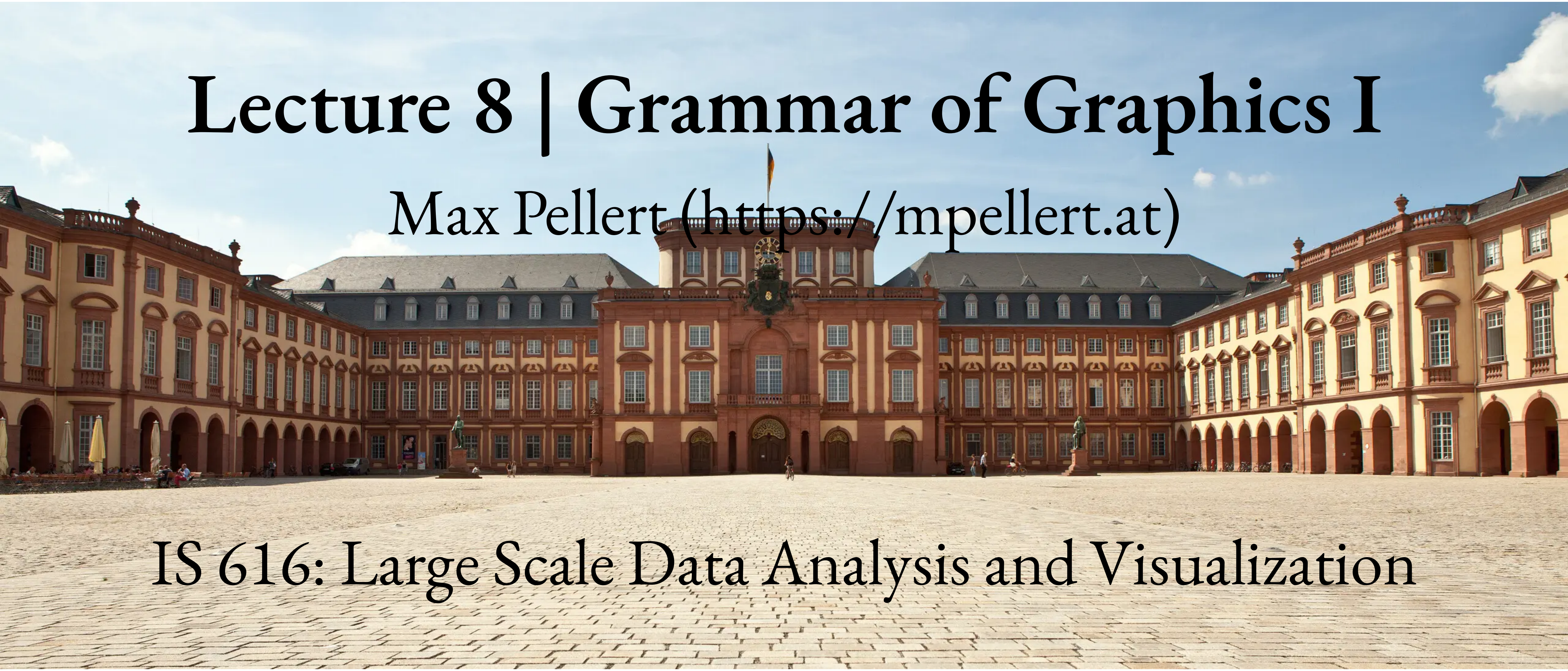
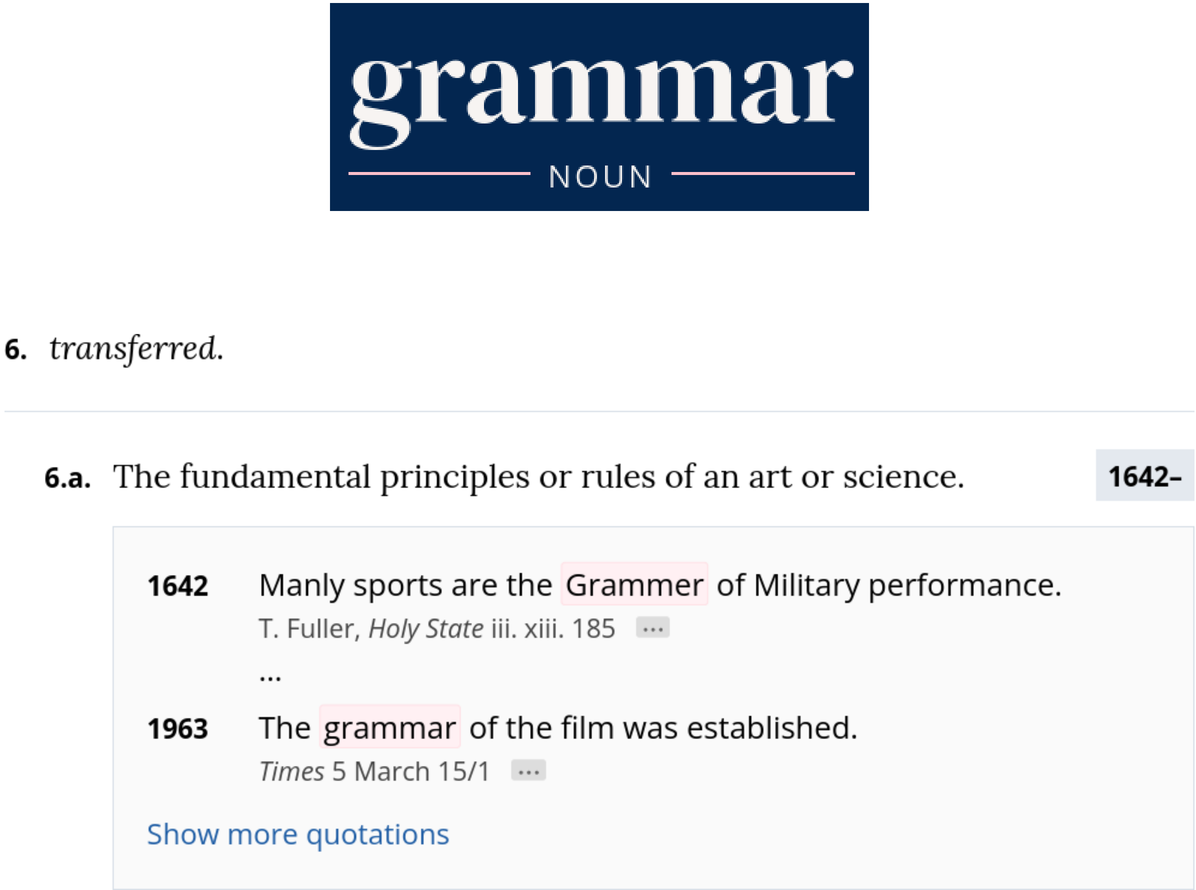


Lecture 8 | Grammar of Graphics I

Max Pellert (https://mpellert.at)

IS 616: Large Scale Data Analysis and Visualization

UNIVERSITY
OF MANNHEIM



grammar

NOUN

6. *transferred.*

6.a. The fundamental principles or rules of an art or science. **1642–**

1642 Manly sports are the Grammer of Military performance.
T. Fuller, *Holy State* iii. xiii. 185 ···

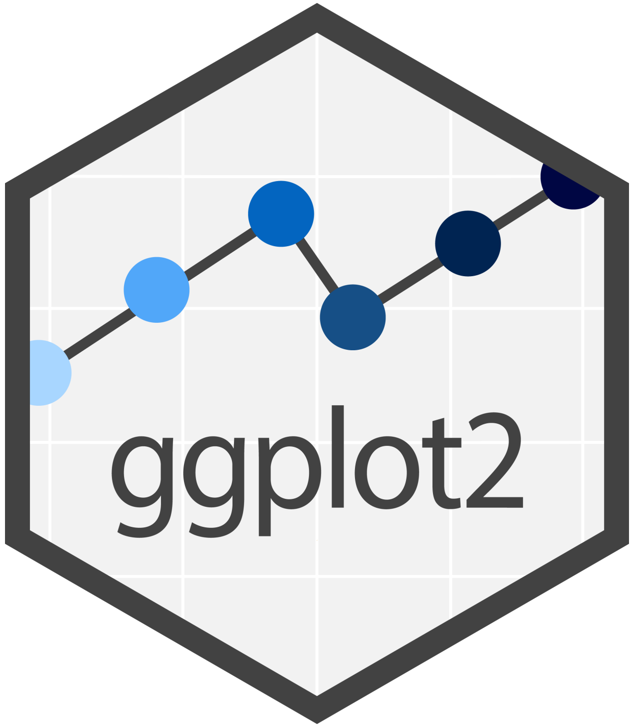
...

1963 The grammar of the film was established.
Times 5 March 15/1 ···

Show more quotations

Oxford English Dictionary, s.v. "grammar, n., sense 6.a", July 2023.
https://doi.org/10.1093/OED/2306046169

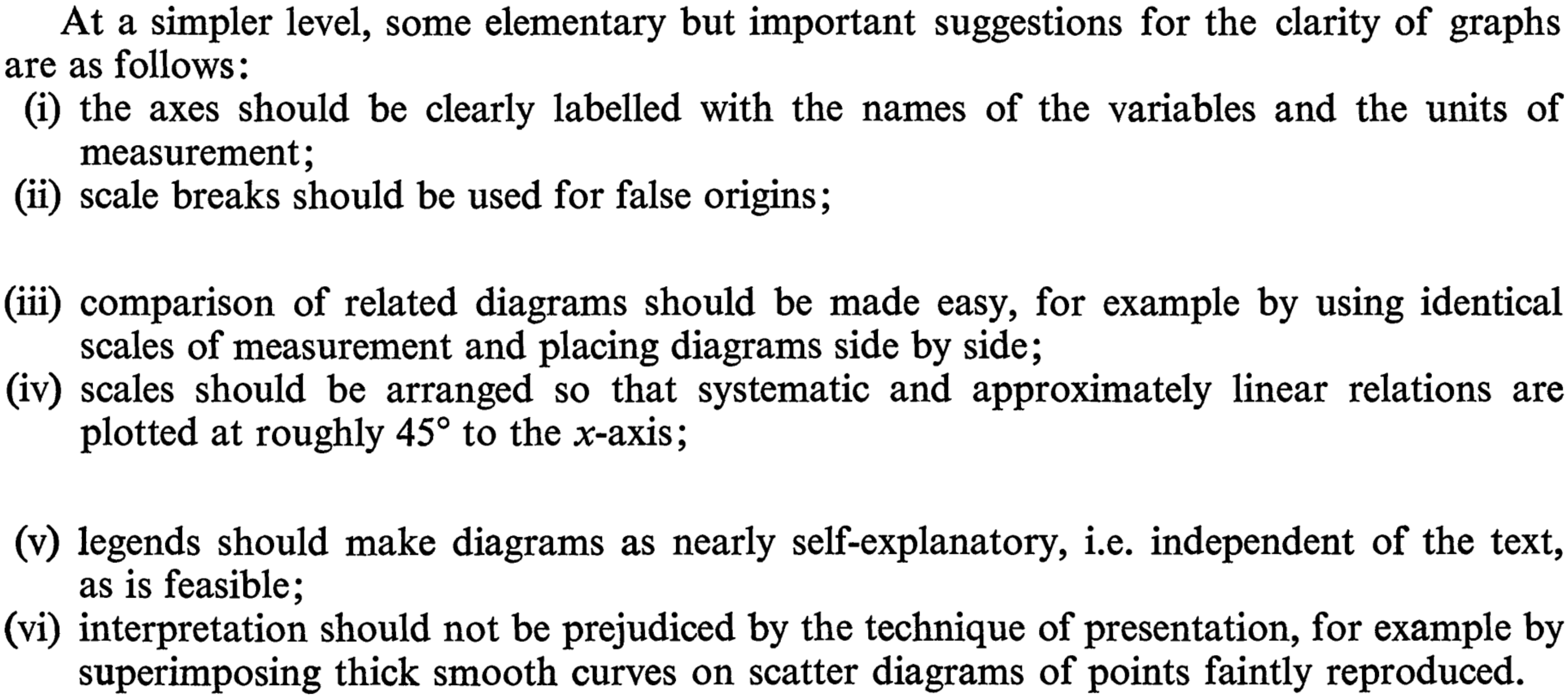
Why *ggplot2*?



https://github.com/erikgahner/awesome-ggplot2

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.

These ideas don’t come out of nowhere



At a simpler level, some elementary but important suggestions for the clarity of graphs are as follows:

(i) the axes should be clearly labelled with the names of the variables and the units of measurement;

(ii) scale breaks should be used for false origins;

(iii) comparison of related diagrams should be made easy, for example by using identical scales of measurement and placing diagrams side by side;

(iv) scales should be arranged so that systematic and approximately linear relations are plotted at roughly 45° to the *x*-axis;

(v) legends should make diagrams as nearly self-explanatory, i.e. independent of the text, as is feasible;

(vi) interpretation should not be prejudiced by the technique of presentation, for example by superimposing thick smooth curves on scatter diagrams of points faintly reproduced.

Cox, D. R. (1978). Some Remarks on the Role in Statistics of Graphical Methods. Applied Statistics, 27(1), 4. https://doi.org/10.2307/2346220

built-in	**ggplot2**
“beginner”	“expert”
“basic”	“advanced”
“easy”	“hard”
“simple”	“complicated”

ggplot2	**built-in**
“beginner”	“expert”
“basic”	“advanced”
“easy”	“hard”
“simple”	“complicated”

Pragmatic reasons

- *Functional* data visualization
  1. Wrangle 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*

“This fits into a general principle I find myself arguing over and over, which is that you should teach your students as you would have wanted to be taught.”

http://varianceexplained.org/r/teach_ggplot2_to_beginners/

How do we express visuals in words?

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

What is a grammar of graphics?

- **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”

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?

