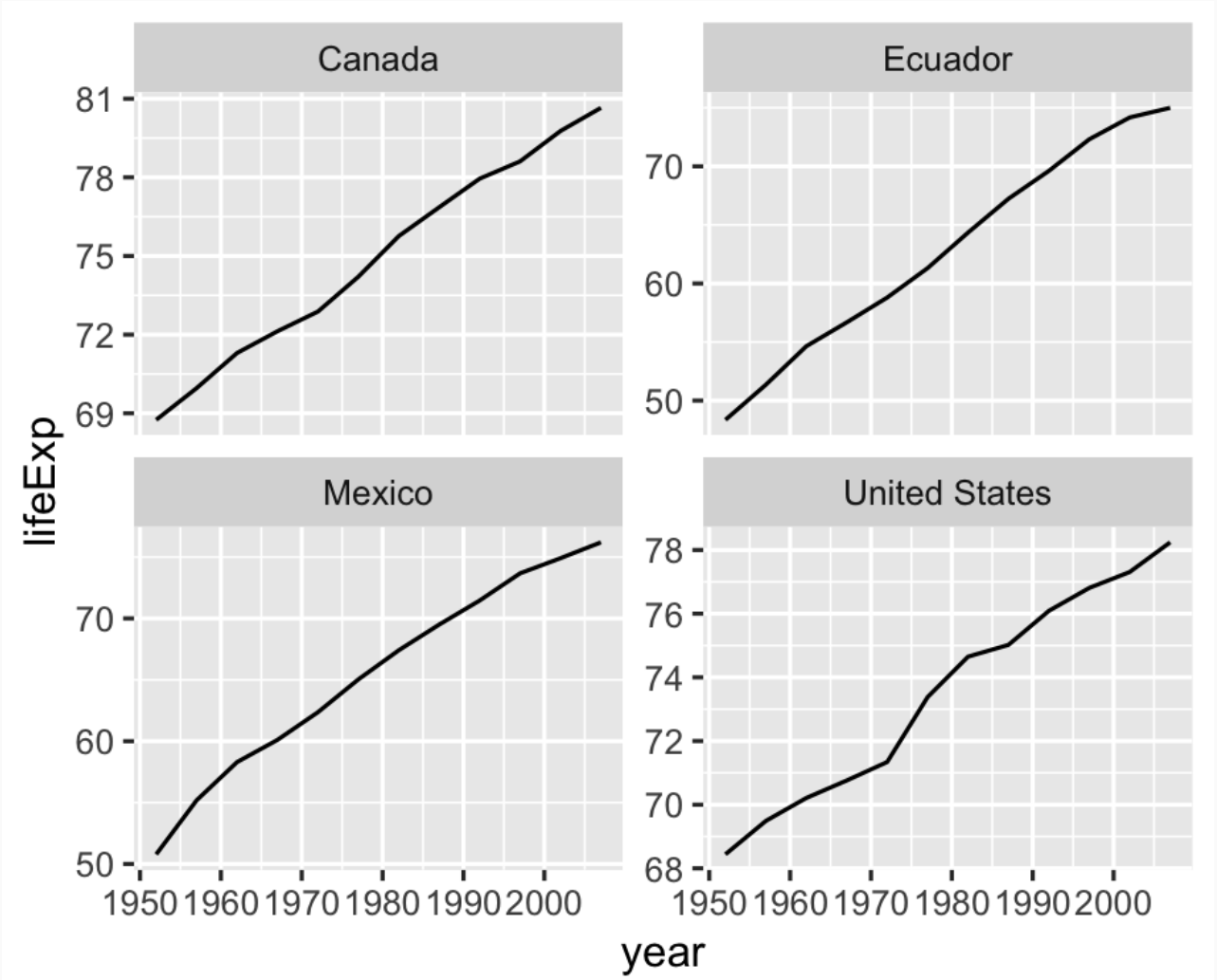


```
ggplot(americas) +
  aes(
    x = year,
    y = lifeExp,
    fill = country
  ) +
  geom_col(
    position = "dodge"
  ) +
  facet_wrap(~ country,
    scales = "free_y") +
  guides(fill = FALSE)
```

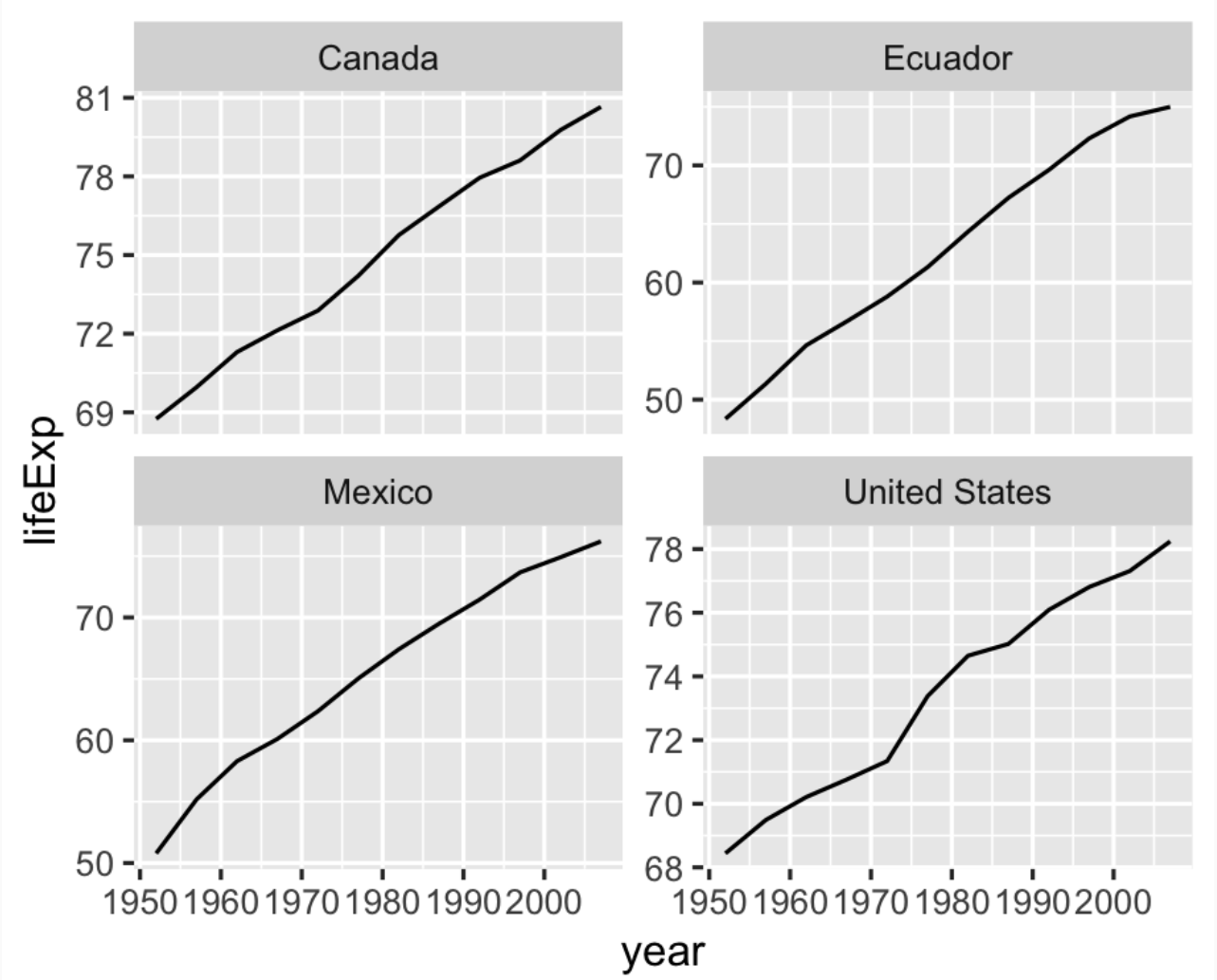


This should really be ▯ instead of ▯

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

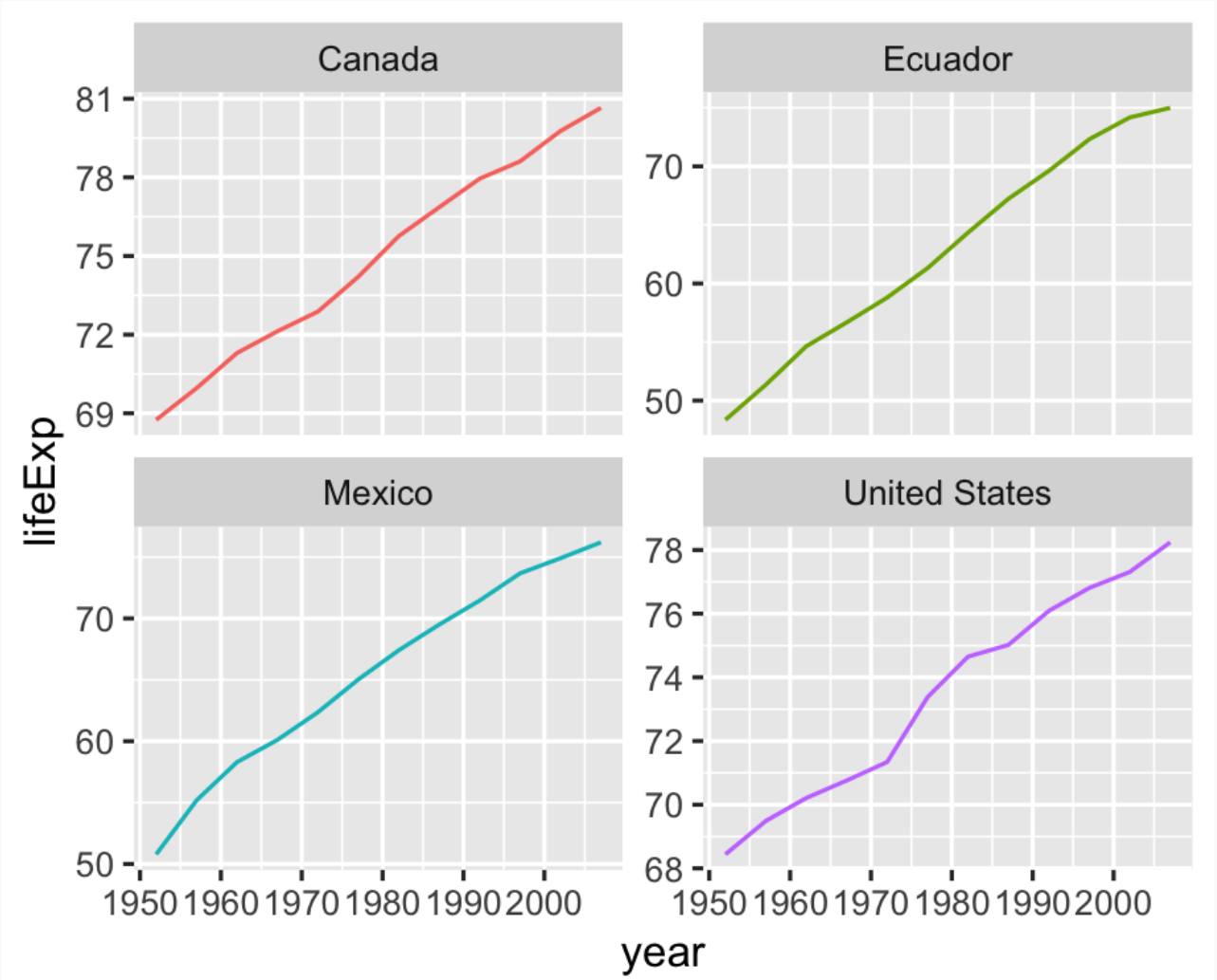


```
ggplot(americas) +
  aes(
    x = year,
    y = lifeExp,
    fill = country
  ) +
  geom_line() +
  facet_wrap(~ country,
    scales = "free_y") +
  guides(fill = FALSE)
```