Data

```r
ggplot(data)
```

country	1997	2002	20
Canada	30.30584	31.90227	33.390
China	1230.07500	1280.40000	1318.683
United States	272.91176	287.67553	301.139

country	year	pop
Canada	1997	30.30584
China	1997	1230.07500
United States	1997	272.91176
Canada	2002	31.90227

Data

Aesthetics

```
+ aes()
```

Map data to visual elements or parameters

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

Map data to visual elements or parameters

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

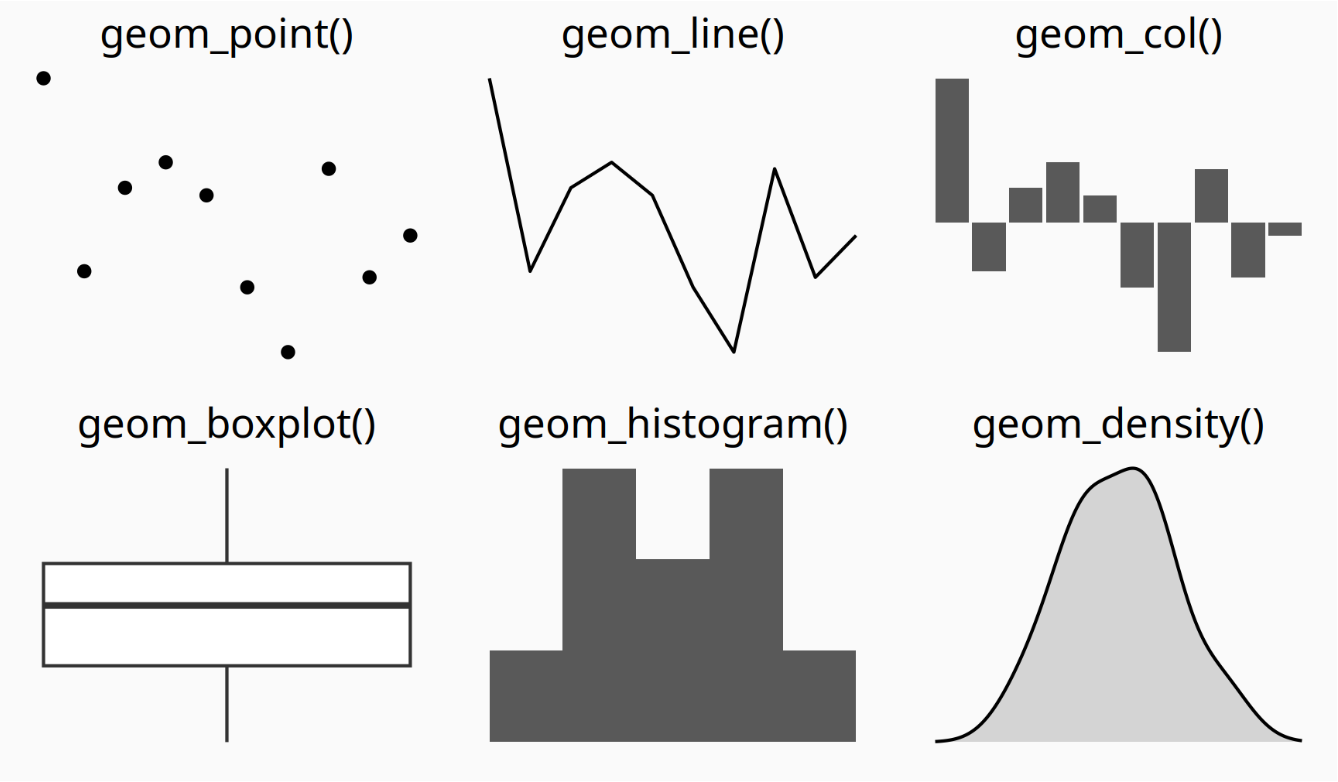
Data

Aesthetics

Geoms

```
+ geom_*()
```

Geometric objects displayed on the plot



See http://ggplot2.tidyverse.org/reference/ for many more options or just start typing `geom_` in RStudio

```
 [1] "geom_abline"            "geom_area"              "geom_bar"
 [4] "geom_bin_2d"            "geom_bin2d"             "geom_blank"
 [7] "geom_boxplot"           "geom_col"               "geom_contour"
[10] "geom_contour_filled"    "geom_count"             "geom_crossbar"
[13] "geom_curve"             "geom_density"           "geom_density_2d"
[16] "geom_density_2d_filled" "geom_density2d"         "geom_density2d_filled"
[19] "geom_dotplot"           "geom_errorbar"          "geom_errorbarh"
[22] "geom_freqpoly"          "geom_function"          "geom_hex"
[25] "geom_histogram"         "geom_hline"             "geom_jitter"
[28] "geom_label"             "geom_line"              "geom_linerange"
[31] "geom_map"               "geom_path"              "geom_point"
[34] "geom_pointrange"        "geom_polygon"           "geom_qq"
[37] "geom_qq_line"           "geom_quantile"          "geom_raster"
[40] "geom_rect"              "geom_ribbon"            "geom_rug"
[43] "geom_segment"           "geom_sf"                "geom_sf_label"
[46] "geom_sf_text"           "geom_smooth"            "geom_spoke"
[49] "geom_step"              "geom_text"              "geom_tile"
[52] "geom_violin"            "geom_vline"
```

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()`

With programming, it’s OK first not to understand what you are doing

Load the libraries:

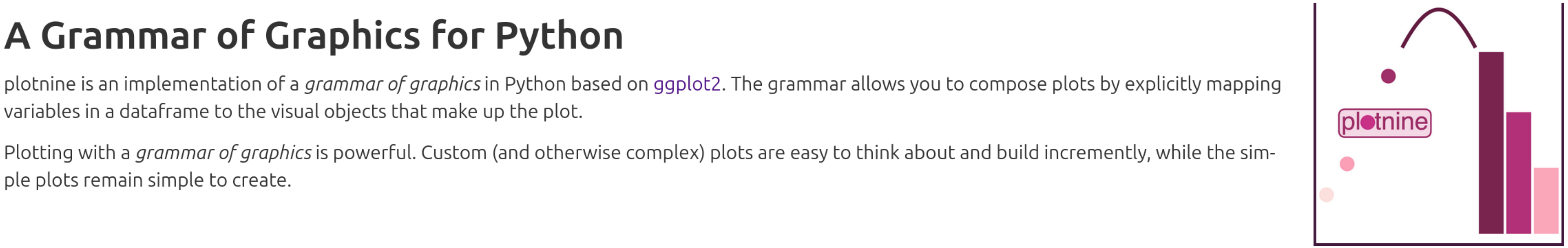
```r
library(gapminder)
library(ggplot2)
library(gganimate)
library(gifski)
library(cowplot)
```

Inspect the data:

```r
head(gapminder)
```

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

What about Python?



A Grammar of Graphics for Python

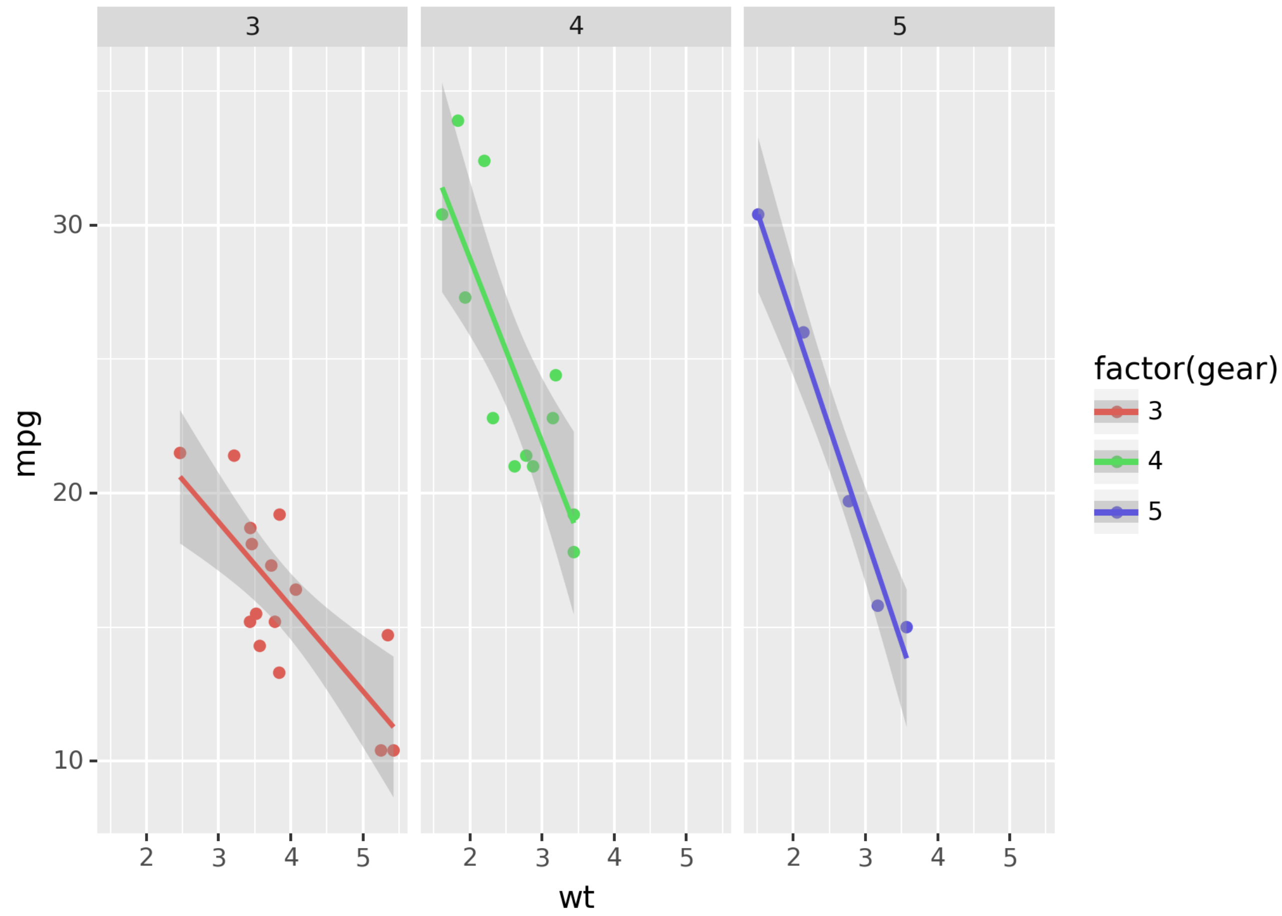
plotnine is an implementation of a *grammar of graphics* in Python based on ggplot2. The grammar allows you to compose plots by explicitly mapping variables in a dataframe to the visual objects that make up the plot.

Plotting with a *grammar of graphics* is powerful. Custom (and otherwise complex) plots are easy to think about and build incremently, while the simple plots remain simple to create.

```
pip install plotnine
```

```python
from plotnine import ggplot, geom_point, aes, stat_smooth, facet_wrap
from plotnine.data import mtcars

print(ggplot(mtcars, aes("wt", "mpg", color="factor(gear)"))
 + geom_point()
 + stat_smooth(method="lm")
 + facet_wrap("~gear"))
```

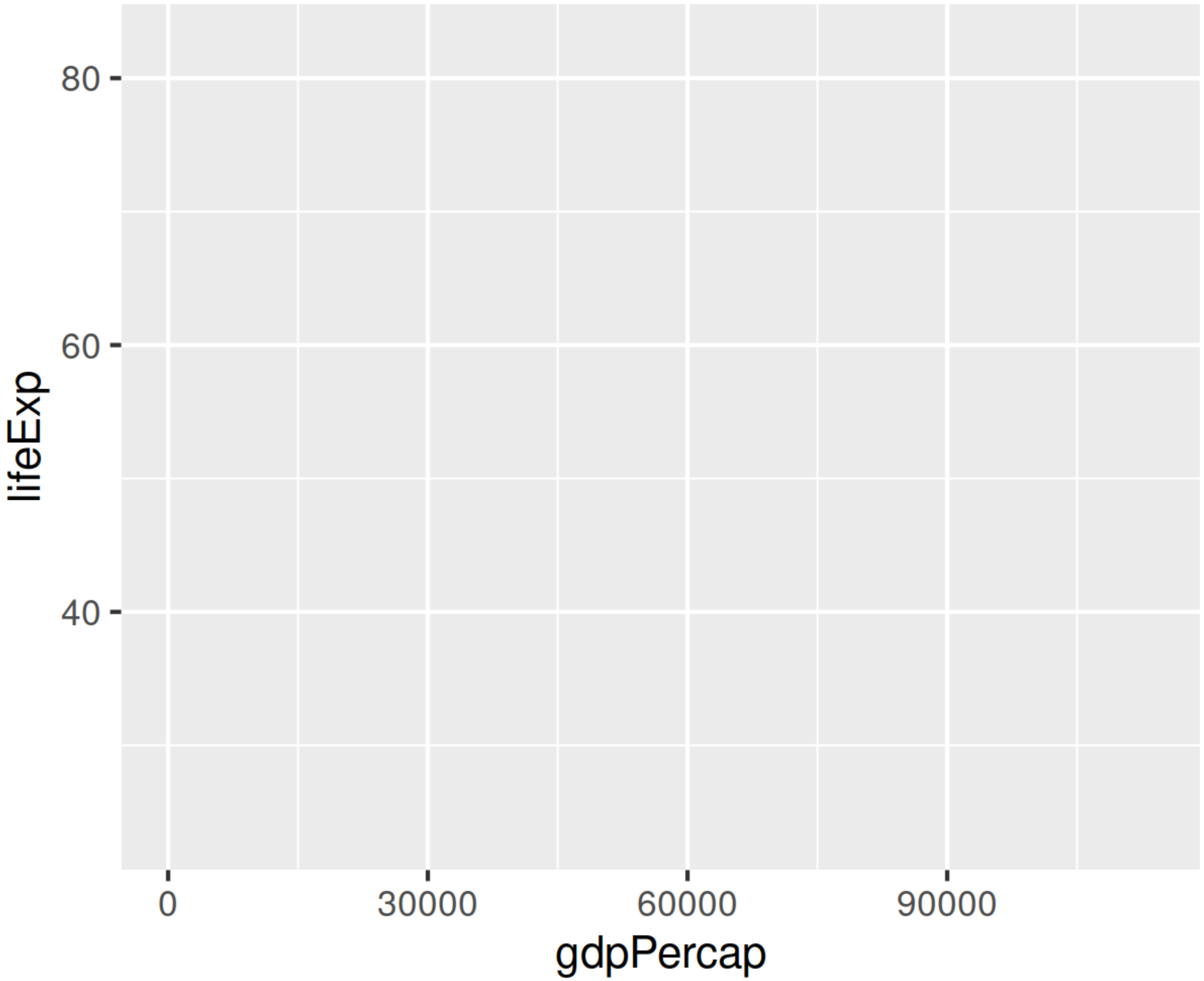


For a different summary of the data frame:

```
Rows: 1,704
Columns: 6
$ country   <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan",
…
$ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia,
…
$ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997,
…
$ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854,
40.8…
$ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372,
12…
$ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134,
…
```