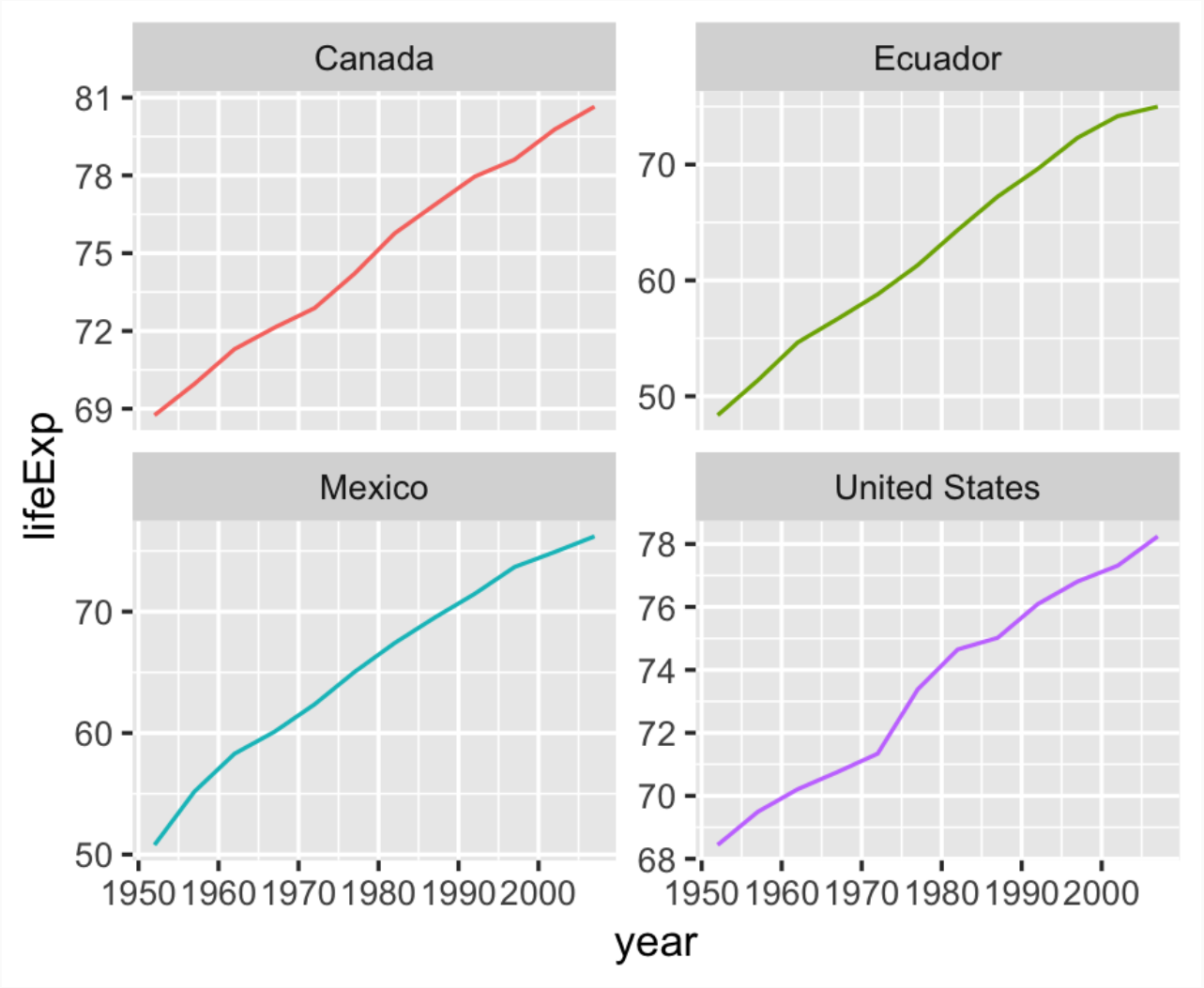


are **fill**ed, are **color**ed

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

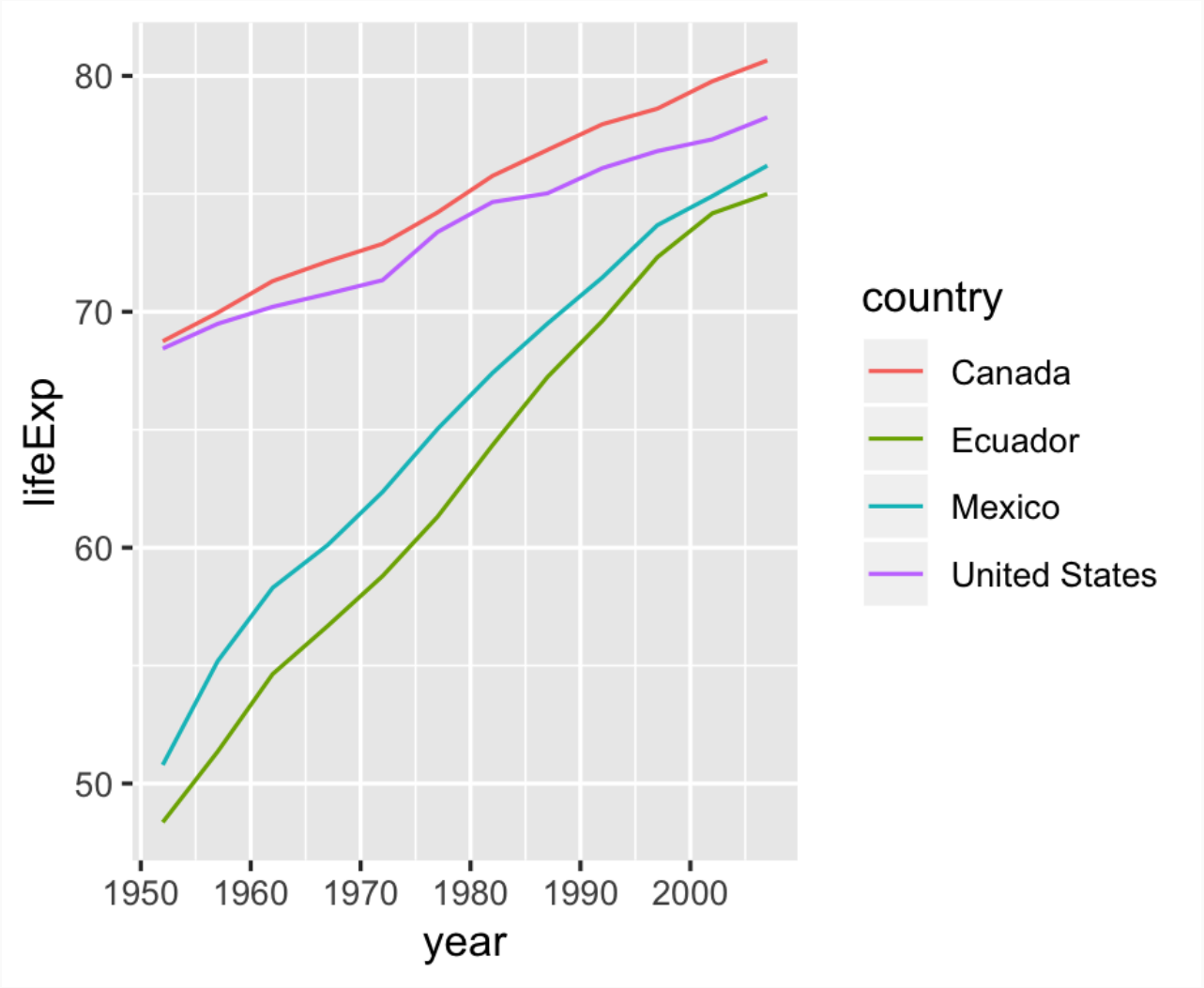


```
ggplot(americas) +
  aes(
    x = year,
    y = lifeExp,
    color = country
  ) +
  geom_line() +
  facet_wrap(~ country,
    scales = "free_y") +
  guides(color = FALSE)
```

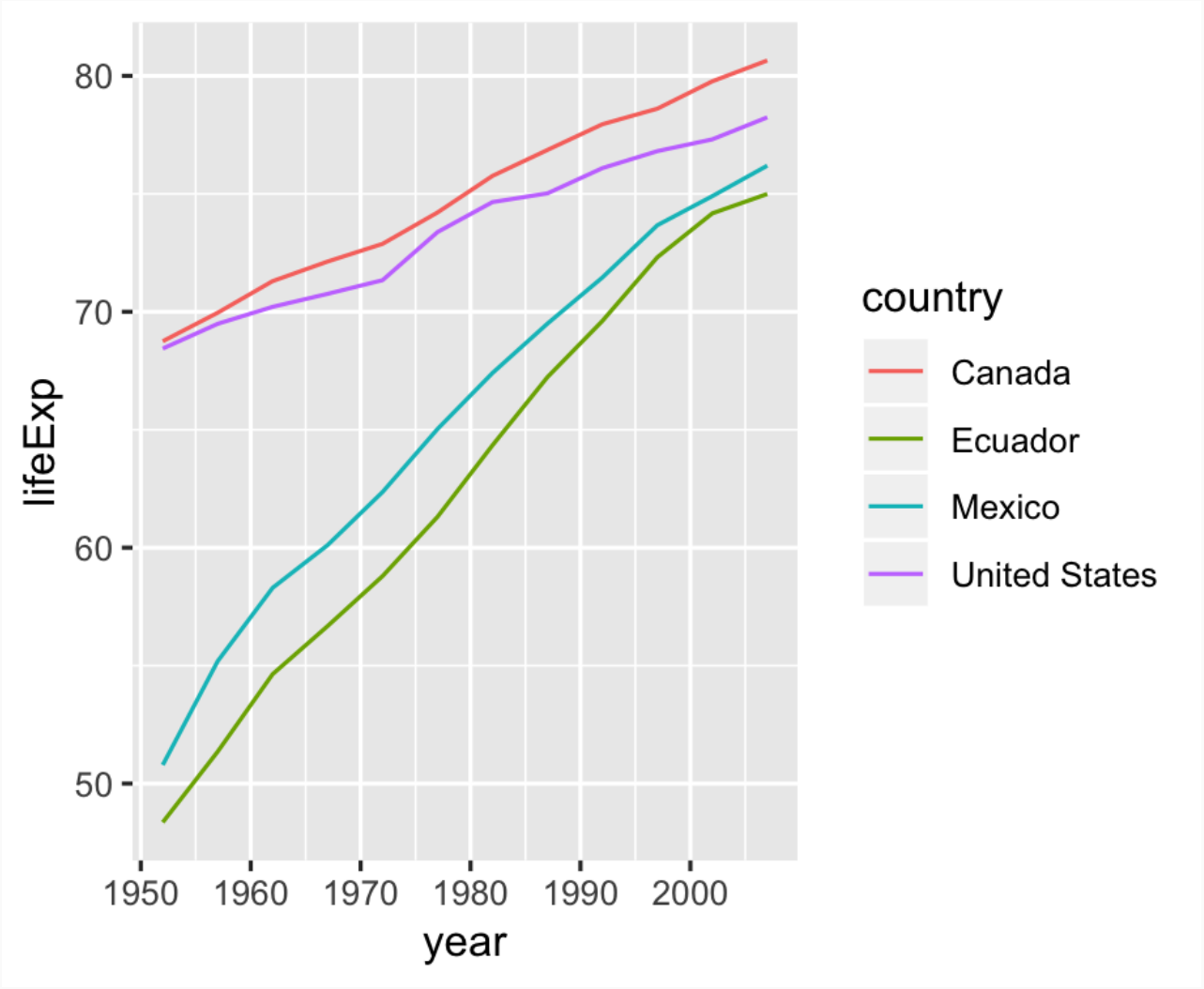


Altogether now!

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



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

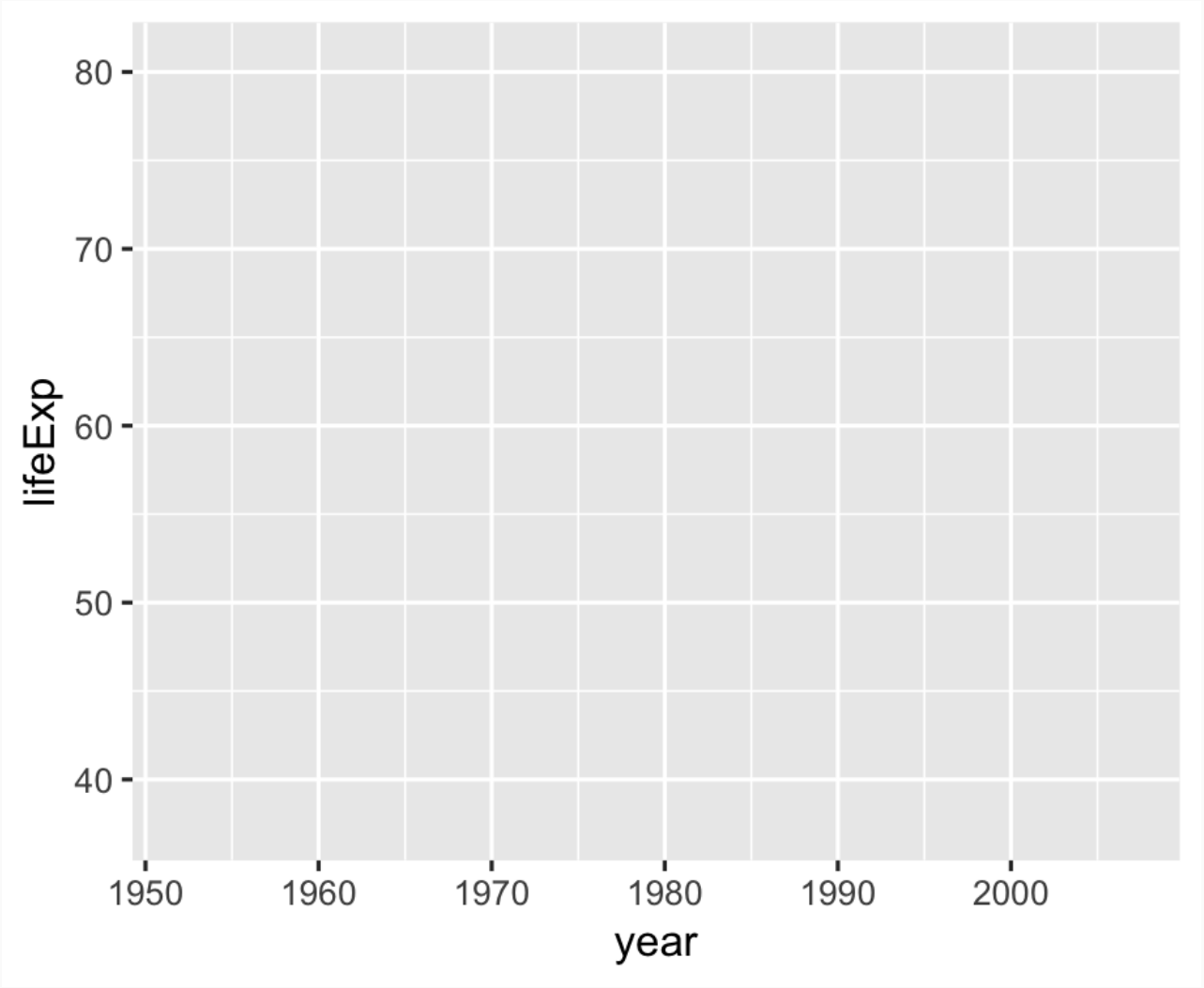


Okay, changing gears again. What is range of life expectancy in Americas?

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

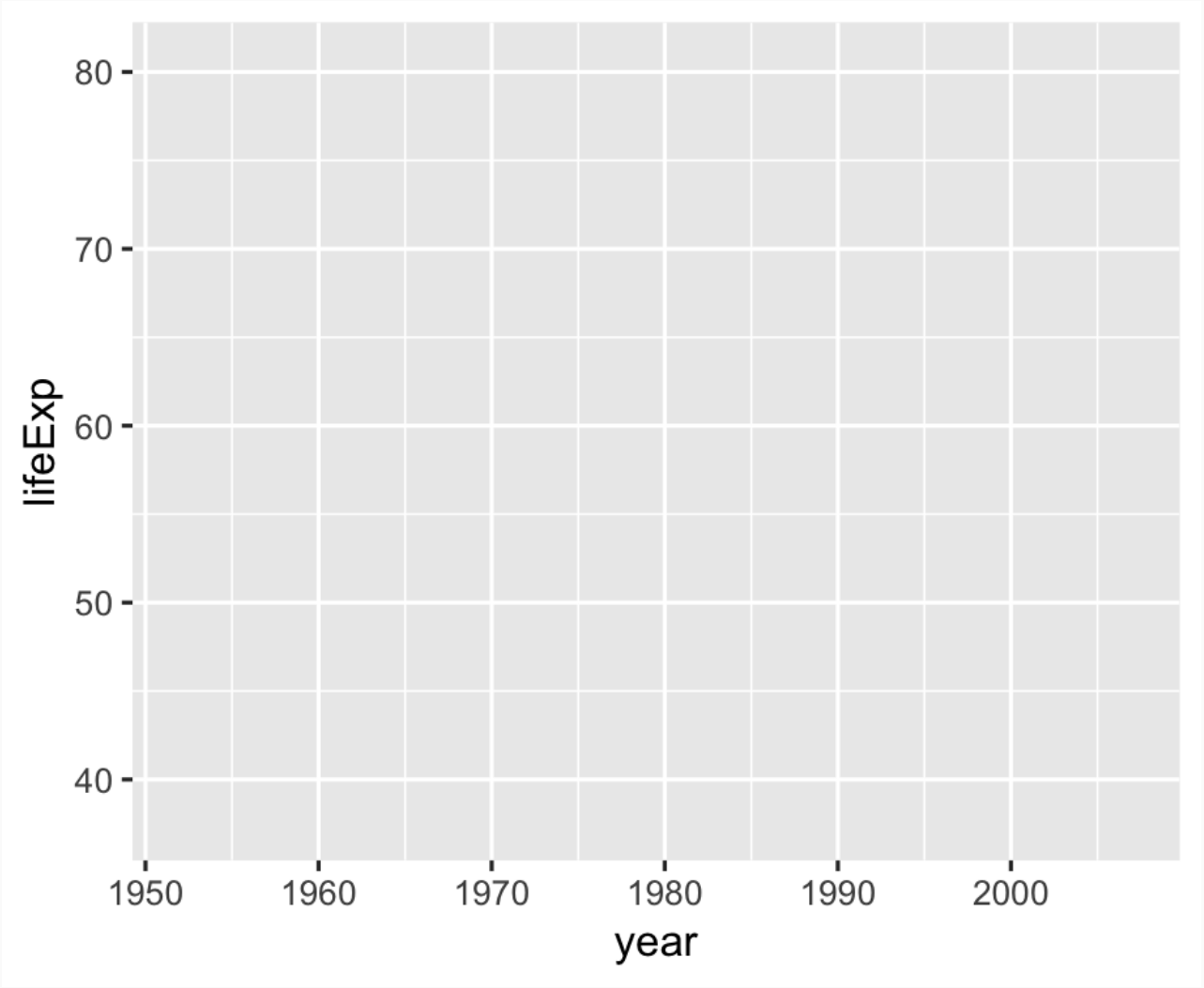
You can pipe into `ggplot()`!

Just watch for `%>%` changing to `+`



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

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



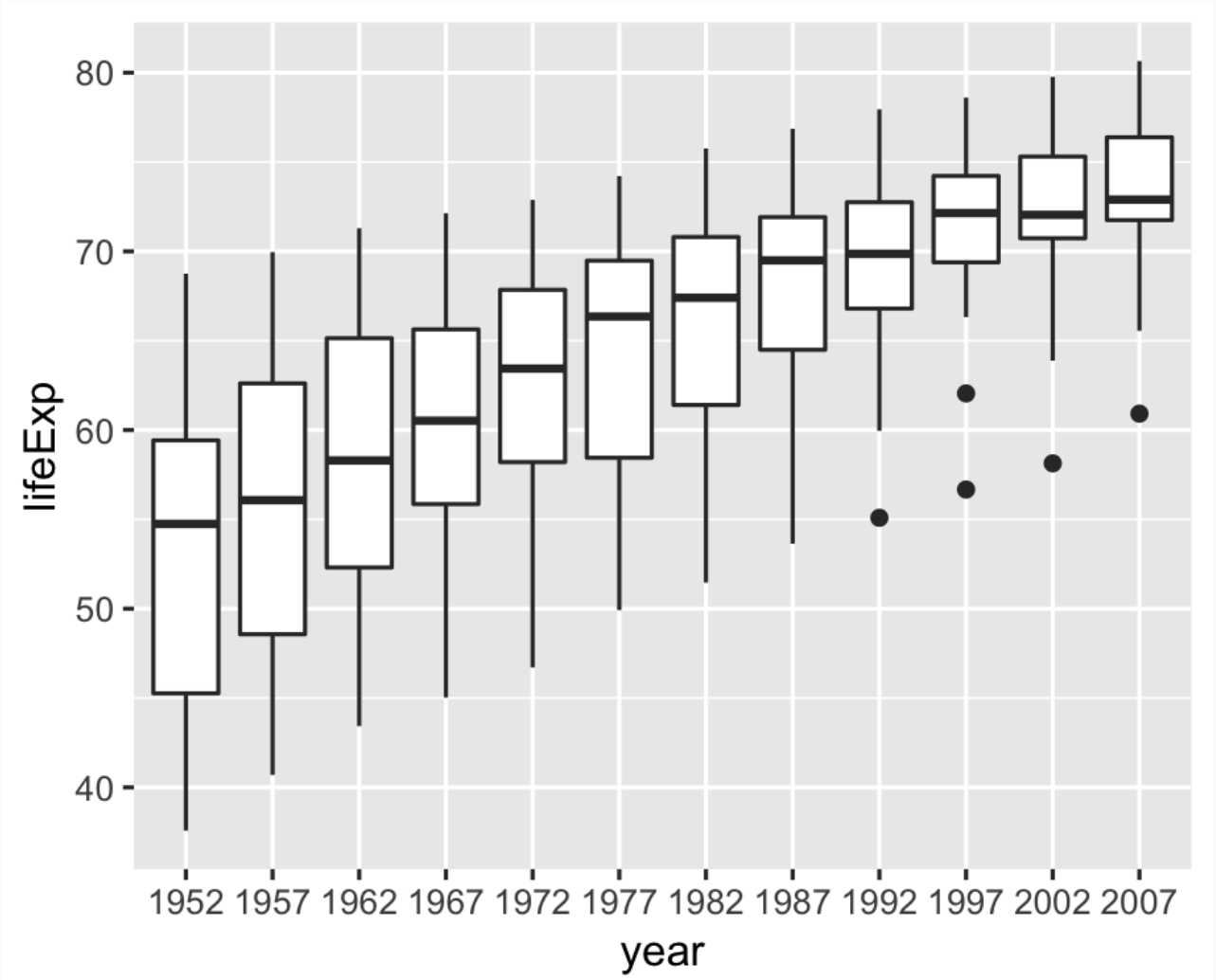
Boxplot for life expectancy range

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

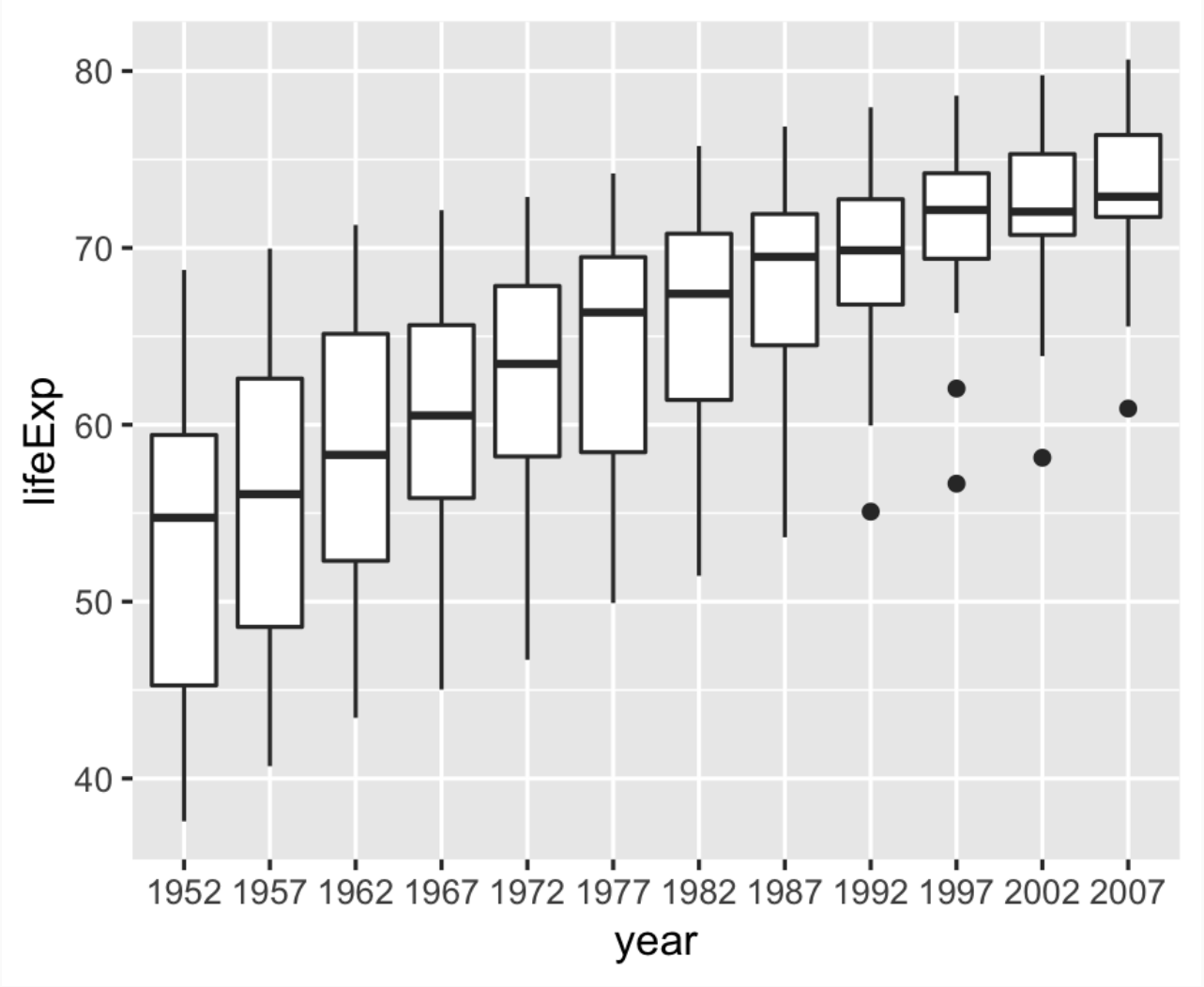
```
gapminder %>%
  filter(
    continent == "Americas"
  ) %>%
  ggplot() +
  aes(
    x = year,
    y = lifeExp
  ) +
  geom_boxplot()
```

Why not boxplots by year?