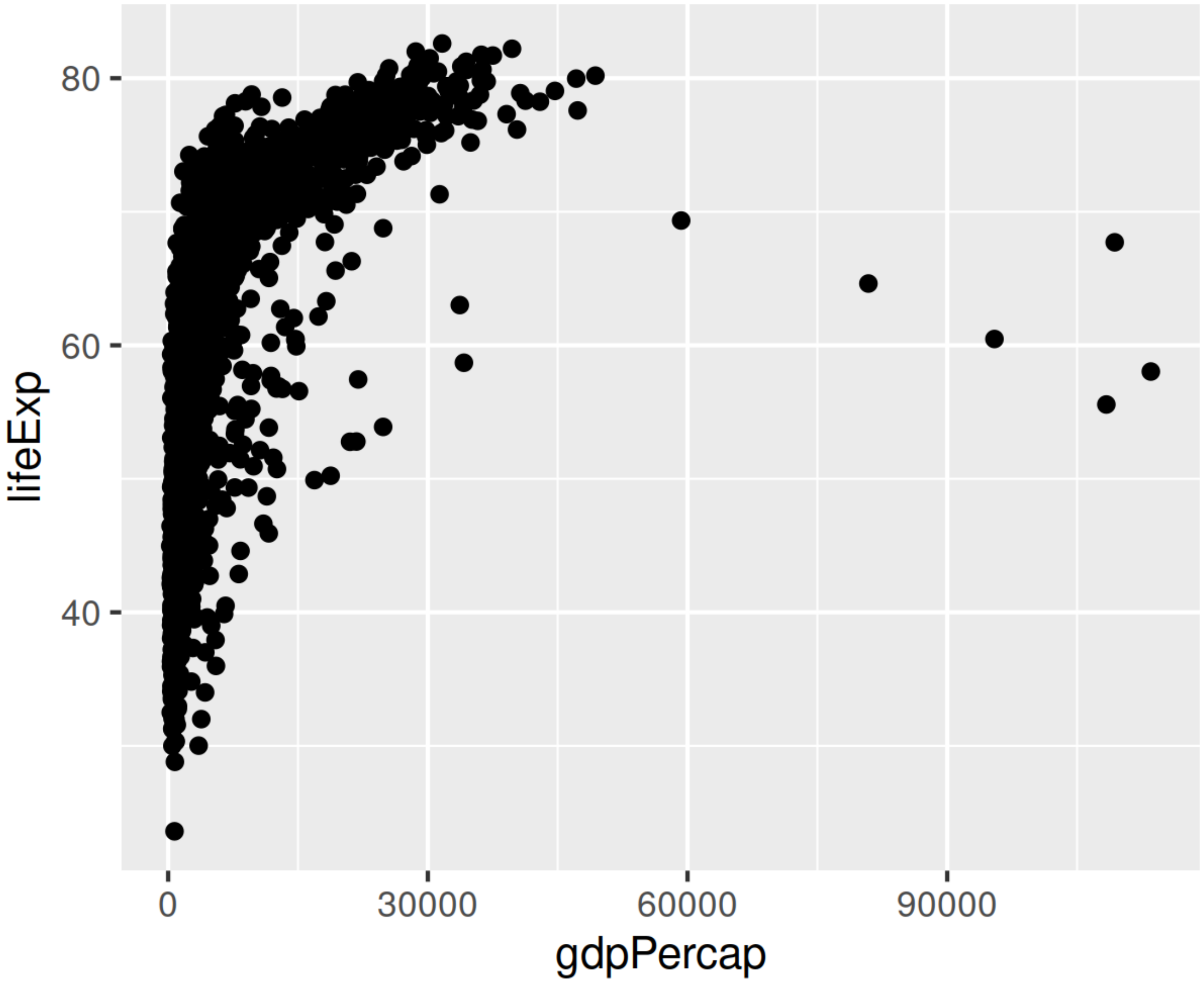
Let’s start with `lifeExp` vs `gdpPercap`

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

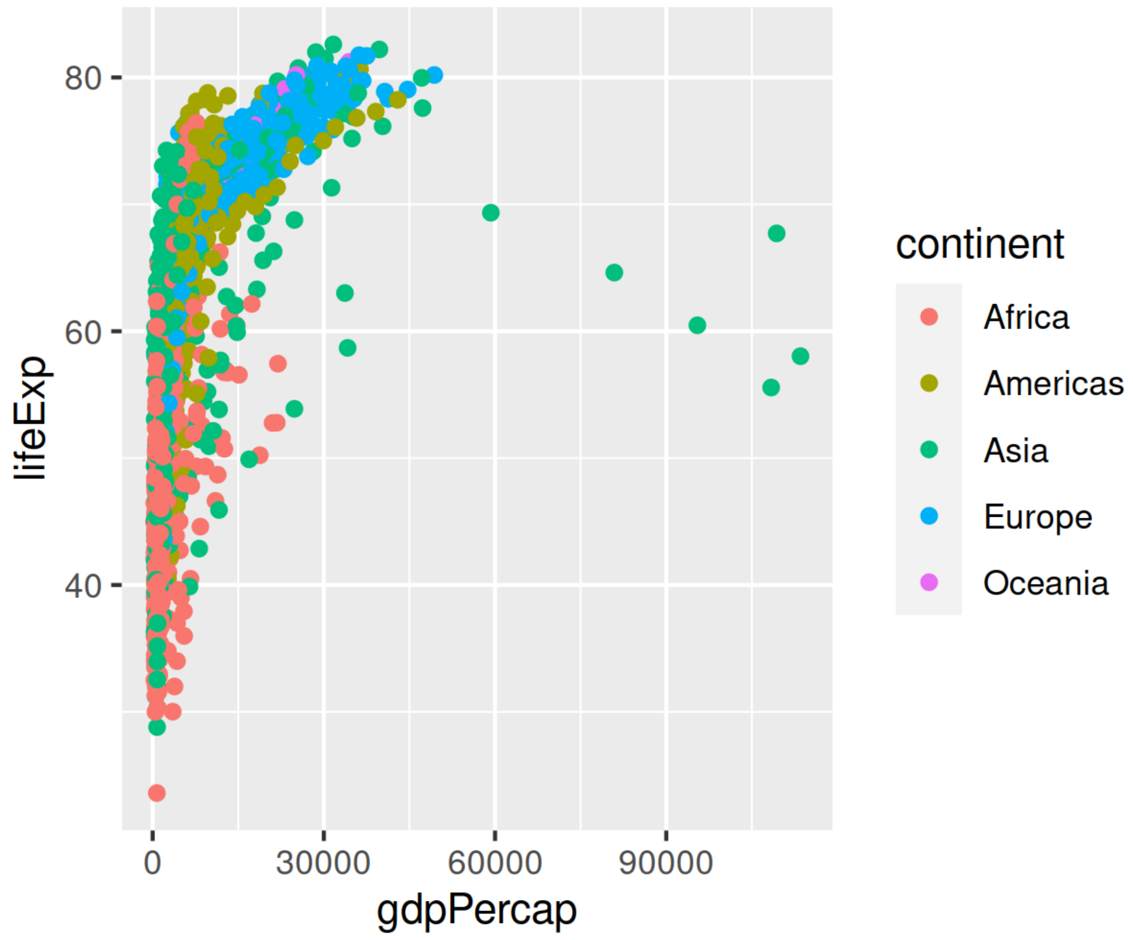


```
ggplot(gapminder) +
  aes(x = gdpPercap,
      y = lifeExp) +
  geom_point()
```

How can I tell countries apart?

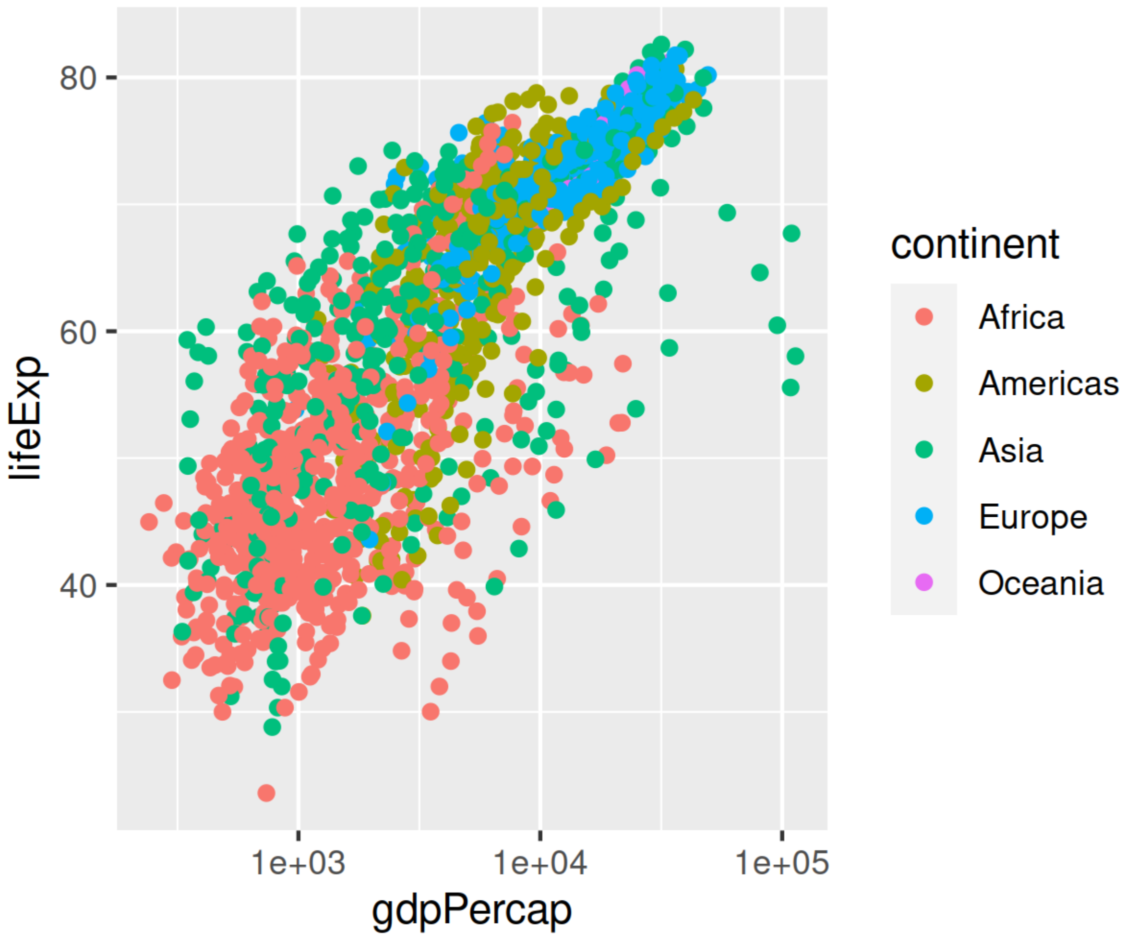


```r
ggplot(gapminder) +
  aes(x = gdpPercap,
      y = lifeExp,
      color = continent) +
  geom_point()
```