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

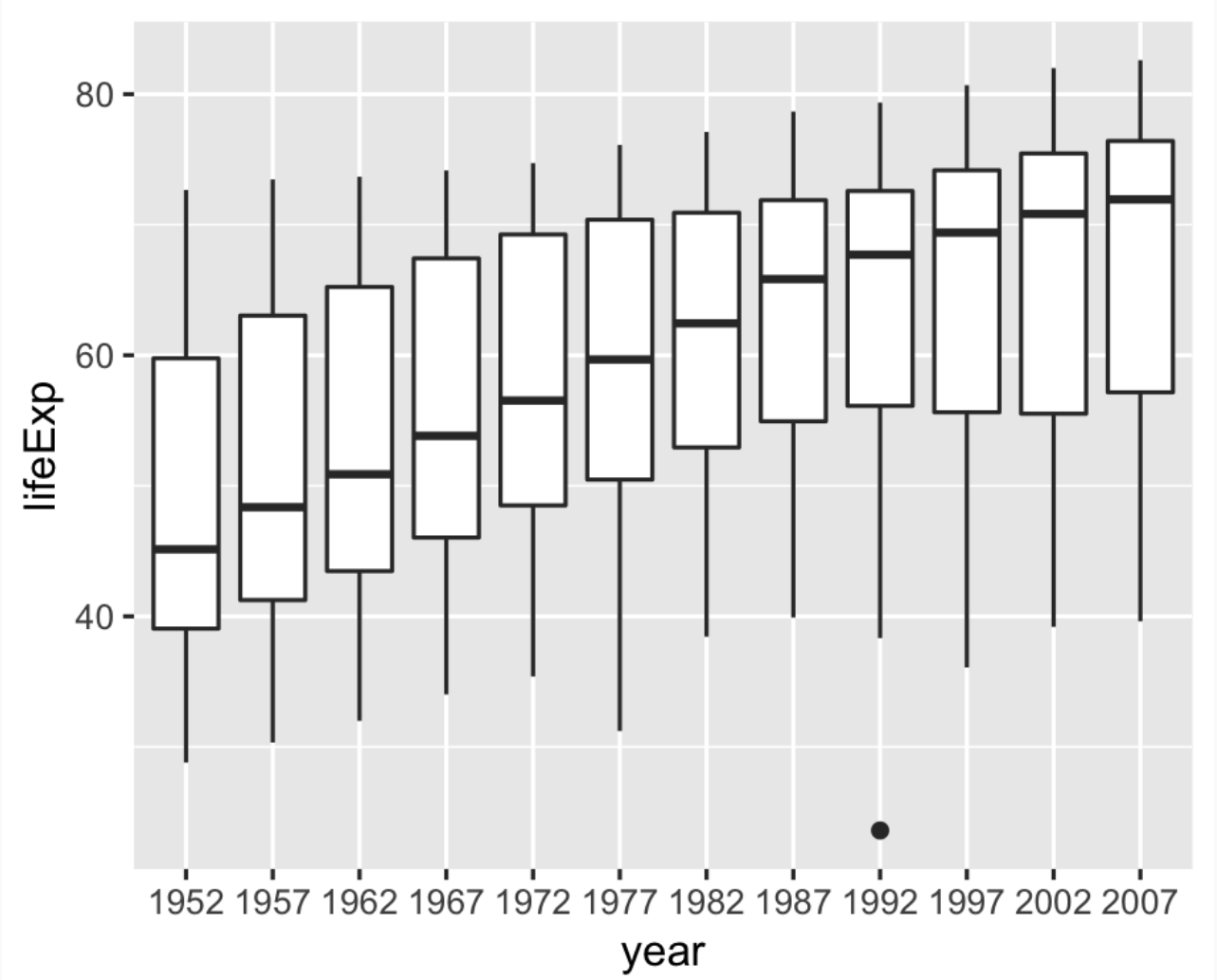


```
gapminder %>%
  filter(
    continent == "Americas"
  ) %>%
  mutate(
    year = factor(year)
  ) %>%
  ggplot() +
  aes(
    x = year,
    y = lifeExp
  ) +
  geom_boxplot()
```

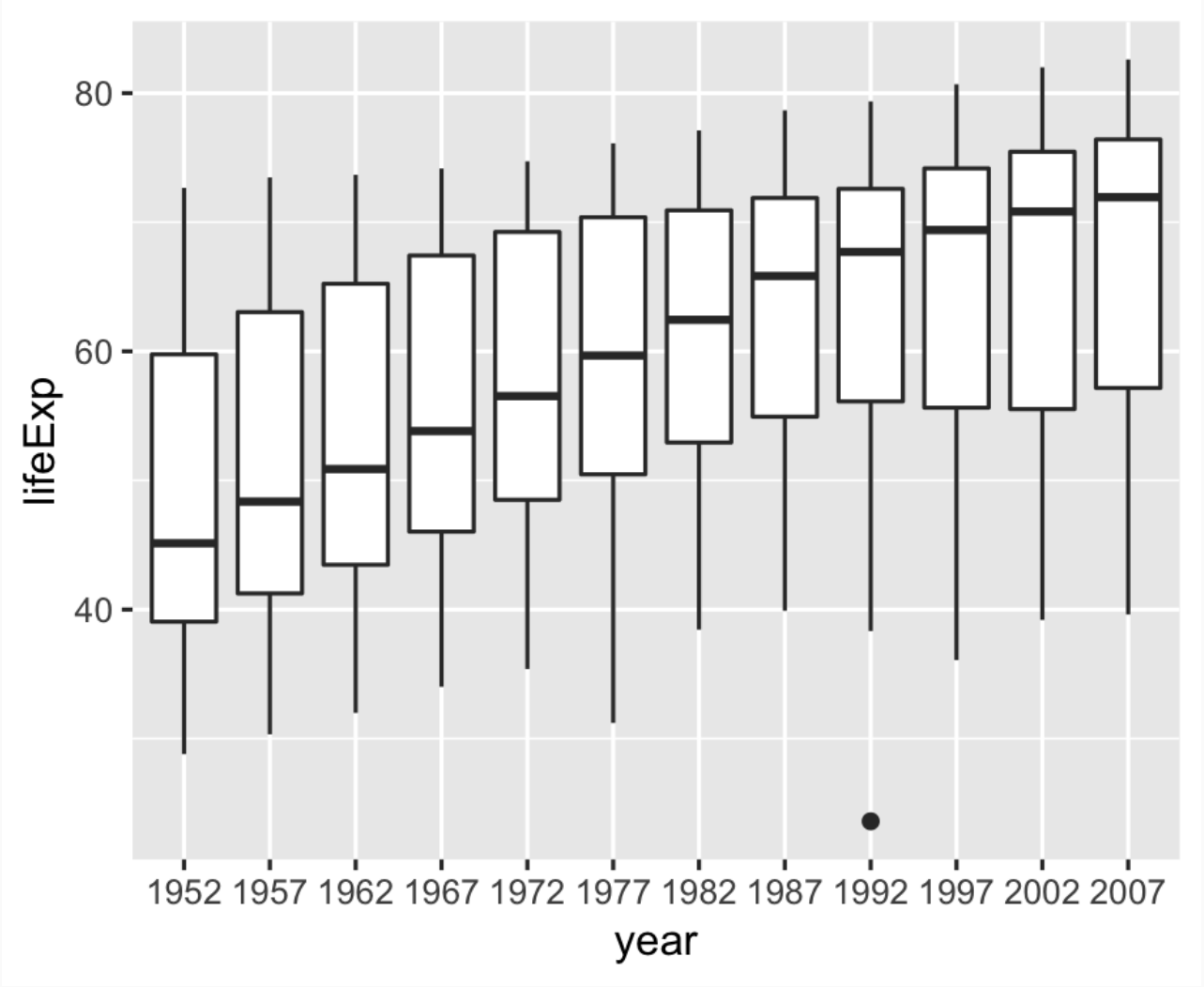


OK, what about global life expectancy?

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

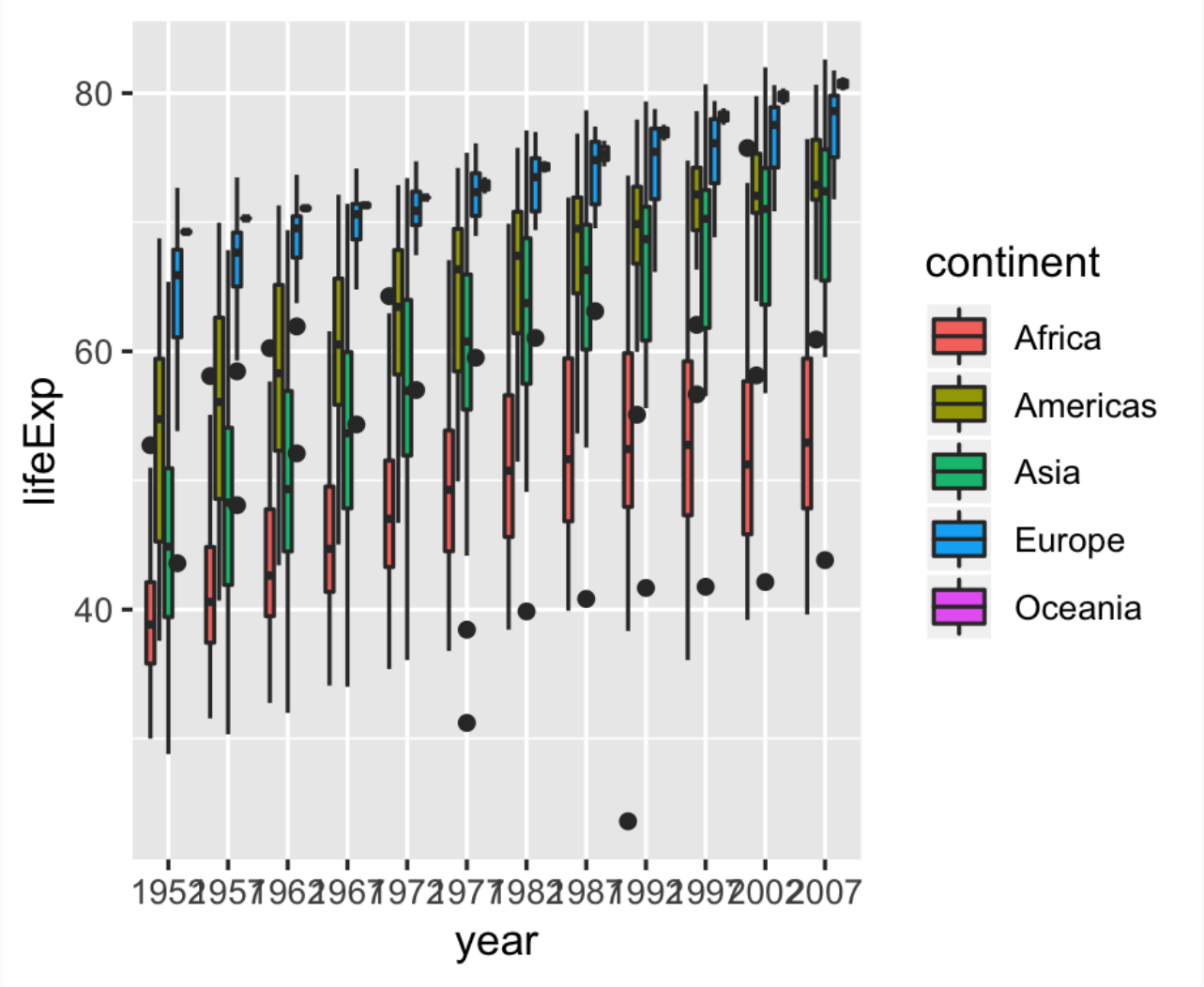


```
gapminder %>%
  # filter(
  #   continent == "Americas"
  # ) %>%
  mutate(
    year = factor(year)
  ) %>%
  ggplot() +
  aes(
    x = year,
    y = lifeExp
  ) +
  geom_boxplot()
```

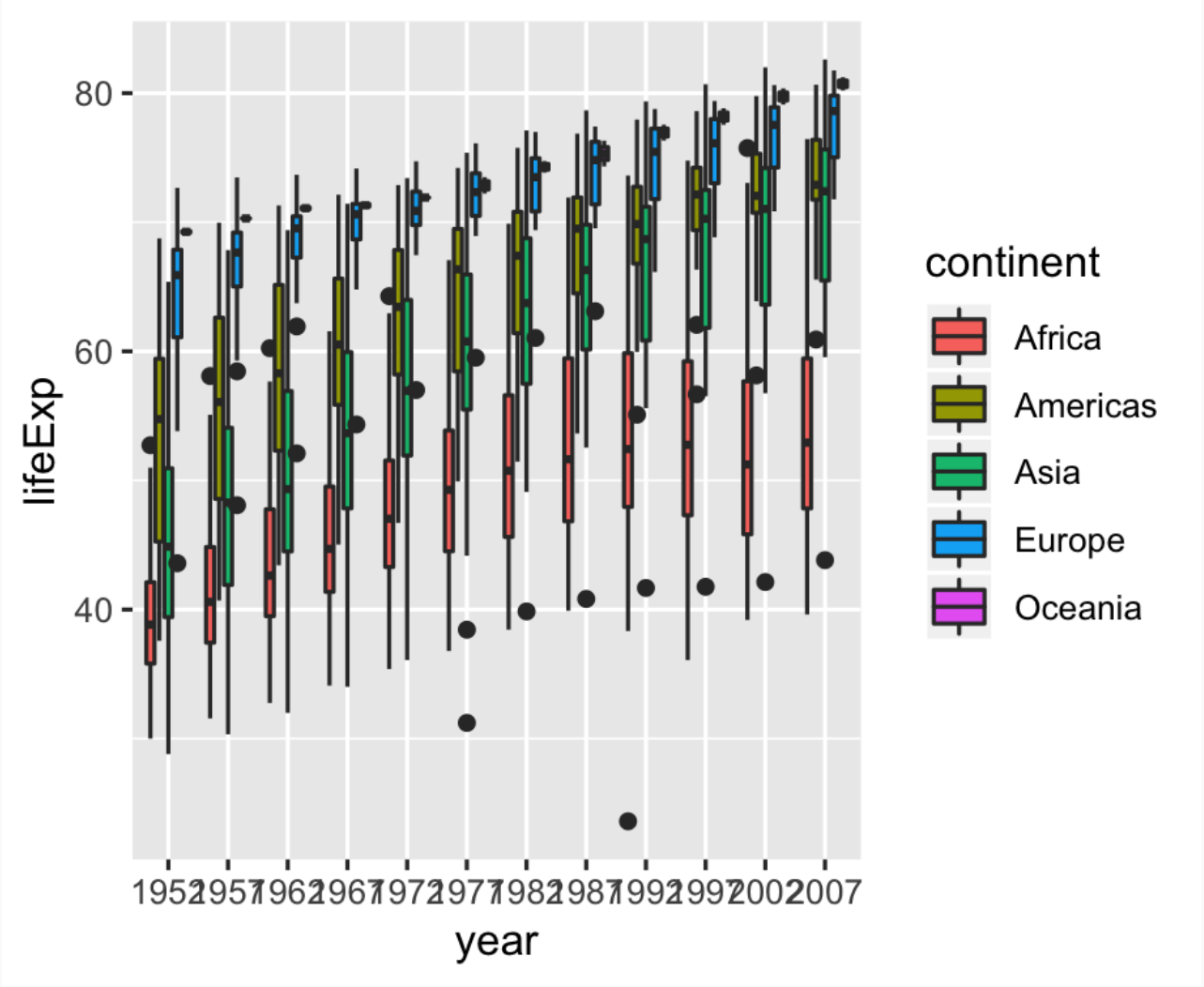


Can we have cute little boxplots for each continent?

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

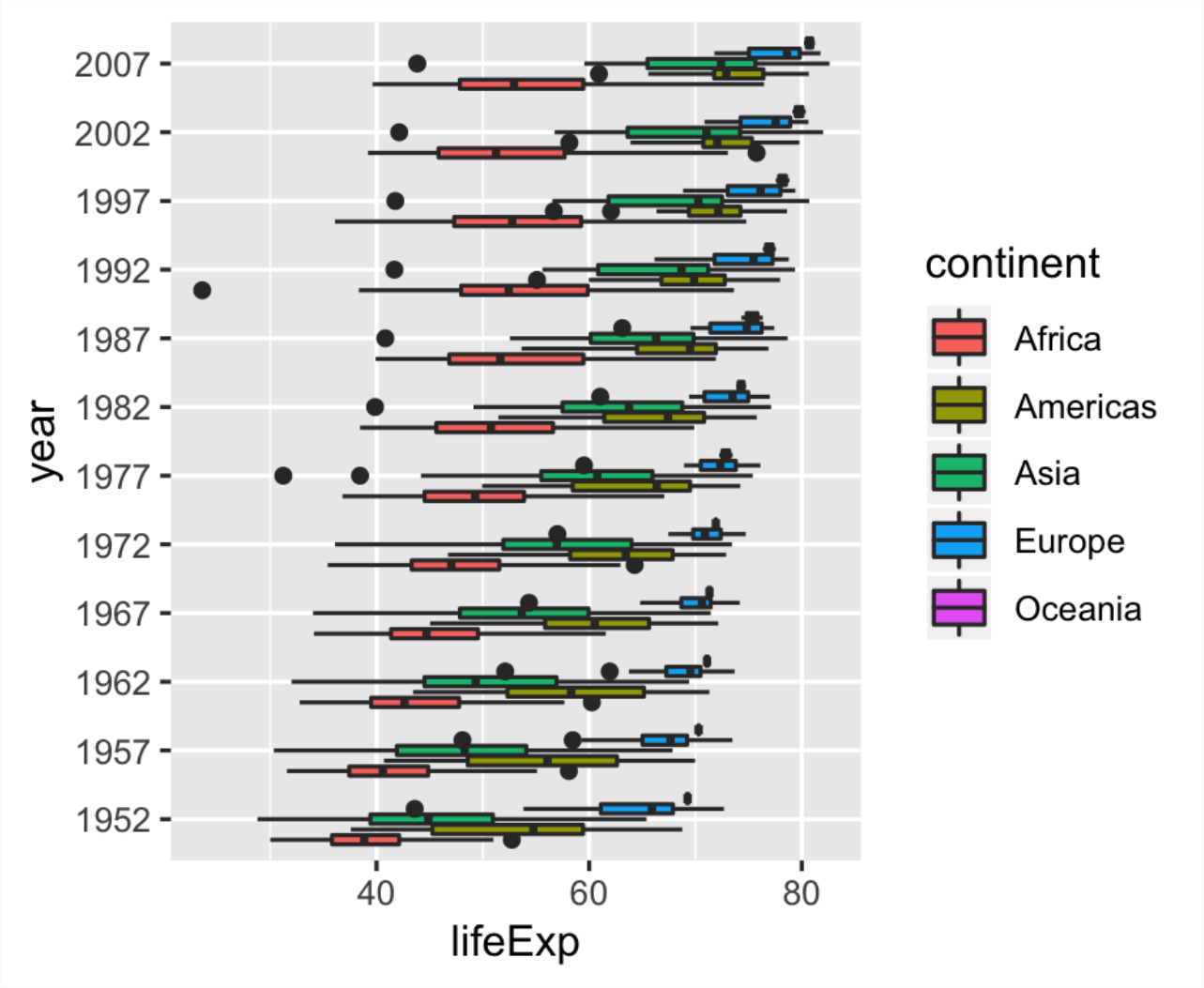


```
gapminder %>%
  mutate(
    year = factor(year)
  ) %>%
  ggplot() +
  aes(
    x = year,
    y = lifeExp,
    fill = continent
  ) +
  geom_boxplot()
```