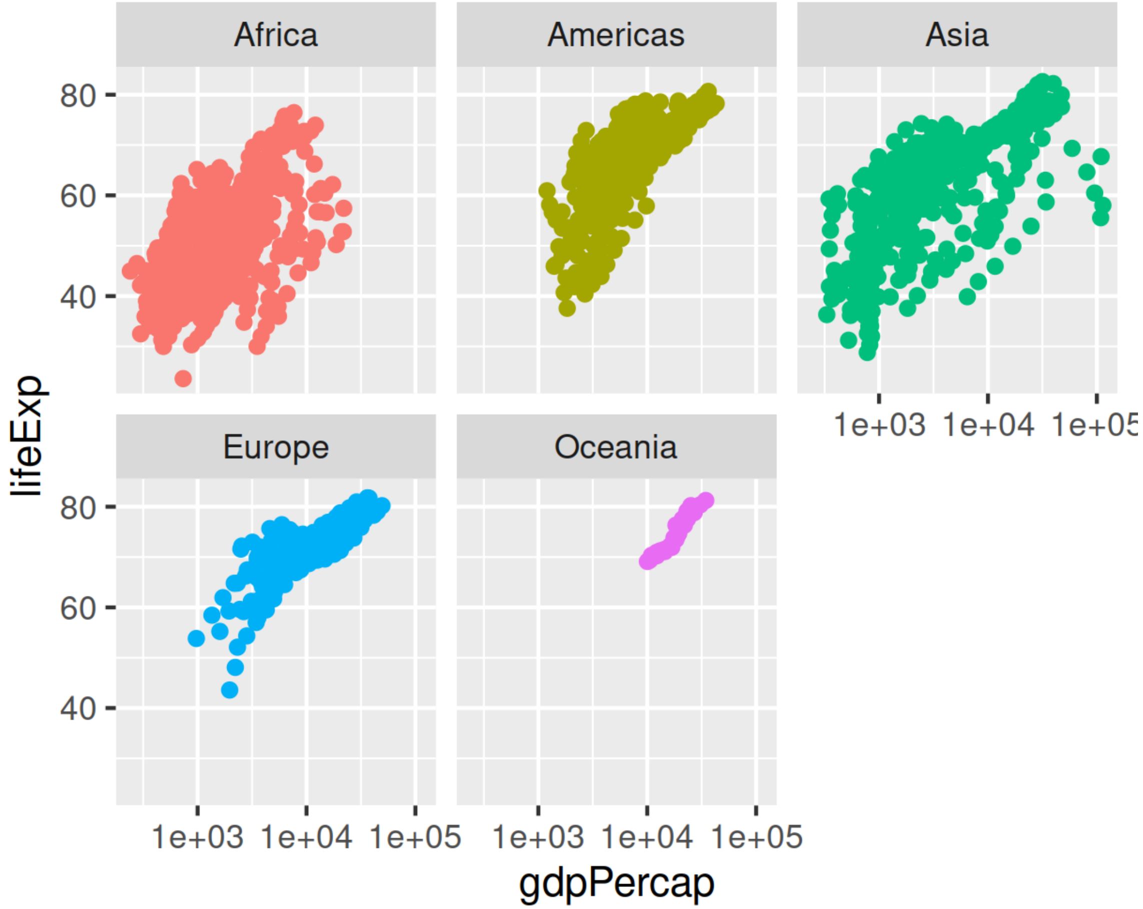
GDP is squished together on the left

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



Still lots of overlap in the countries

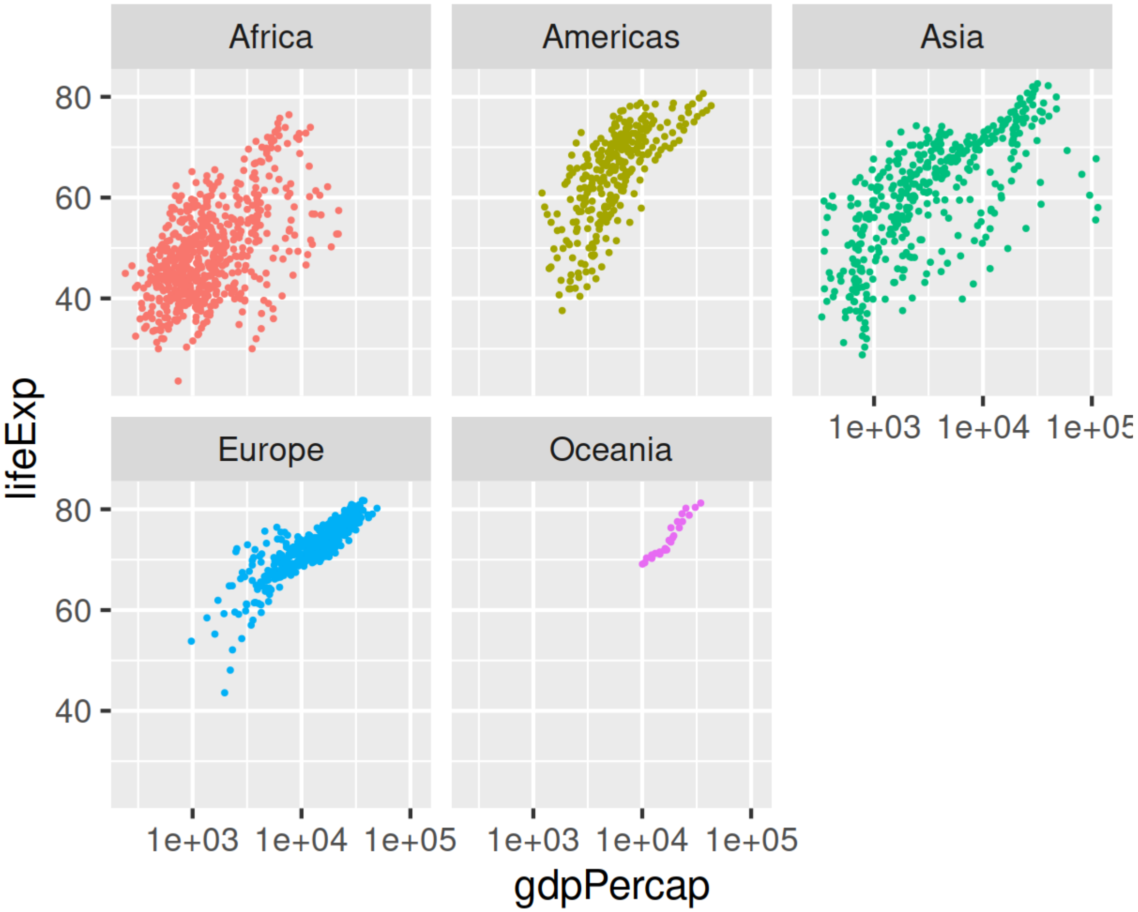
```r
ggplot(gapminder) +
  aes(x = gdpPercap,
      y = lifeExp,
      color = continent) +
  geom_point() +
  scale_x_log10() +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```



No need for color legend thanks to facet titles

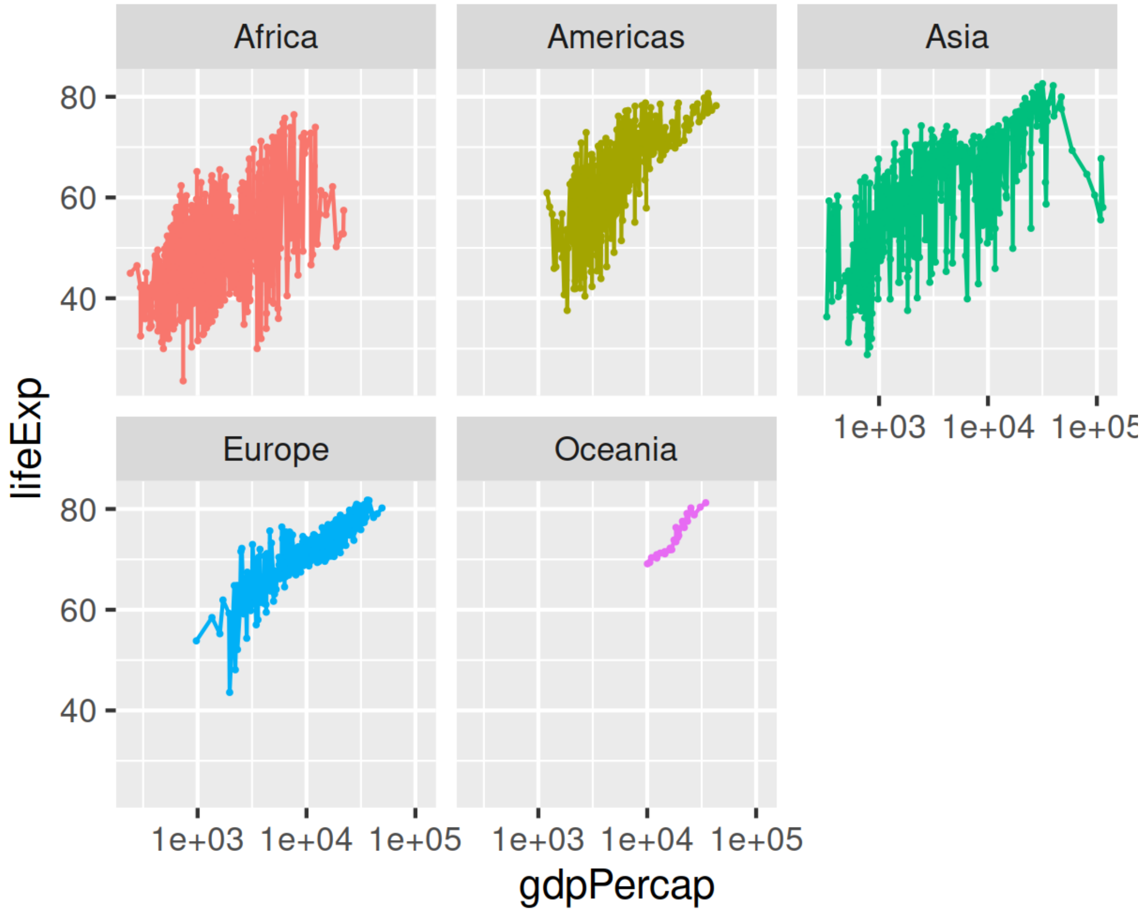
Lots of overplotting due to point size

```r
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?

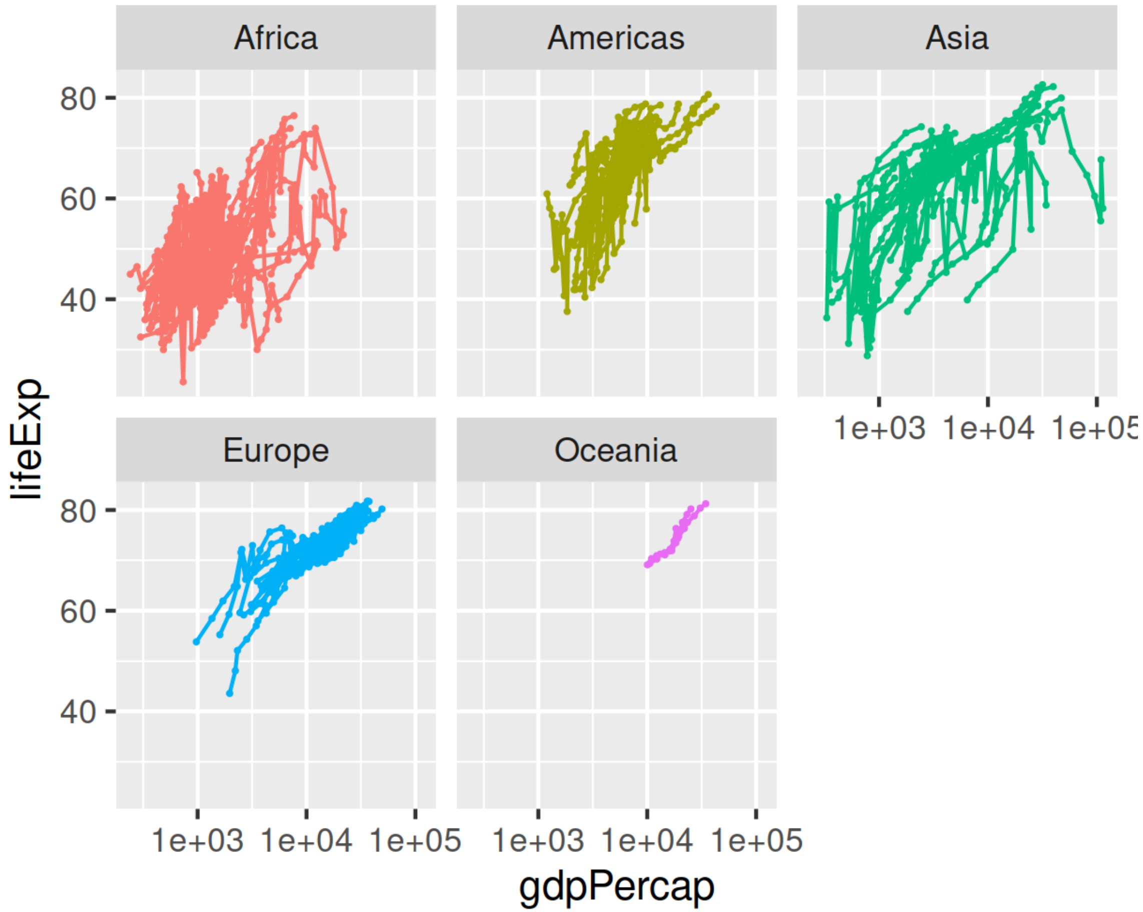
```r
ggplot(gapminder) +
  aes(x = gdpPercap,
      y = lifeExp,
      color = continent) +
  geom_line() +
  geom_point(size=0.25) +
  scale_x_log10() +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```



That line just connected all of the points sequentially...

UNIVERSITY
OF MANNHEIM

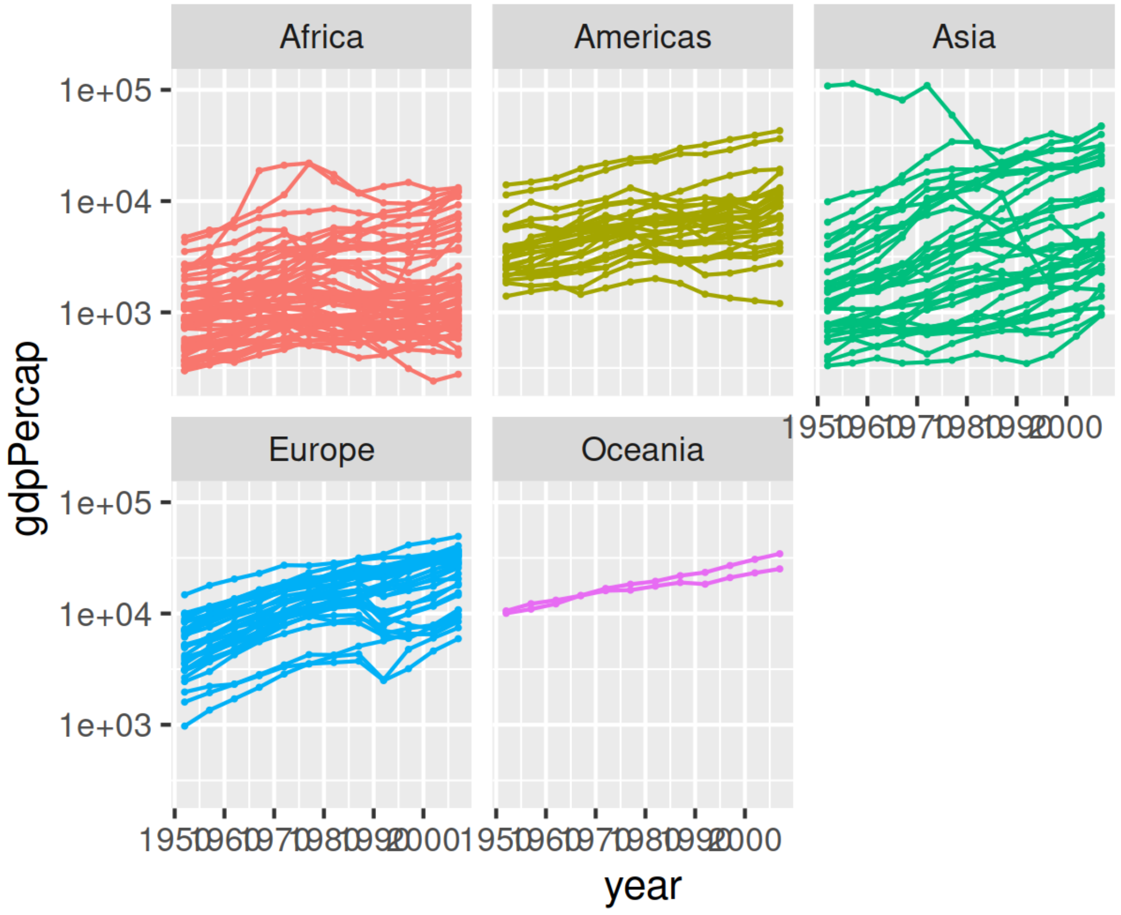
```r
ggplot(gapminder) +
  aes(x = gdpPercap,
      y = lifeExp,
      color = continent) +
  geom_line(
    aes(group = country)
  ) +
  geom_point(size=0.25) +
  scale_x_log10() +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```



We need time on the x-axis!