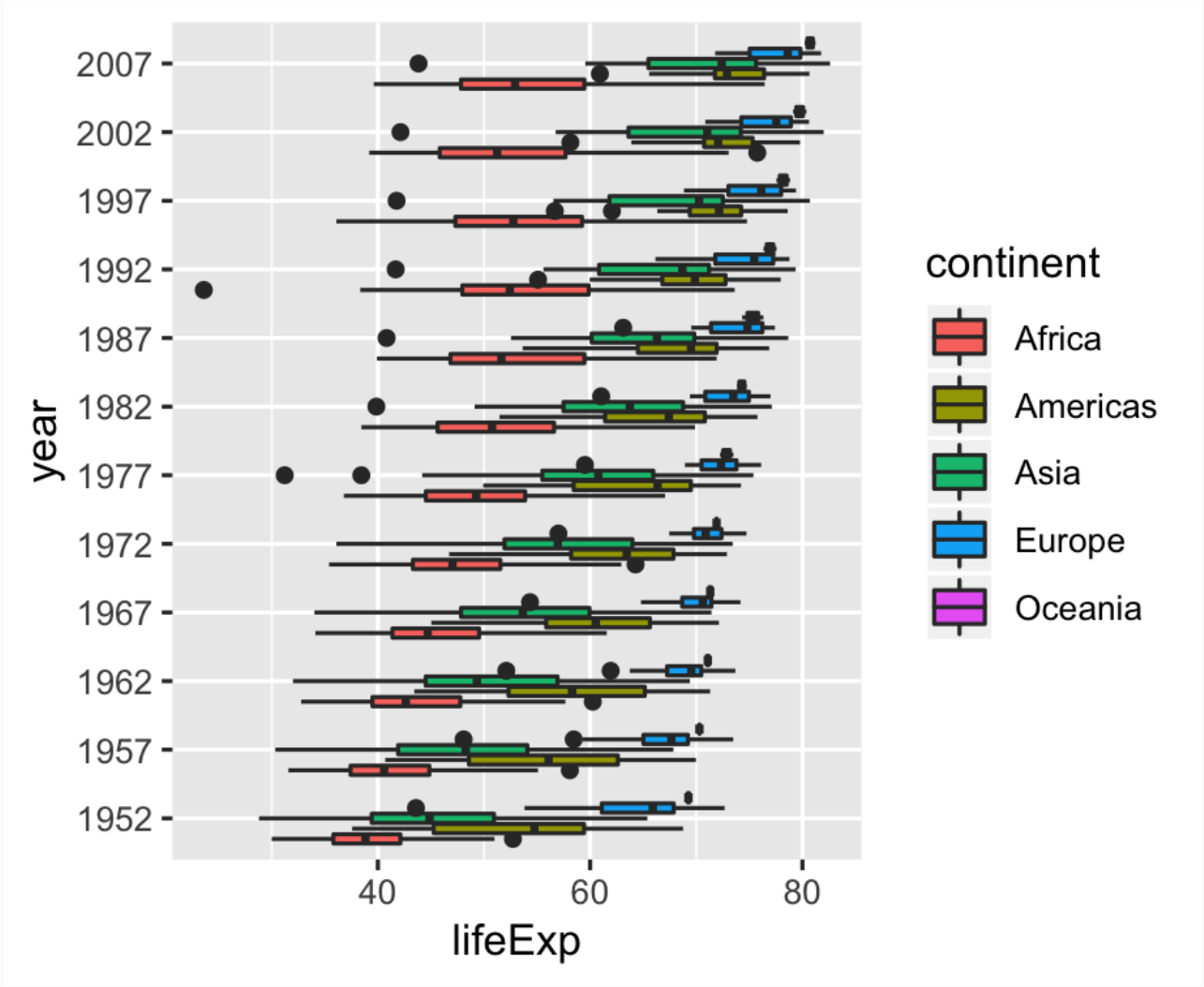


Hard to read years, let's rotate

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

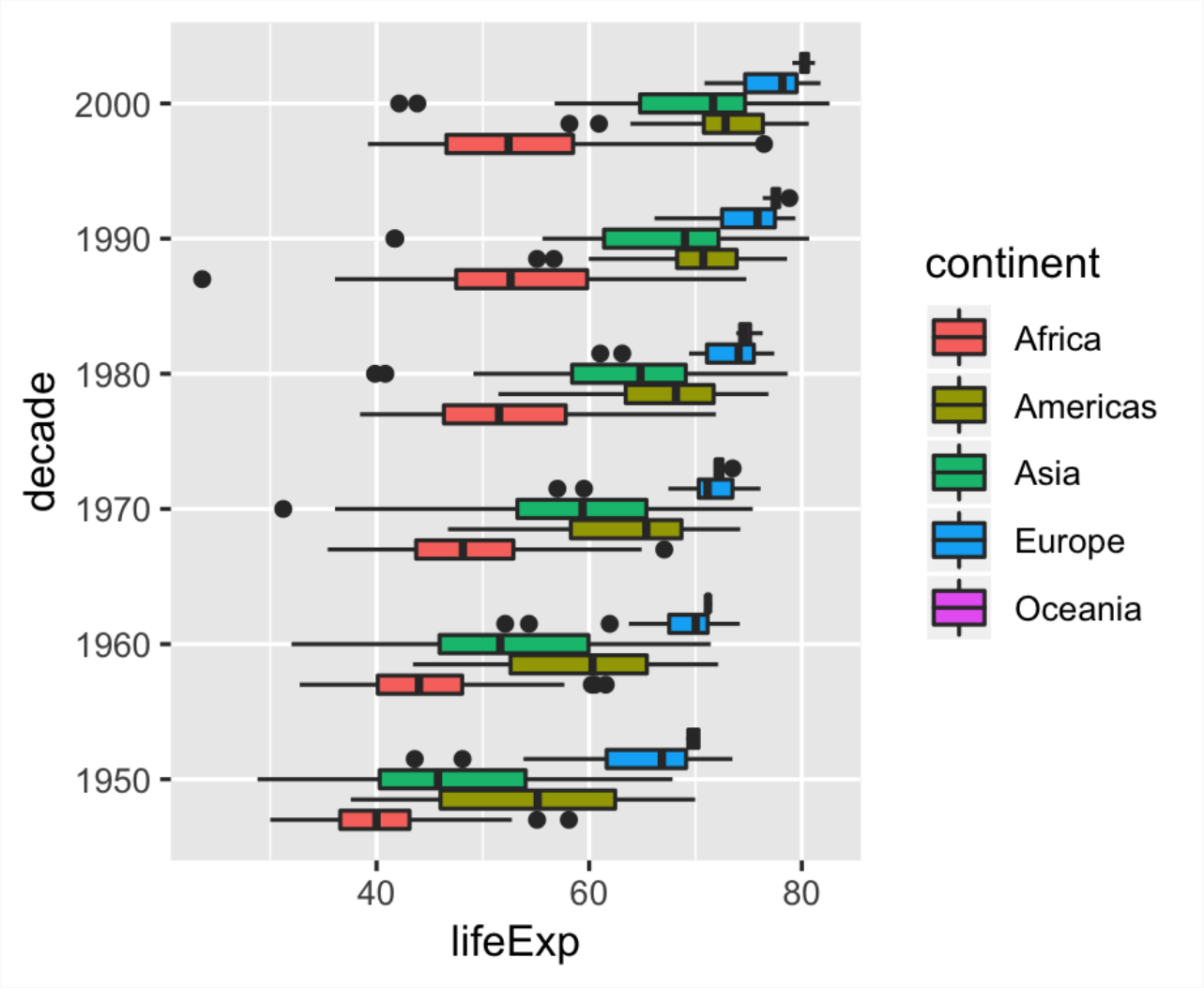


```
gapminder %>%
  mutate(
    year = factor(year)
  ) %>%
  ggplot() +
  aes(
    x = year,
    y = lifeExp,
    fill = continent
  ) +
  geom_boxplot() +
  coord_flip()
```

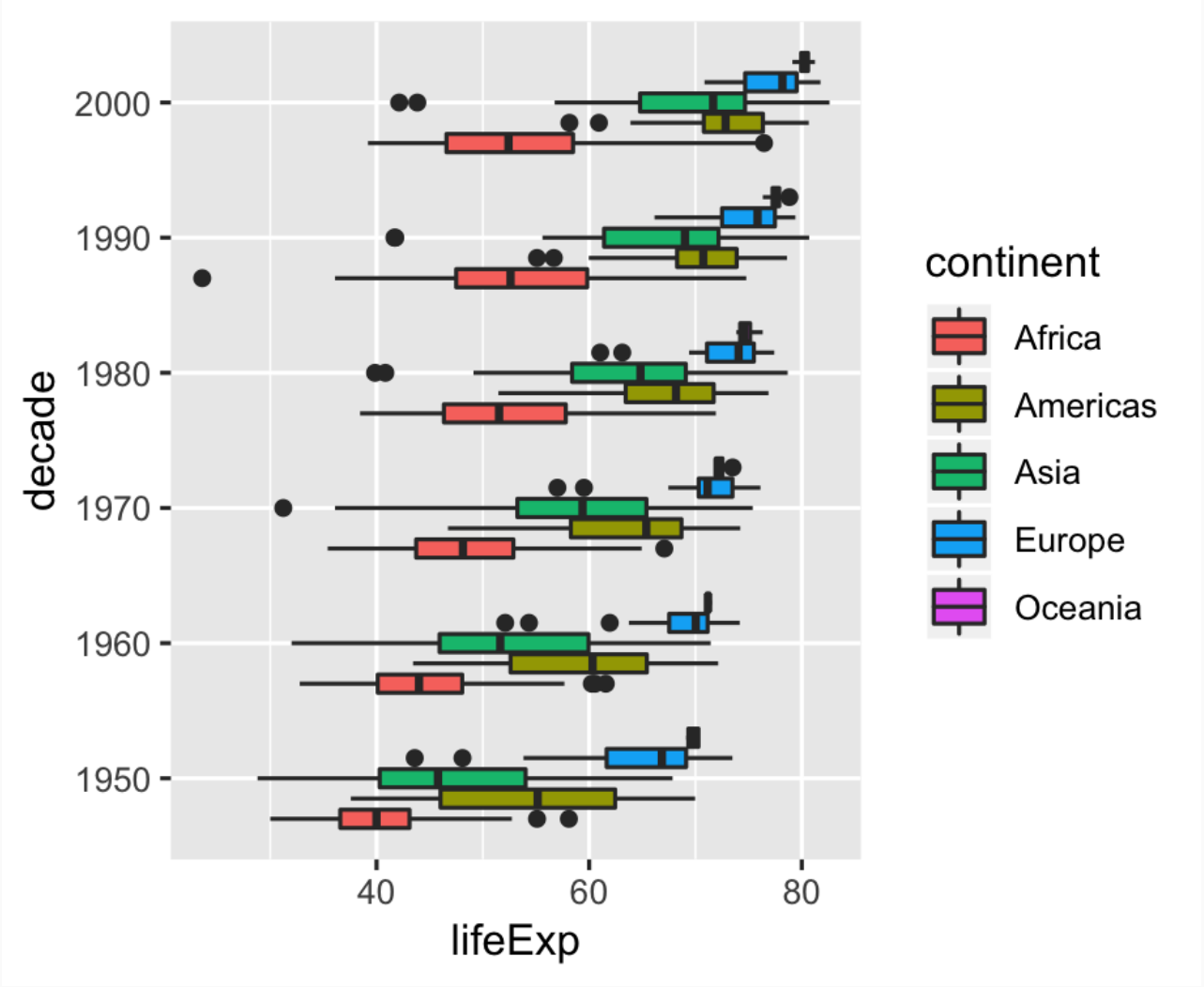


Use `dplyr` to group by decade

```r
gapminder %>%
  mutate(
    decade = floor(year / 10)
    decade = decade * 10,
    decade = factor(decade)
  ) %>%
  ggplot() +
  aes(
    x = decade,
    y = lifeExp,
    fill = continent
  ) +
  geom_boxplot() +
  coord_flip()
```

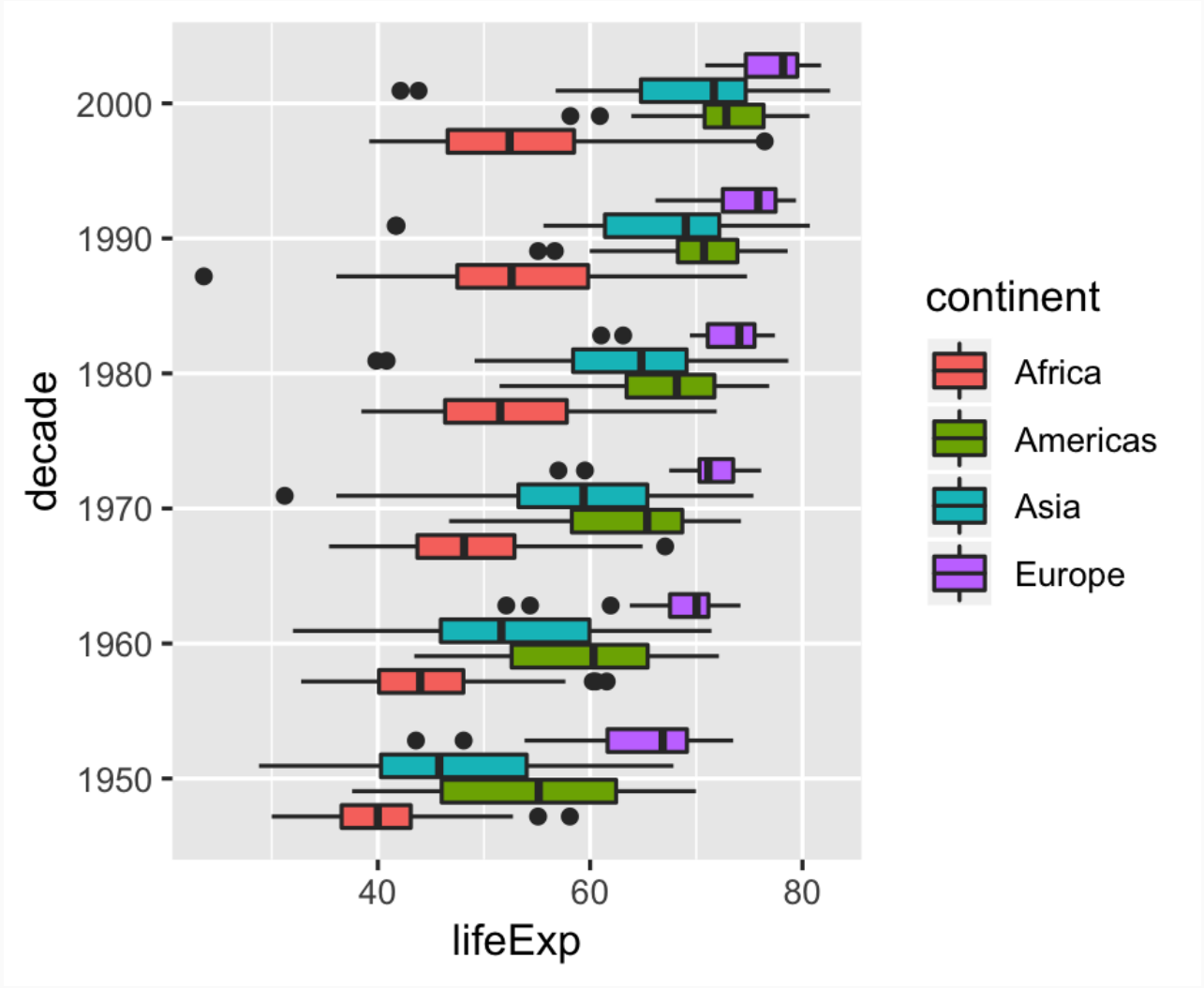


```
gapminder %>%
  mutate(
    decade = floor(year / 10)
    decade = decade * 10,
    decade = factor(decade)
  ) %>%
  ggplot() +
  aes(
    x = decade,
    y = lifeExp,
    fill = continent
  ) +
  geom_boxplot() +
  coord_flip()
```



Let's hide Oceania, sorry

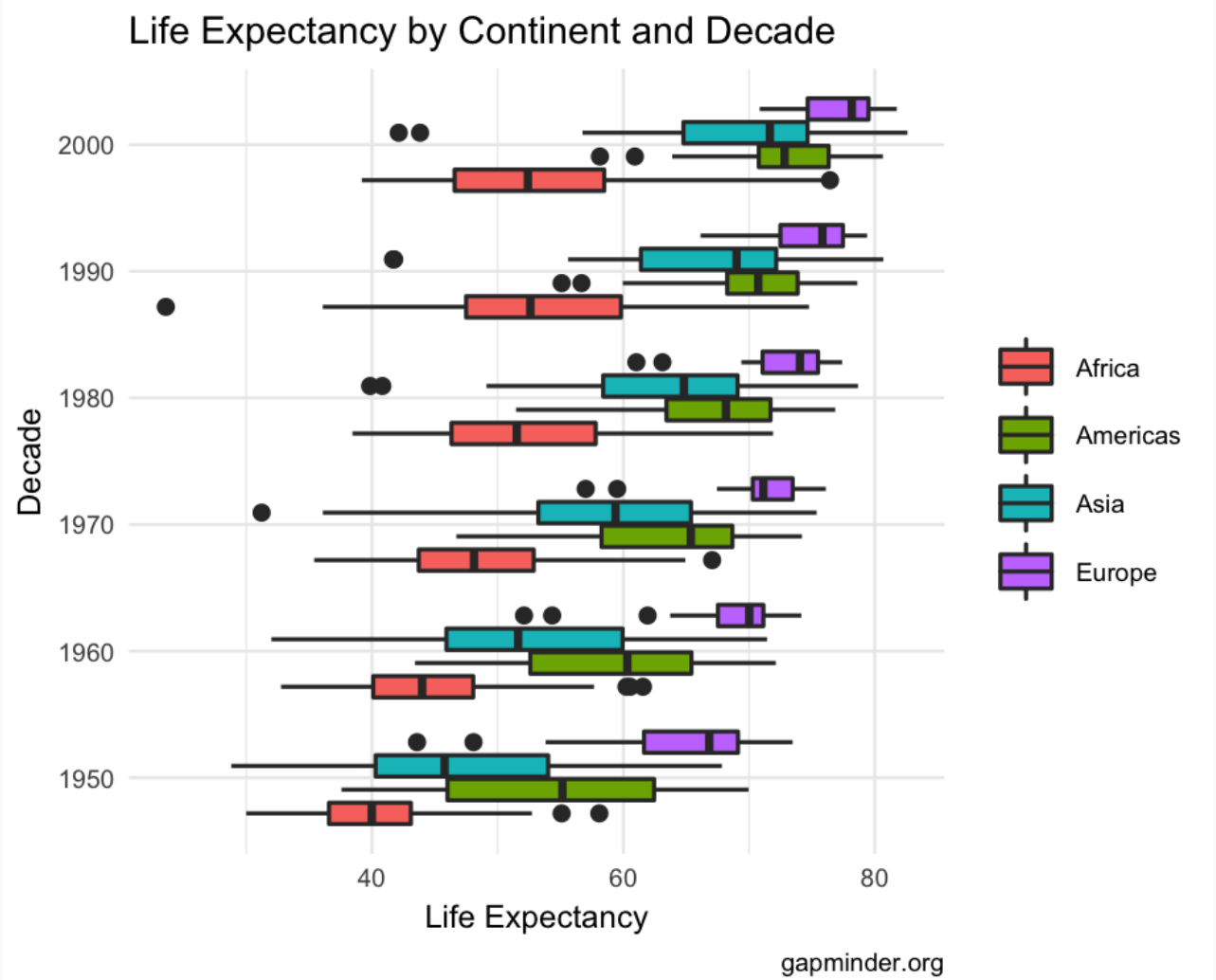
```r
g ← gapminder %>%
  filter(
    continent ≠ "Oceania"
  ) %>%
  mutate(
    decade = floor(year / 10)
  ) %>%
  ggplot() +
  aes(
    x = decade,
    y = lifeExp,
    fill = continent
  ) +
  geom_boxplot() +
  coord_flip()
```



```
g +
  theme_minimal(8) +
  labs(
    y = "Life Expectancy",
    x = "Decade",
    fill = NULL,
    title = "Life Expectancy b
    caption = "gapminder.org"
  )
```

Note `x` and `y` are *original* aesthetics, `coord_flip()` happens *after*.

Remove labels by setting `= NULL`.



Level up

Inspired by "The Best Stats You've Ever Seen" by Hans Rosling

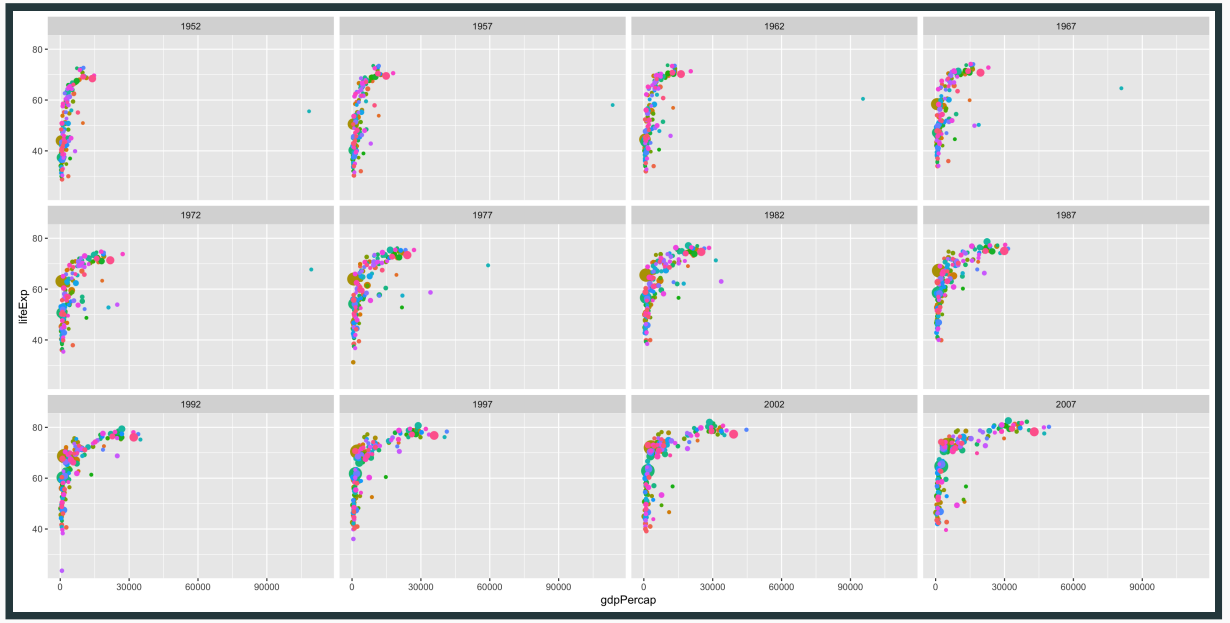
http://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen

Create the initial layout

```
g_hr ←
  ggplot(gapminder) +
  aes(x = gdpPercap, y = lifeExp, size = pop, color = country) +
  geom_point() +
  facet_wrap(~year)
```

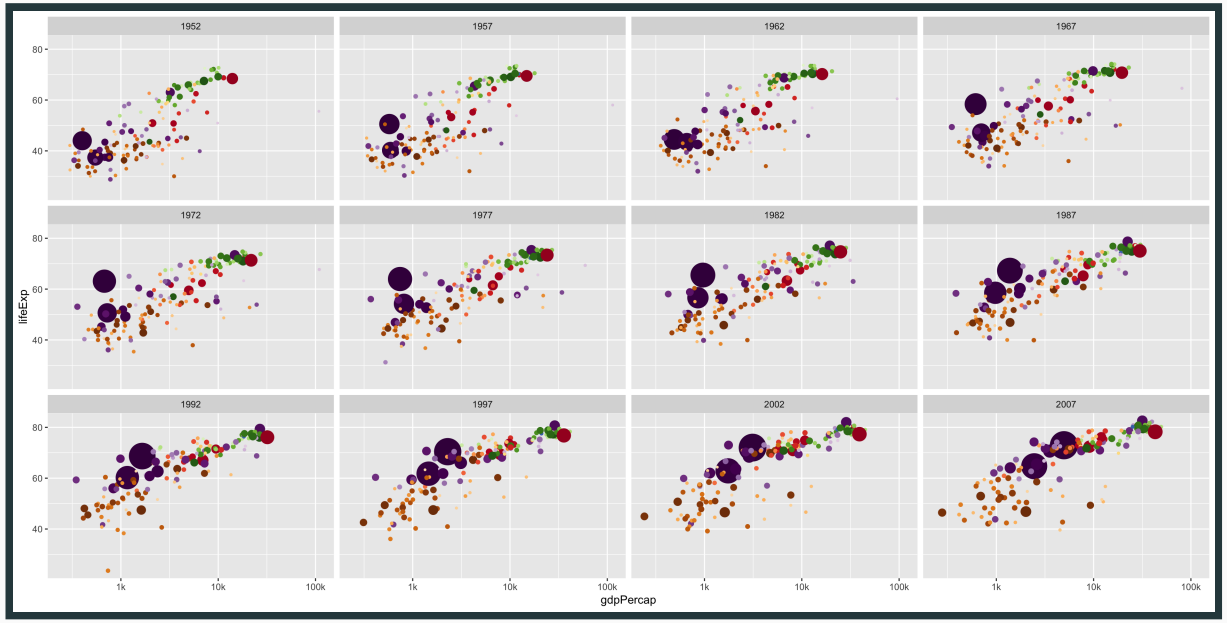
Hide the guides

```
g_hr ←
  ggplot(gapminder) +
  aes(x = gdpPercap, y = lifeExp, size = pop, color = country) +
  geom_point() +
  facet_wrap(~year) +
  guides(color = FALSE, size = FALSE)
```



Adjust scales of x-axis, color, and size

```
g_hr ←
  g_hr +
  scale_x_log10(breaks = c(10^3, 10^4, 10^5), labels = c("1k", "10k", "100k")) +
  scale_color_manual(values = gapminder::country_colors) +
  scale_size(range = c(0.5, 12))
```

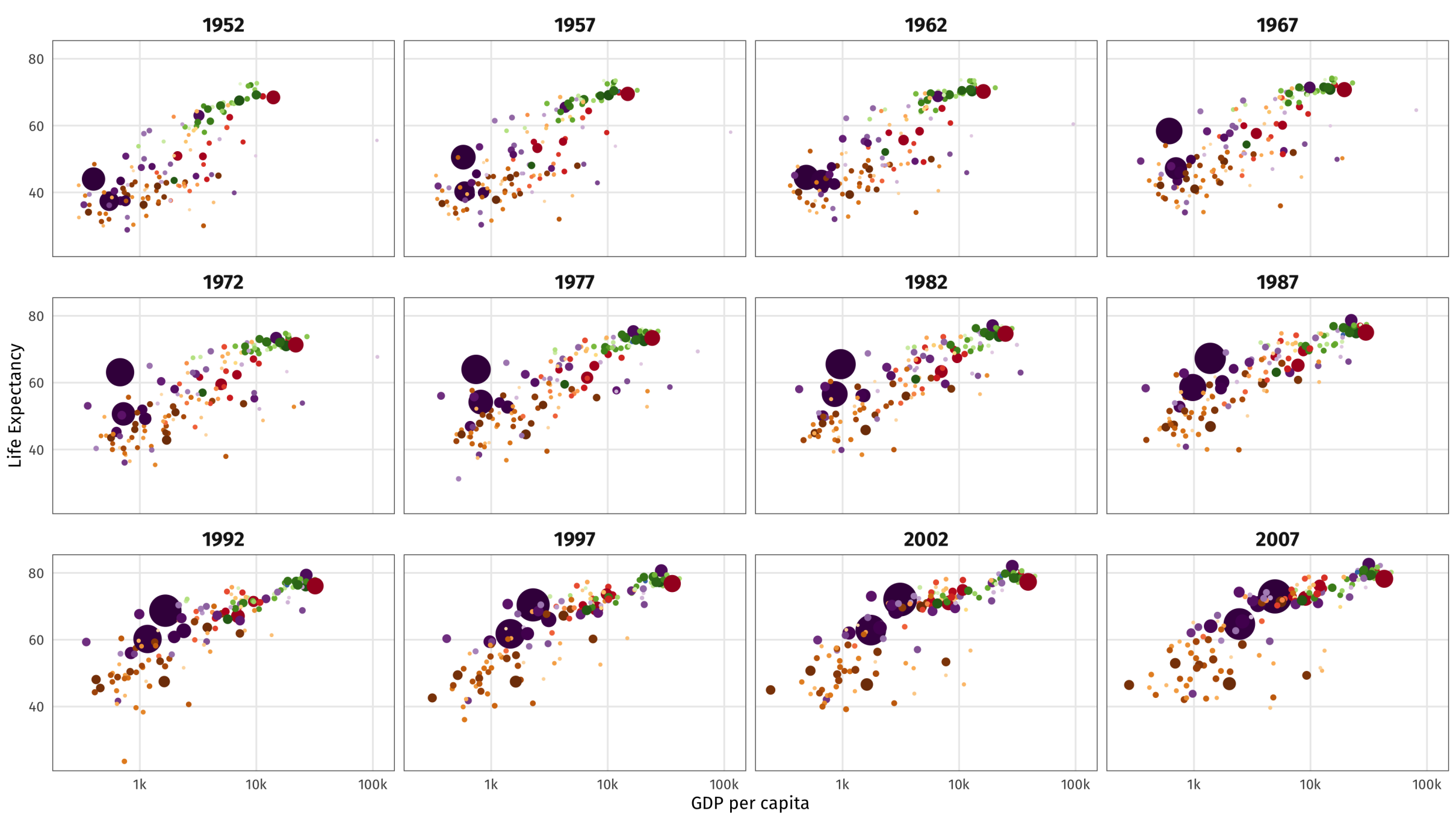


Tweak Anotations


```
g_hr ←
  g_hr +
  labs(
    x = "GDP per capita",
    y = "Life Expectancy"
  ) +
  theme_minimal(base_family = "Fira Sans") +
  theme(
    strip.text = element_text(size = 16, face = "bold"),
    panel.border = element_rect(fill = NA, color = "grey40"),
    panel.grid.minor = element_blank()
  )
```