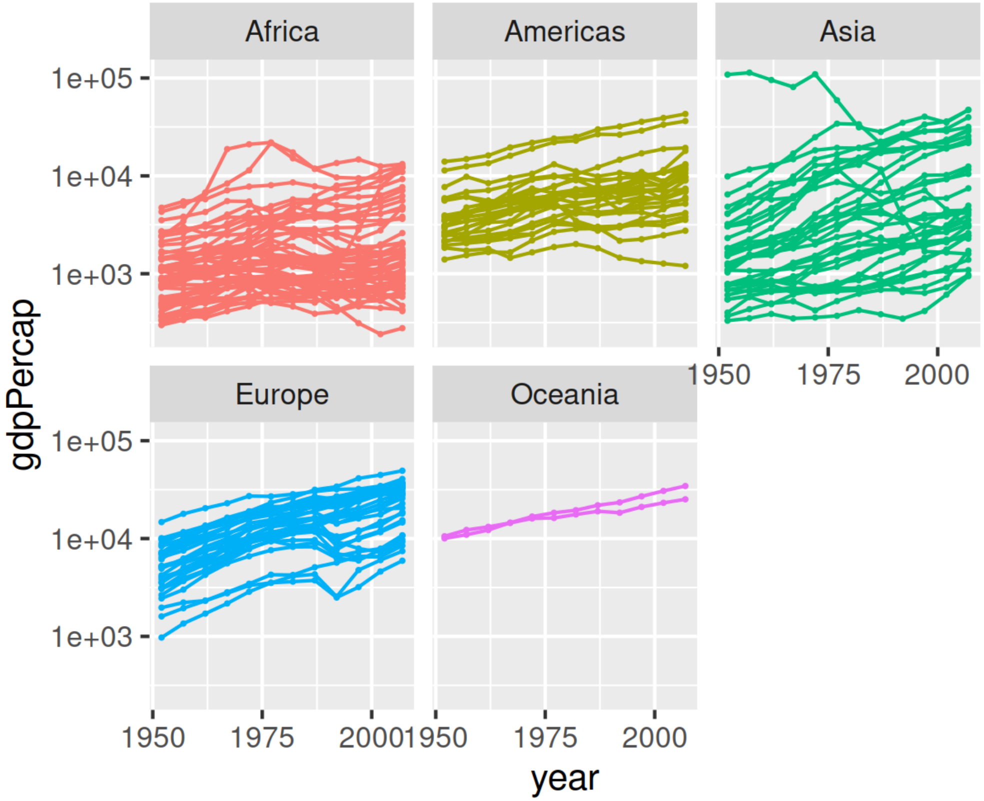
UNIVERSITY
OF MANNHEIM

```r
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, fix that

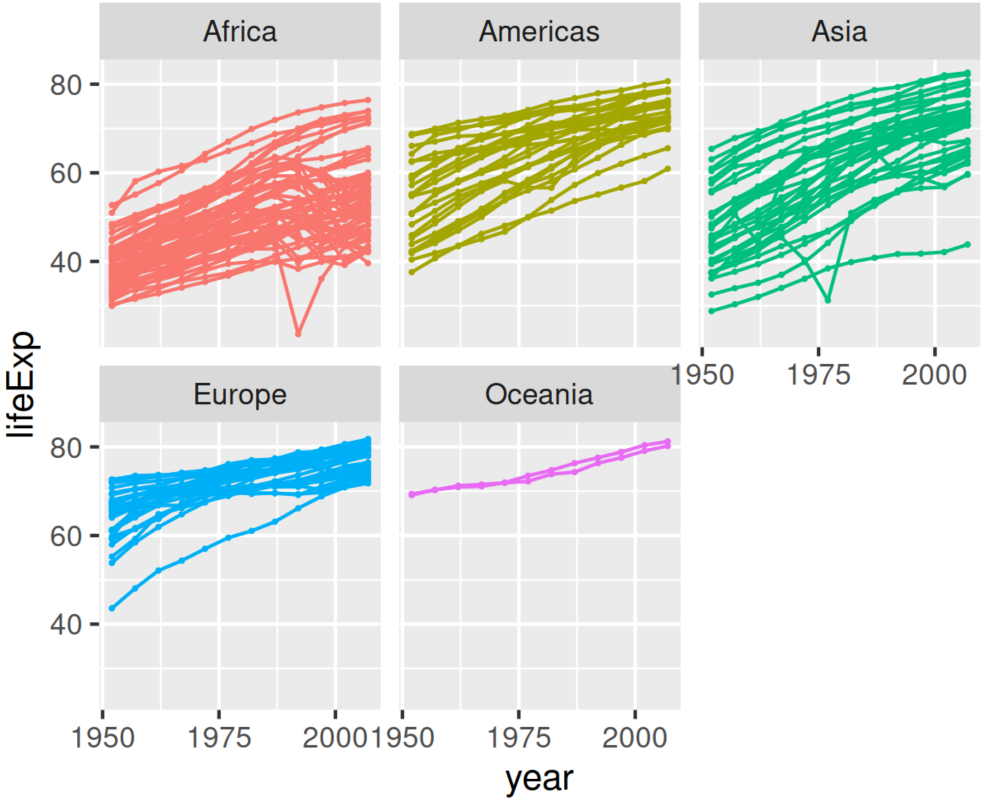
```r
ggplot(gapminder) +
  aes(x = year,
      y = gdpPercap,
      color = continent) +
  geom_point(size=0.25) +
  geom_line(
    aes(group = country)
  ) +
  scale_y_log10() +
  scale_x_continuous(
    breaks = seq(1950, 2000, 25)
  ) +
  facet_wrap(~ continent) +
  guides(color = FALSE)
```



What about life expectancy?

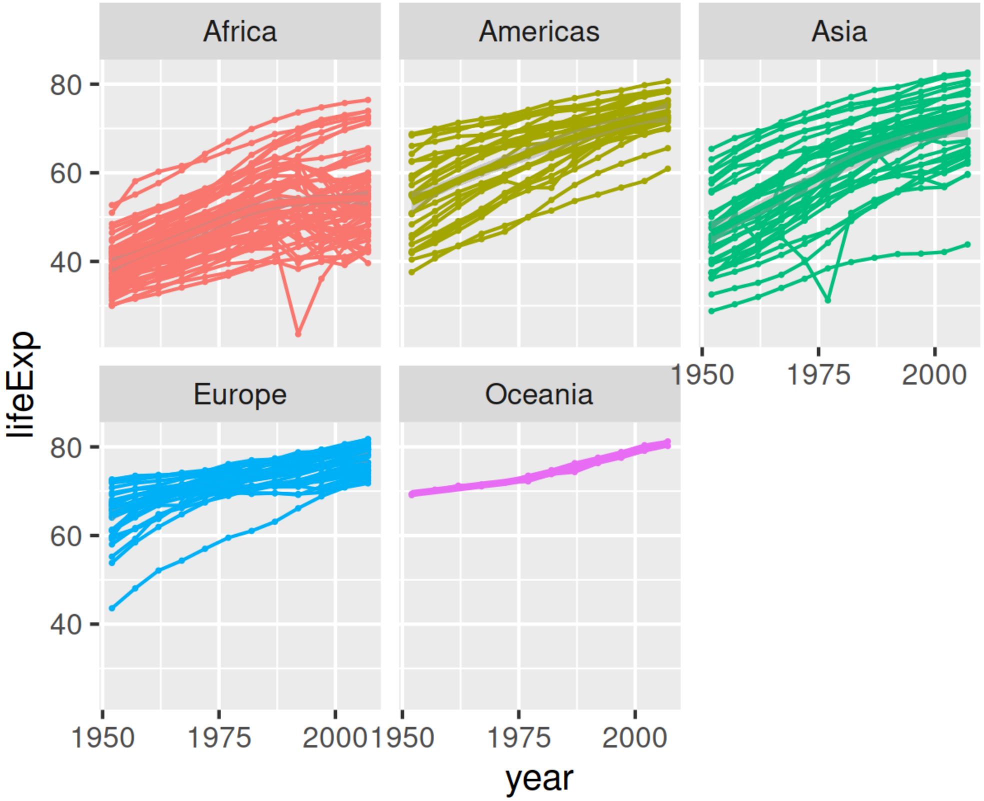
UNIVERSITY OF MANNHEIM

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



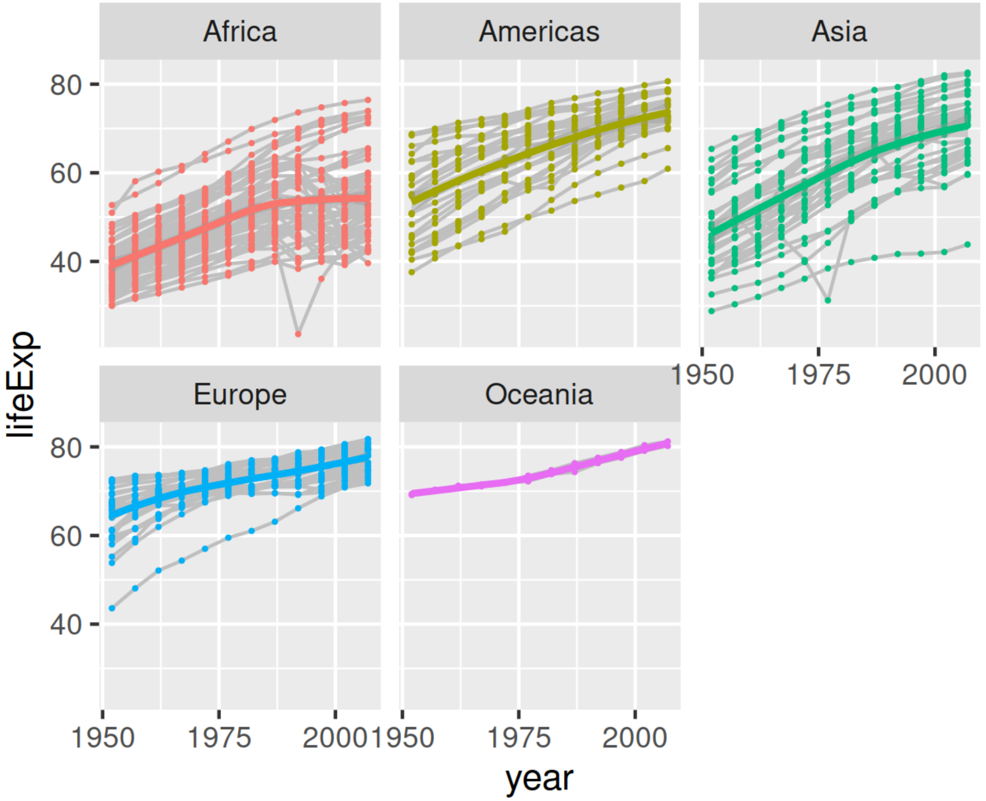
Let’s add a trend line

```r
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)
```



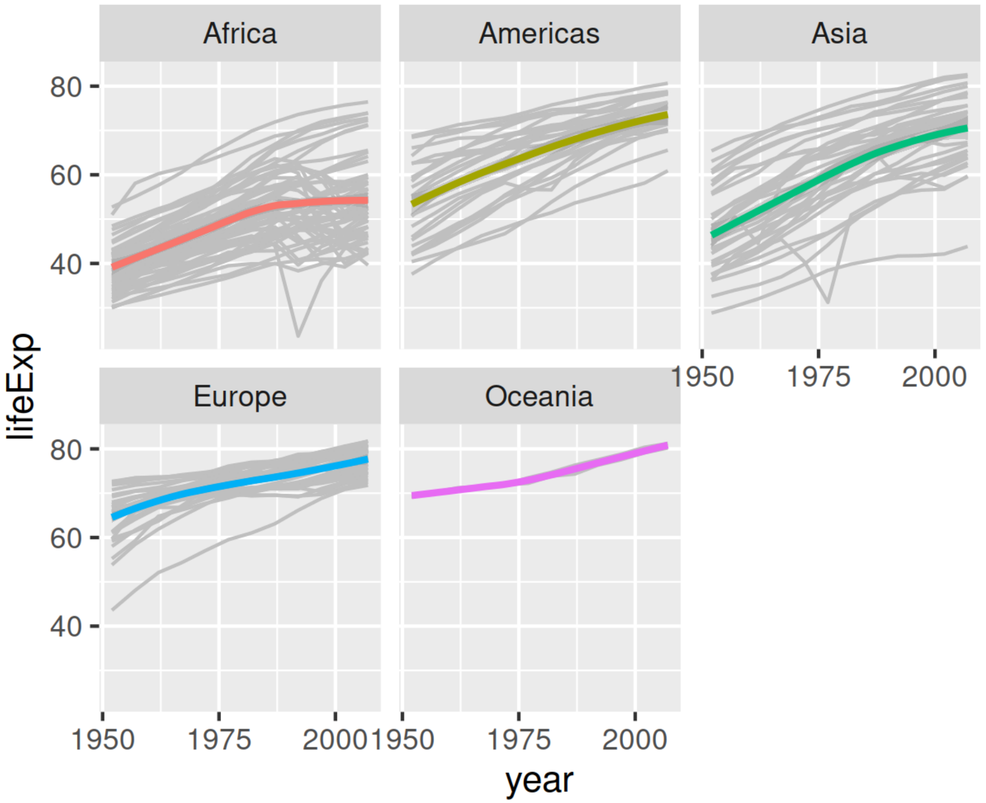
De-emphasize individual countries

```r
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)
```



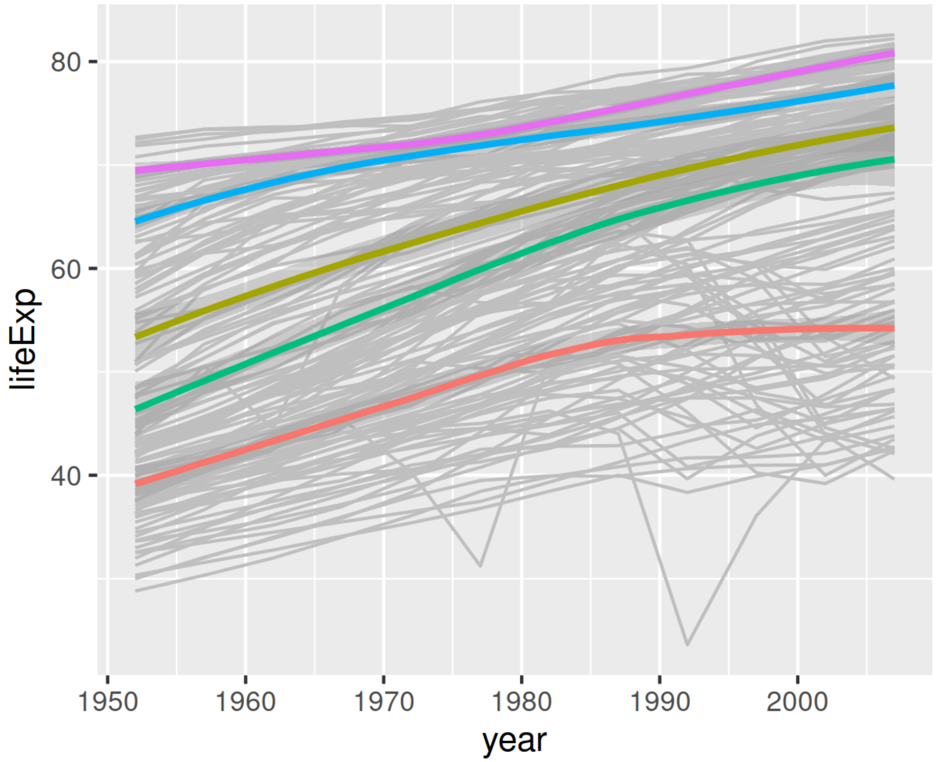
Points are still in the way

```r
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)
```



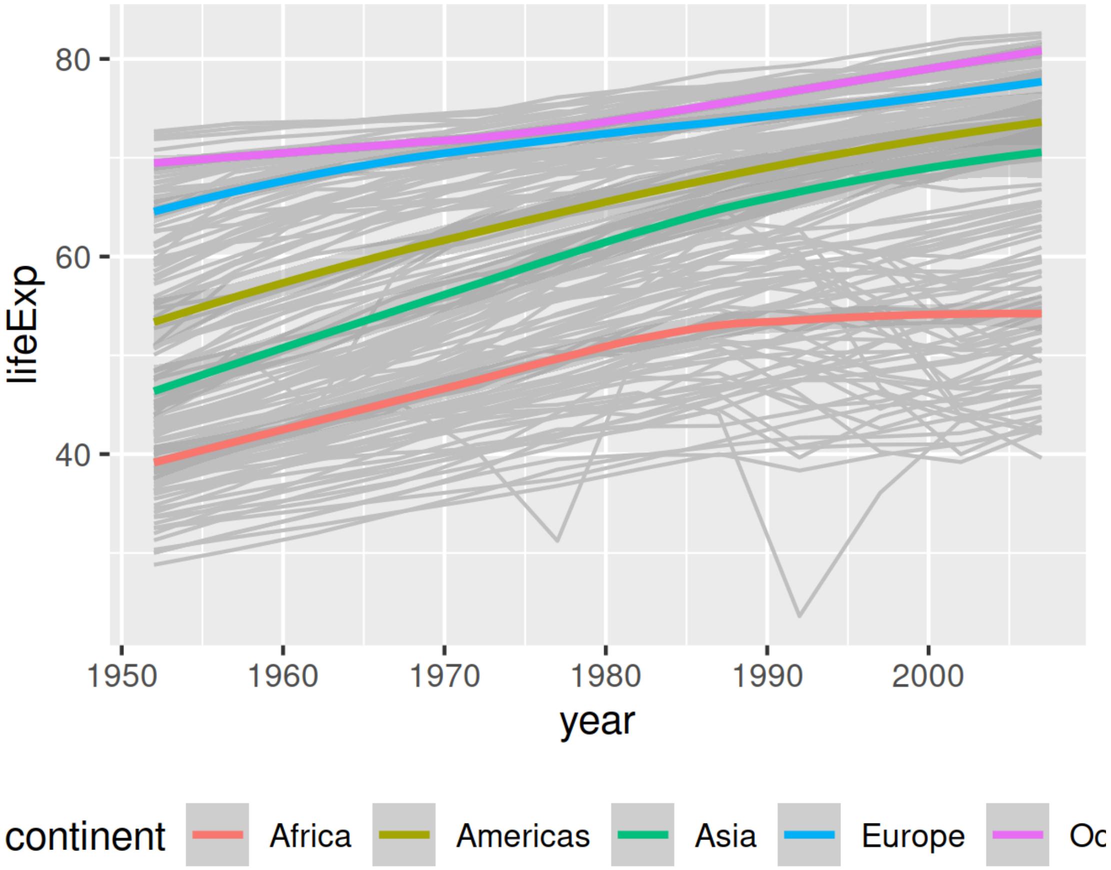
Let’s compare continents