The final code and plot

```
ggplot(gapminder) +
  aes(x = gdpPercap, y = lifeExp, size = pop, color = country) +
  geom_point() +
  facet_wrap(~year) +
  guides(color = FALSE, size = FALSE) +
  scale_x_log10(
    breaks = c(10^3, 10^4, 10^5),
    labels = c("1k", "10k", "100k")) +
  scale_color_manual(values = gapminder::country_colors) +
  scale_size(range = c(0.5, 12)) +
  labs(
    x = "GDP per capita",
    y = "Life Expectancy") +
  theme_minimal(14, base_family = "Fira Sans") +
  theme(
    strip.text = element_text(size = 16, face = "bold"),
    panel.border = element_rect(fill = NA, color = "grey40"),
    panel.grid.minor = element_blank())
```


1952
1957
1962
1967
1972
1977
1982
1987
1992
1997
2002
2007
80
60
40
Life Expectancy
1k
10k
100k
GDP per capita

Special Bonus: Animated!

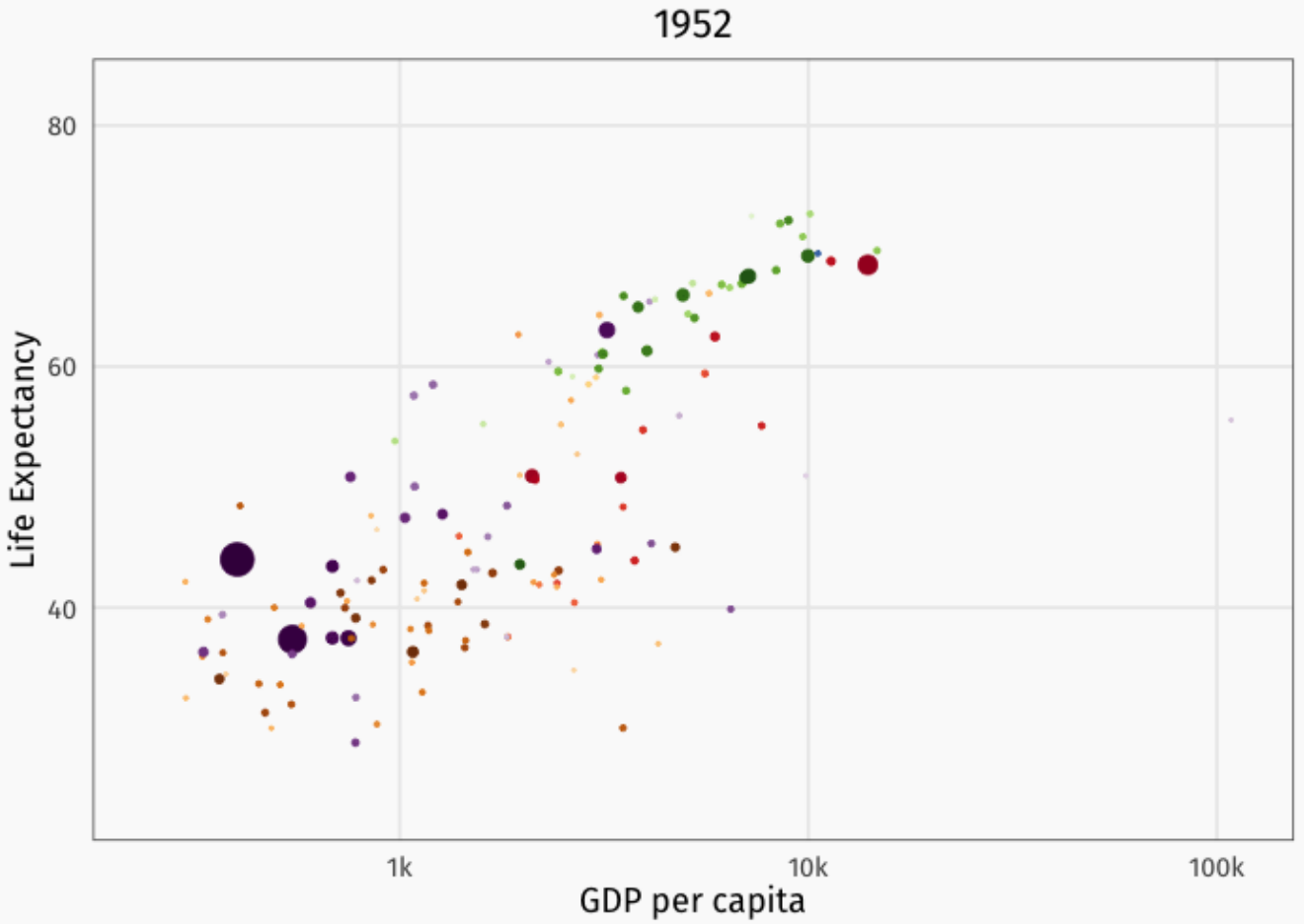
```
# library(devtools)
# install_github("thomasp85/gganimate")
library(gganimate)

# Same plot without facet_wrap()
g_hra +
  transition_states(year, 1, 0) +
  ggtitle("{closest_state}")
```

Special Bonus: Animated!

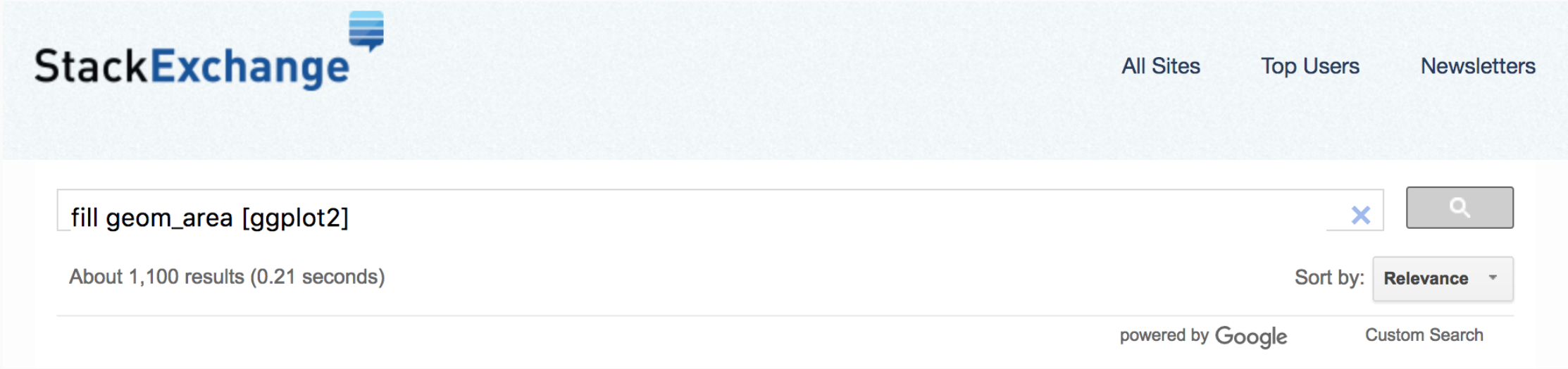
```
# library(devtools)
# install_github("thomasp85/gganimate")
library(gganimate)

# Same plot without facet_wrap()
g_hra +
  transition_states(year, 1, 0) +
  ggtitle("{closest_state}")
```



g is for Goodbye

Stack Exchange is Awesome



Stack Exchange is Awesome

1 Answer

active oldest votes

7

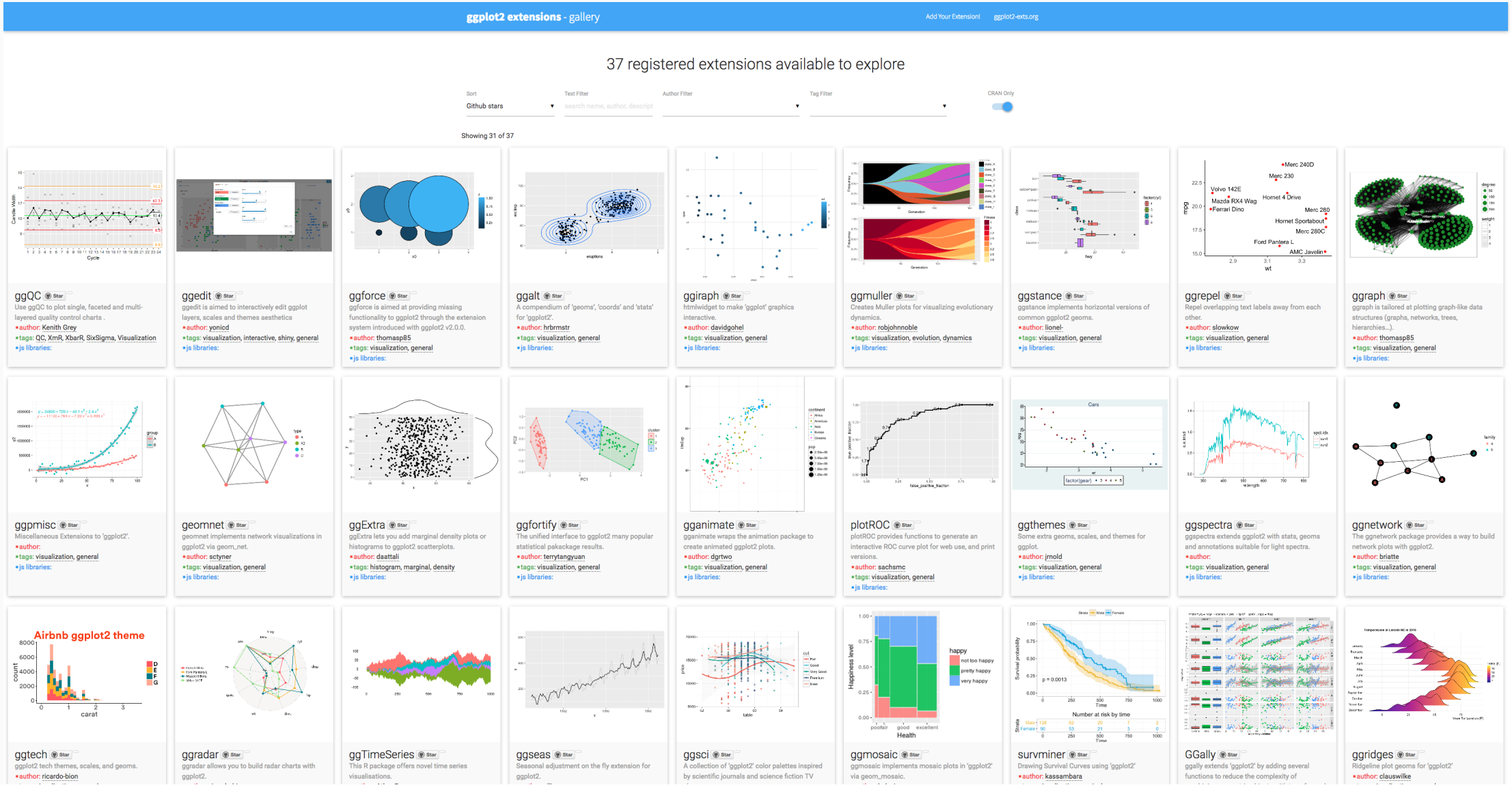
You need to make a new grouping variable for each positive/negative segment. To make the transitions less "blocky", you can just first interpolate the data:

```
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

- **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/

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`

Review

- Slides and code on GitHub: http://github.com/gadenbuie/gentle-ggplot2

Thanks!

@grrrck
github.com/gadenbuie
Garrick Aden-Buie