```r
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)
```



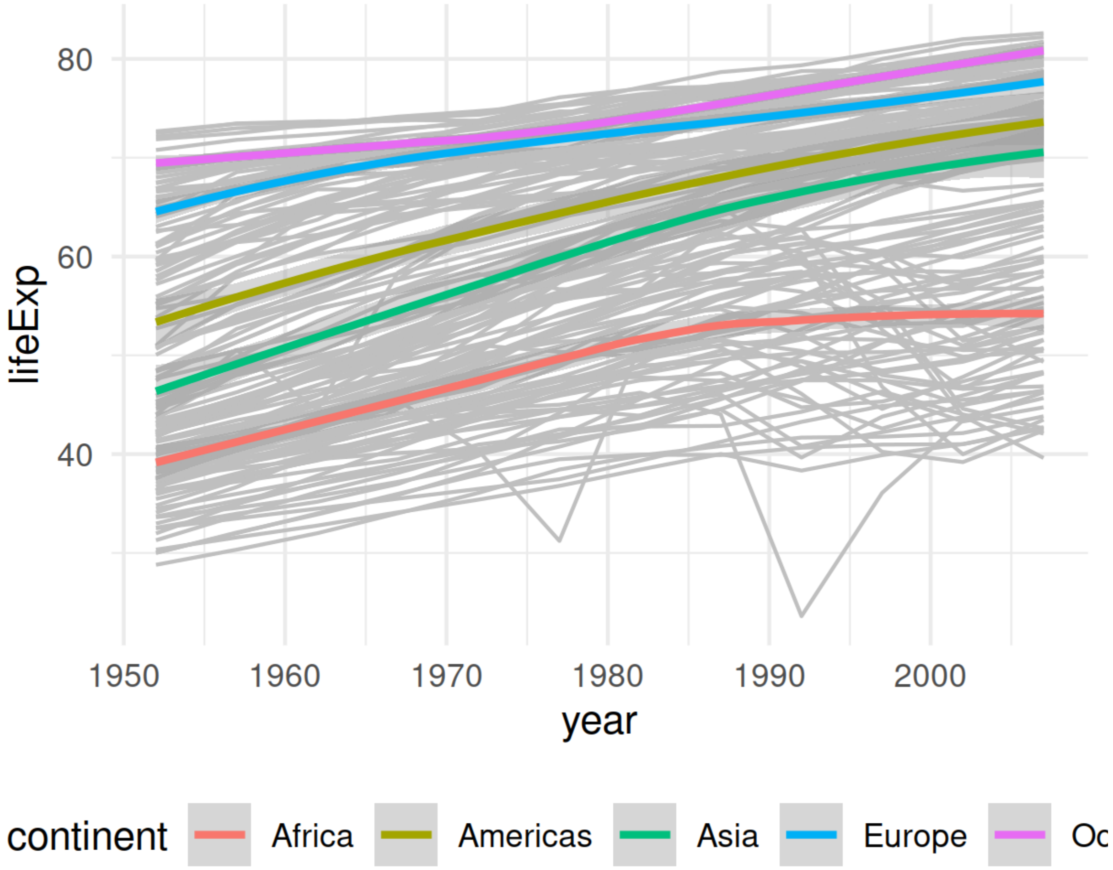
Wait, what color is each continent?

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



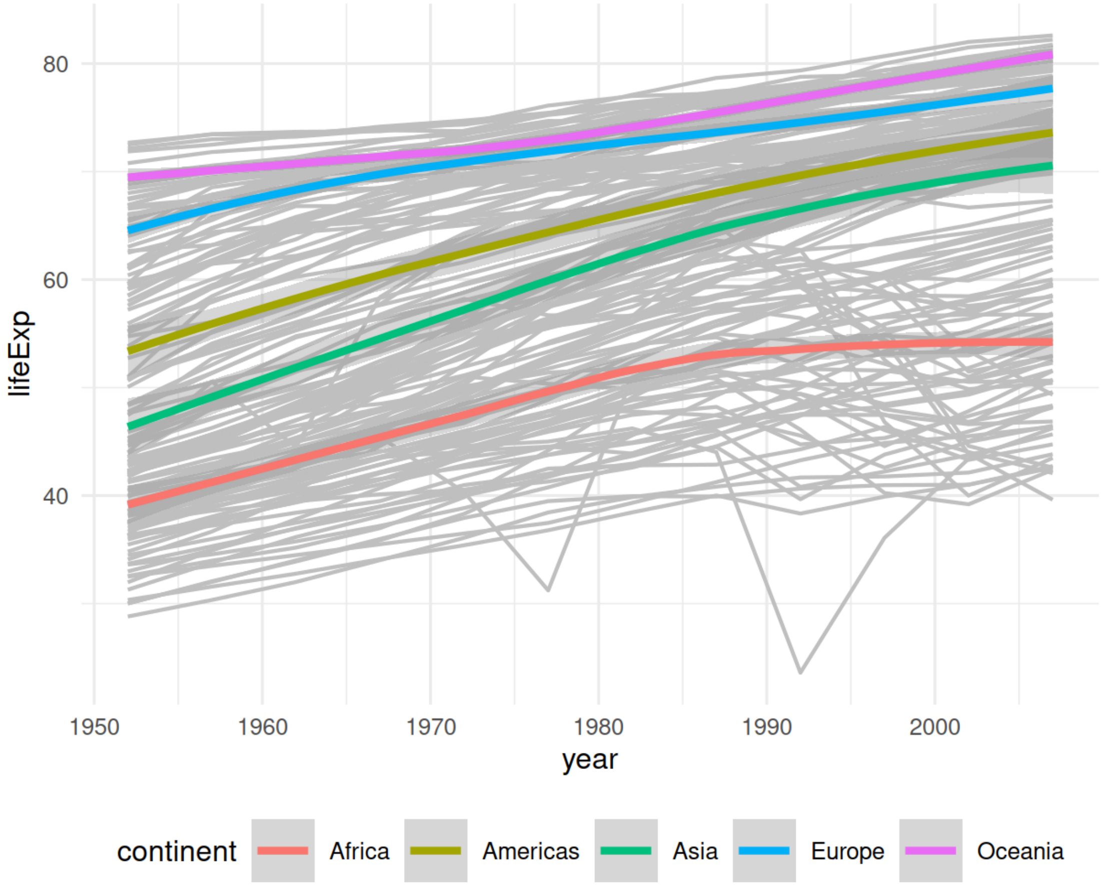
Let’s try the minimal theme

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



Fonts get cut off because they are too big

```r
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, but what about different population size?

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

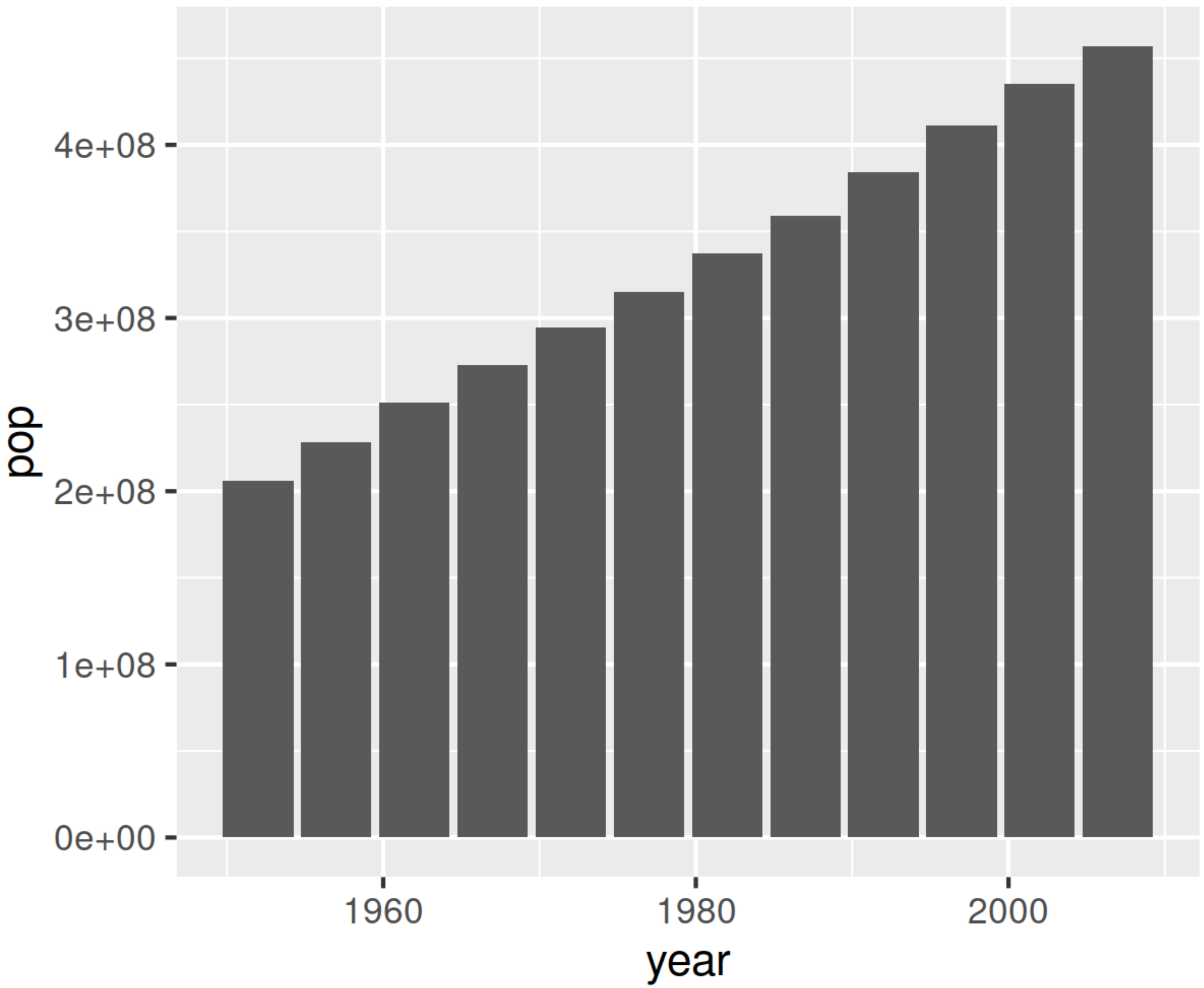
```
# A tibble: 6 × 6
  country continent  year lifeExp      pop
gdpPercap
  <fct>   <fct>     <int>   <dbl>    <int>
<dbl>
1 Canada  Americas   1952    68.8 14785584
11367.
2 Canada  Americas   1957    70.0 17010154
12490.
3 Canada  Americas   1962    71.3 18985849
13462.
4 Canada  Americas   1967    72.1 20819767
16077.
5 Canada  Americas   1972    72.9 22284500
18971.
6 Canada  Americas   1977    74.2 23796400
22091.
```

Let’s look at four countries in more detail.

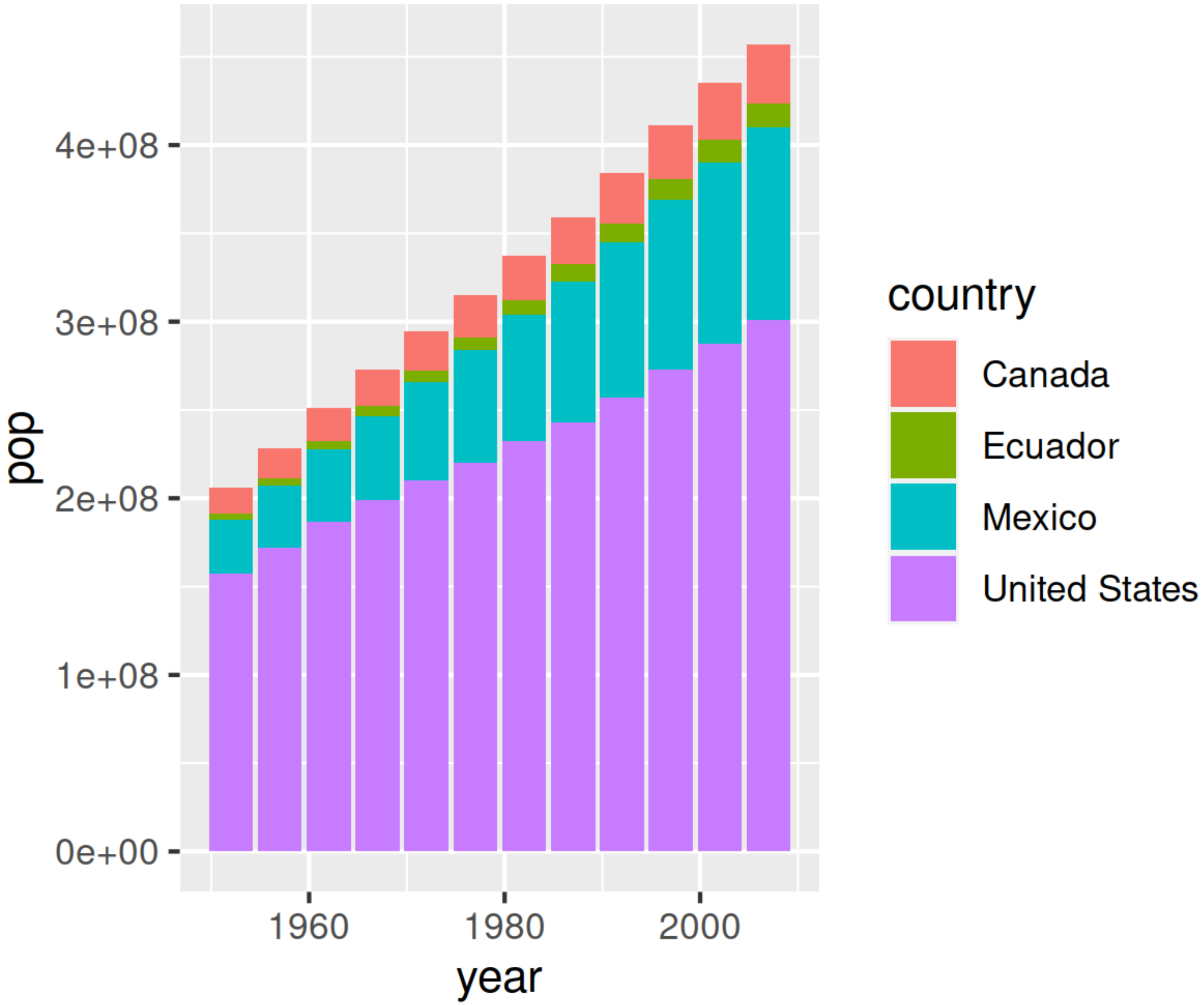
How do their populations compare to each other?

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

But how many people are in each country?



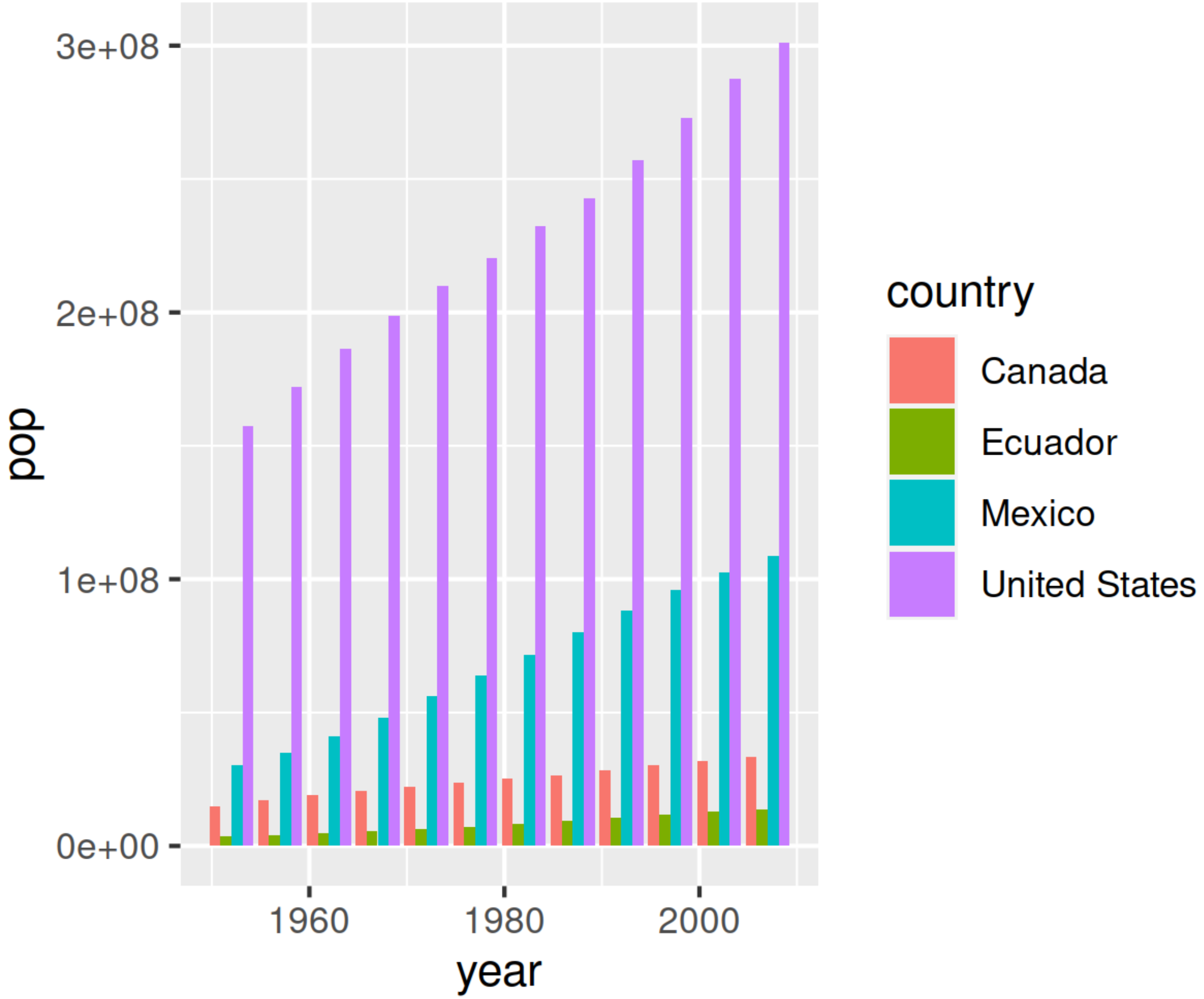
```r
ggplot(americas) +
  aes(
    x = year,
    y = pop,
    fill = country
  ) +
  geom_col()
```



Bars are “stacked”, how to separate them?

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

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

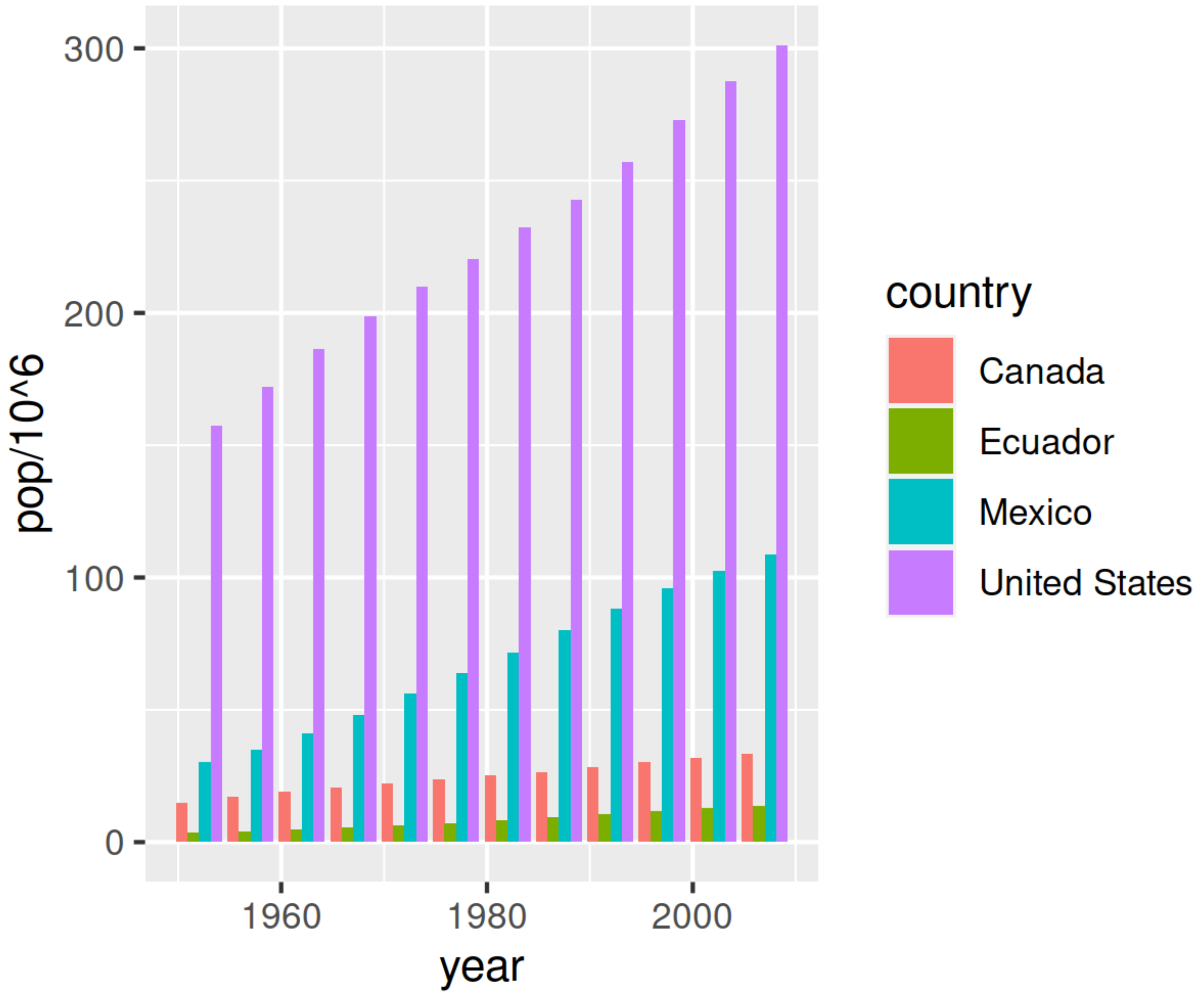


What is scientific notation anyway?

UNIVERSITY OF MANNHEIM

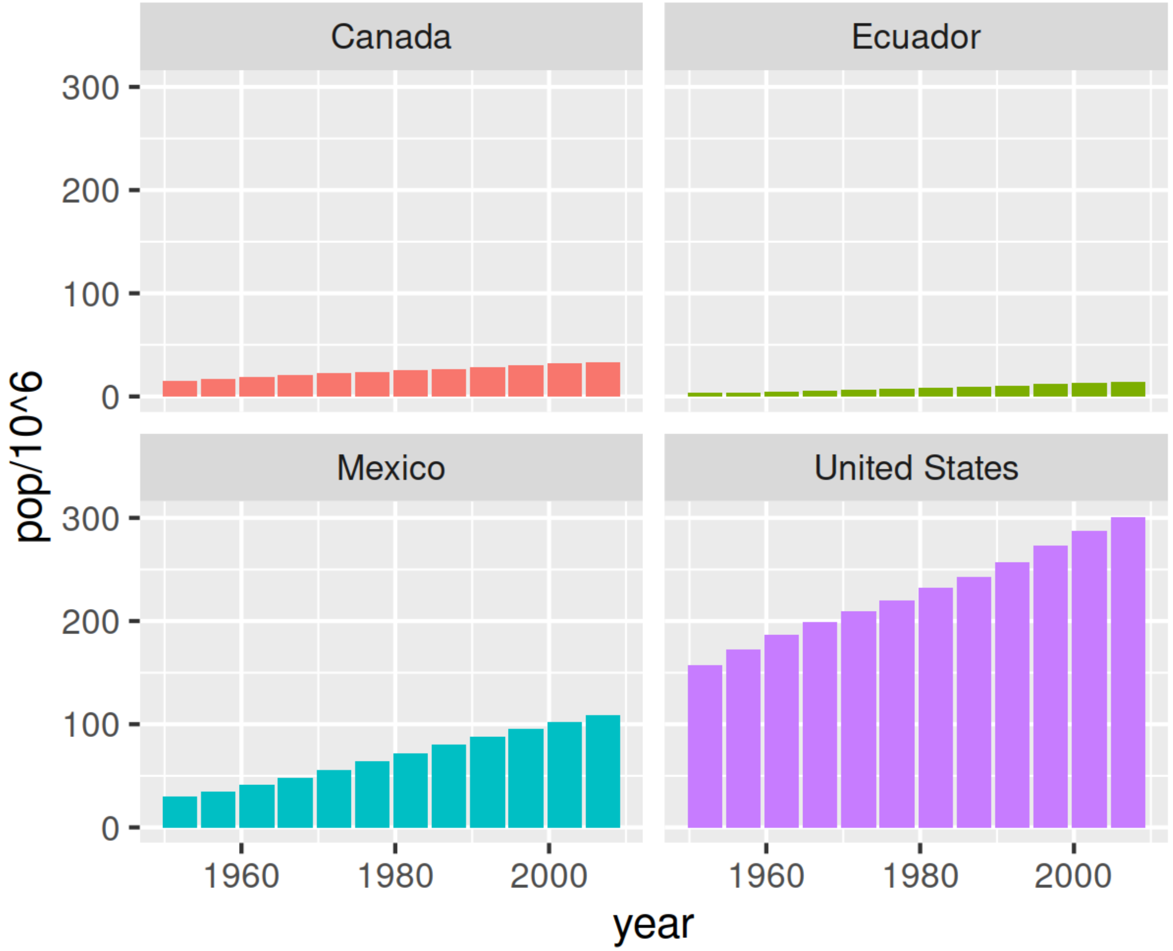
```r
ggplot(americas) +
  aes(
    x = year,
    y = pop / 10^6,
    fill = country
  ) +
  geom_col(
    position = "dodge"
  )
```

ggplot aesthetics can take expressions!



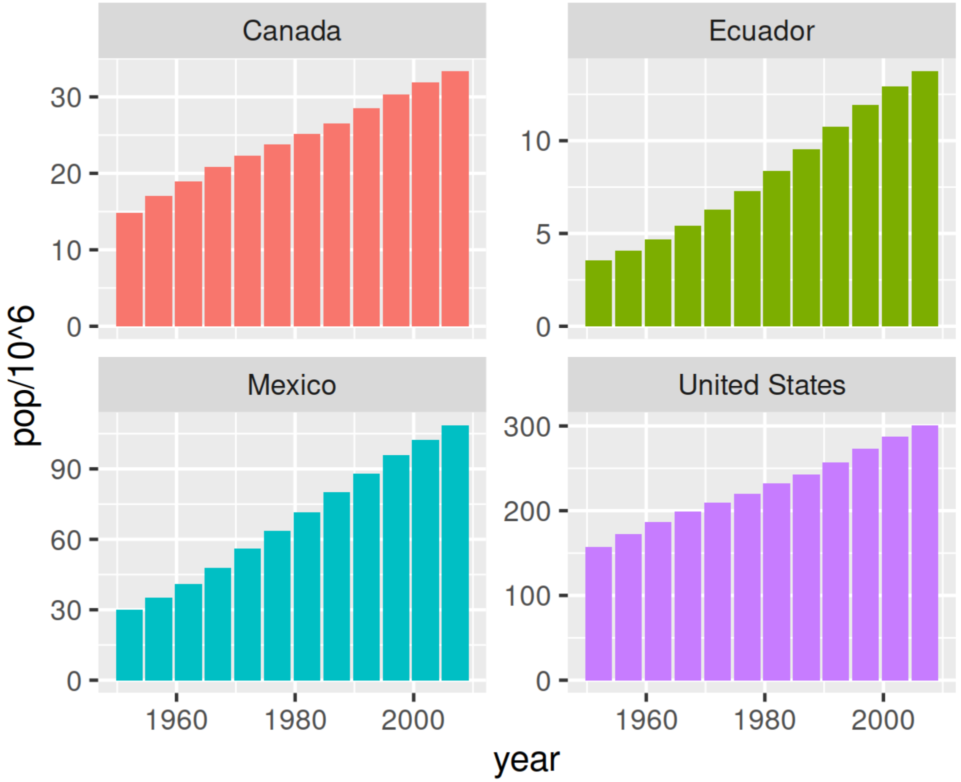
Might be easier to see countries individually

```r
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

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

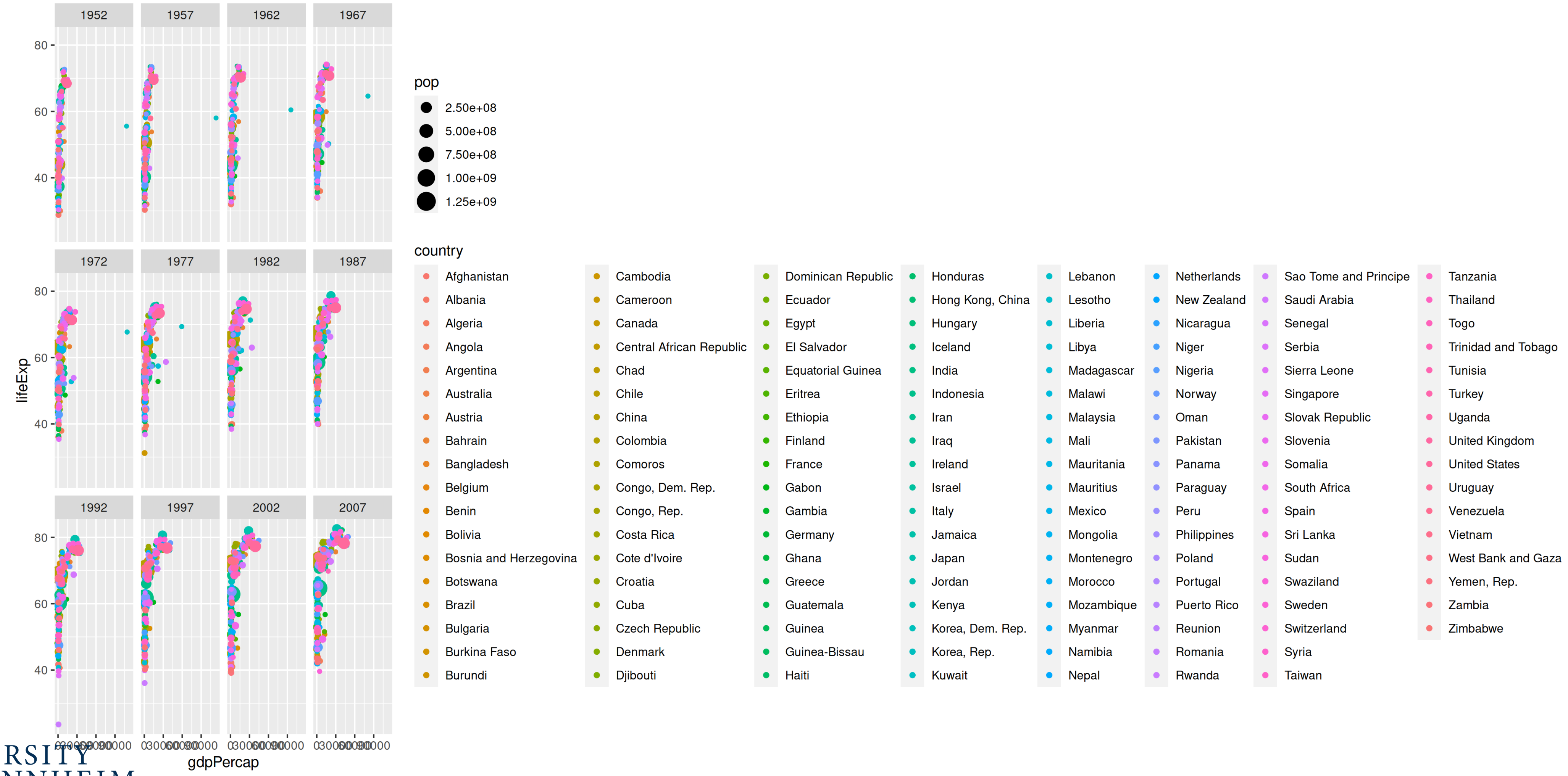


Let’s pause and think how to combine the two parts of our analysis

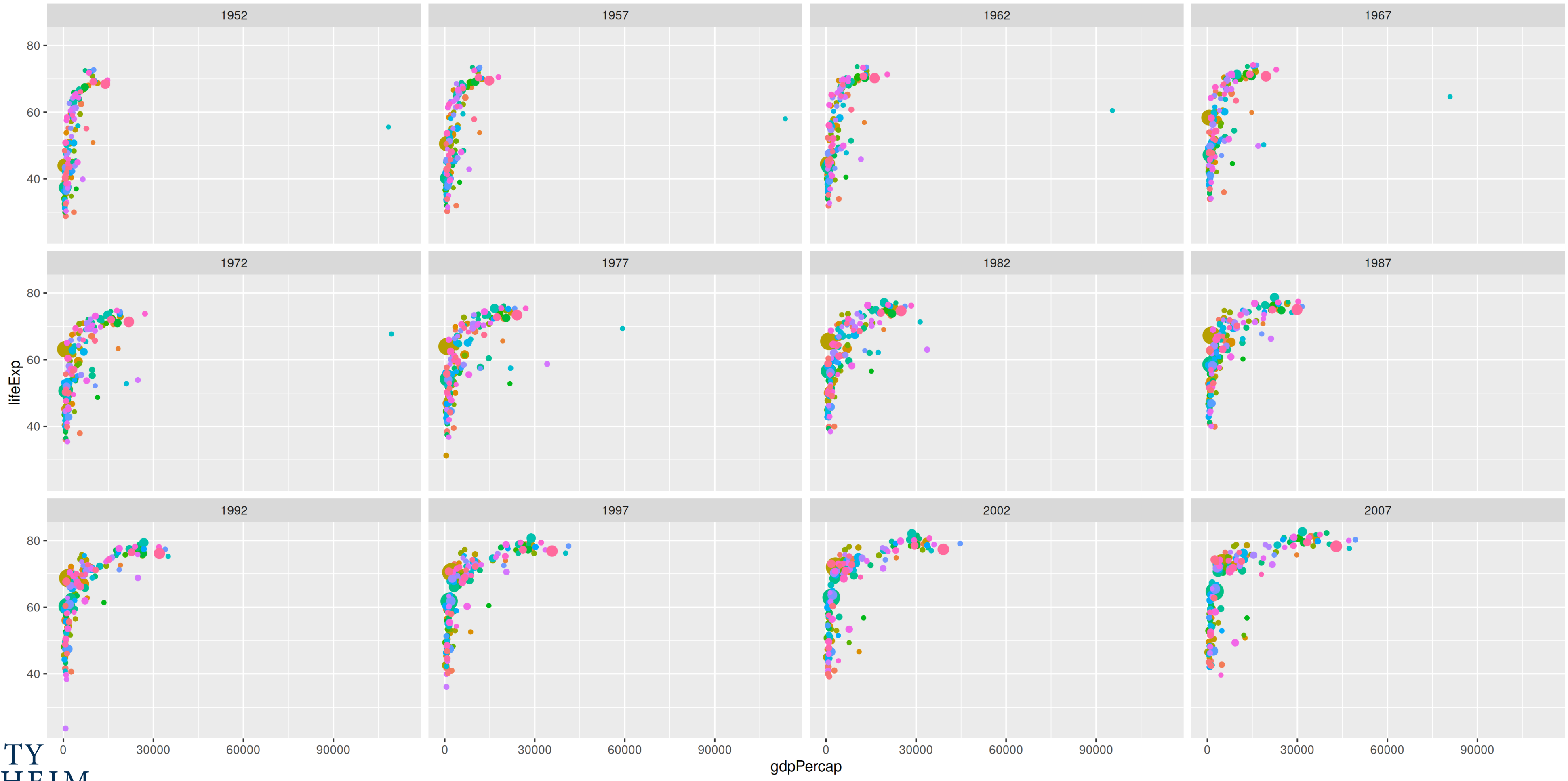
To get inspiration, you can check out “The Best Stats You’ve Ever Seen” by Hans Rosling

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

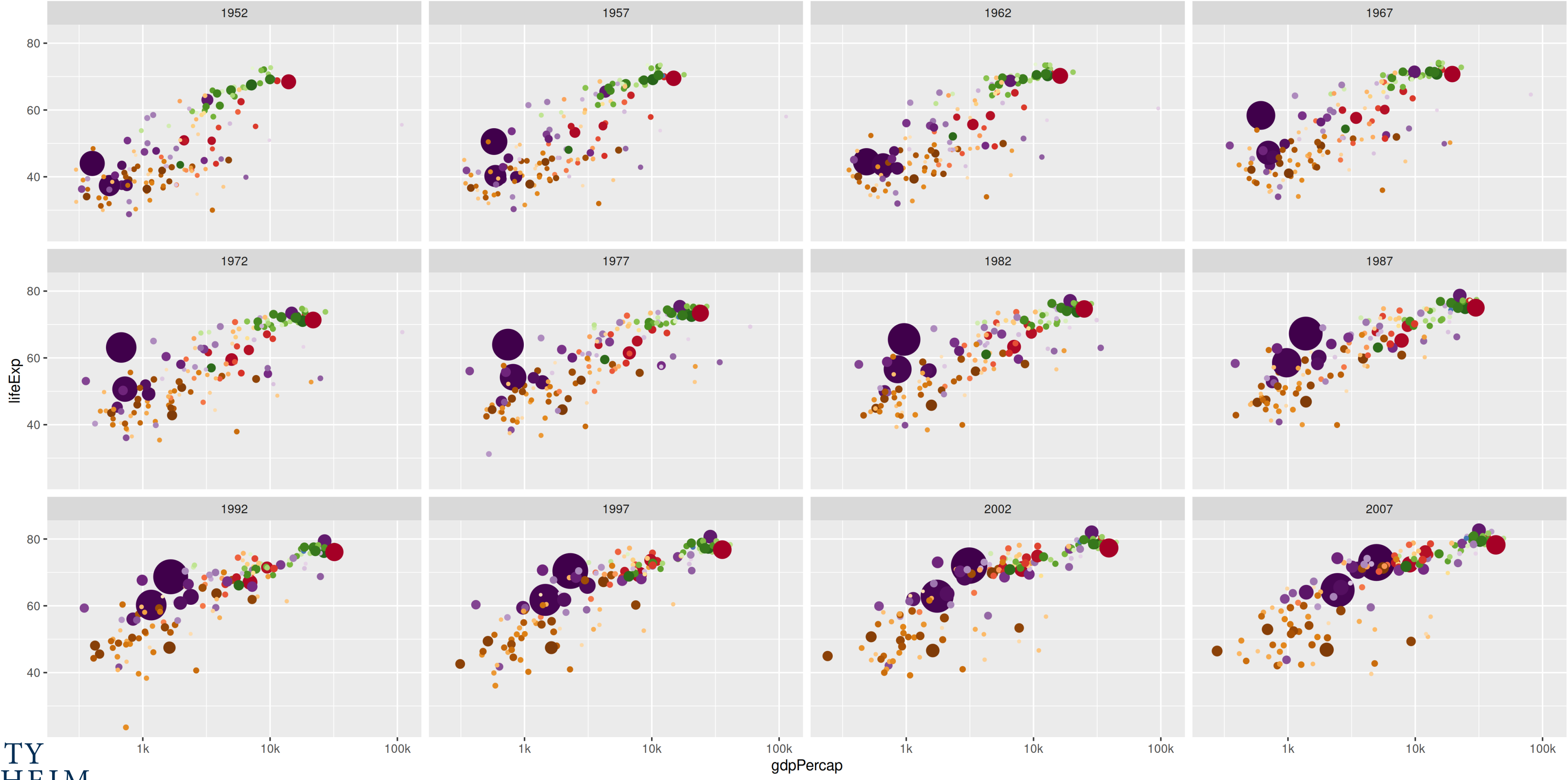
```r
g_hr <-
  ggplot(gapminder) +
  aes(x = gdpPercap, y = lifeExp, size = pop, color = country) +
  geom_point() +
  facet_wrap(~year)
```



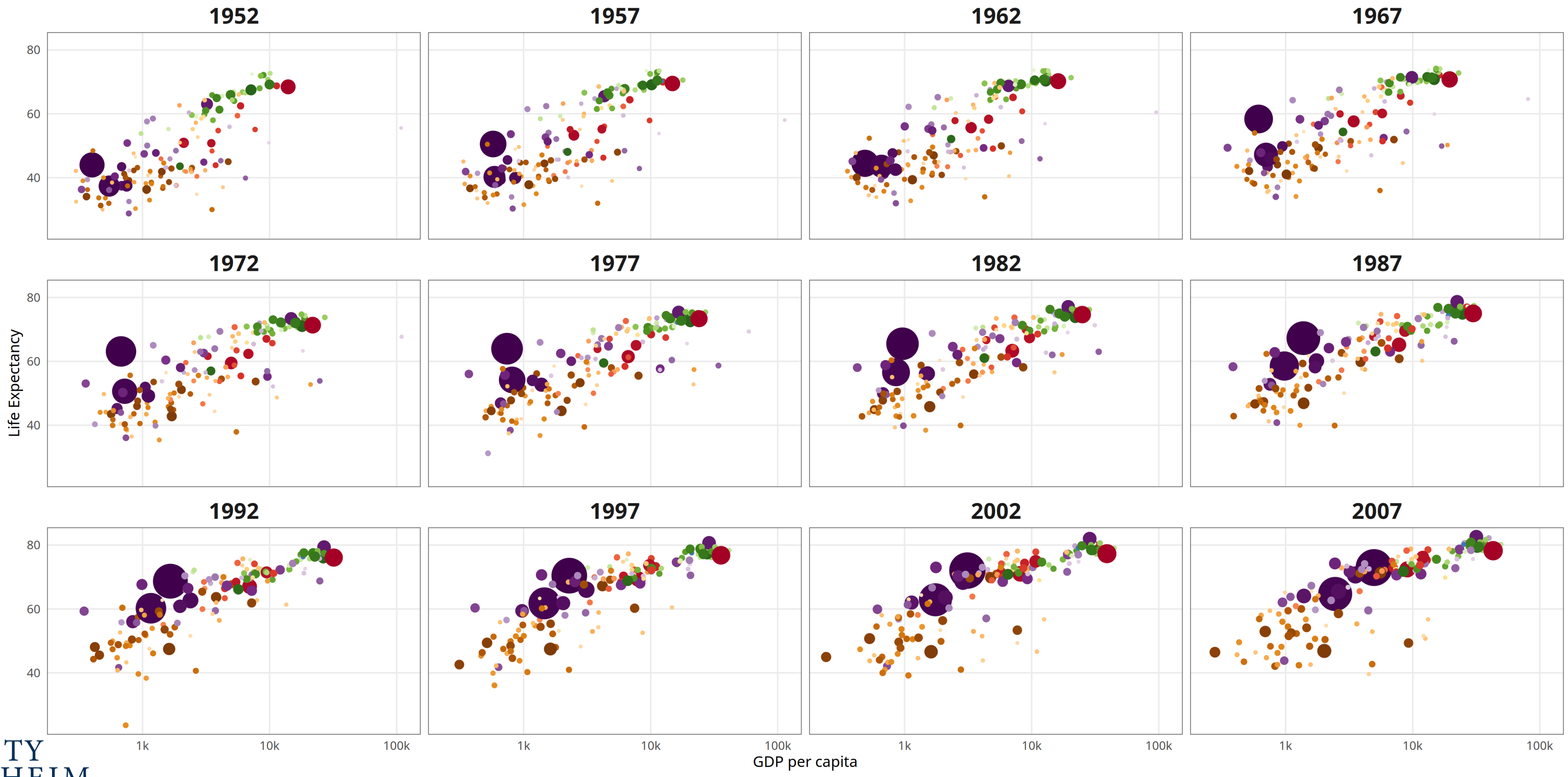
```r
g_hr <-
  ggplot(gapminder) +
  aes(x = gdpPercap, y = lifeExp, size = pop, color = country) +
  geom_point() +
  facet_wrap(~year) +
  guides(color = FALSE, size = FALSE)
```



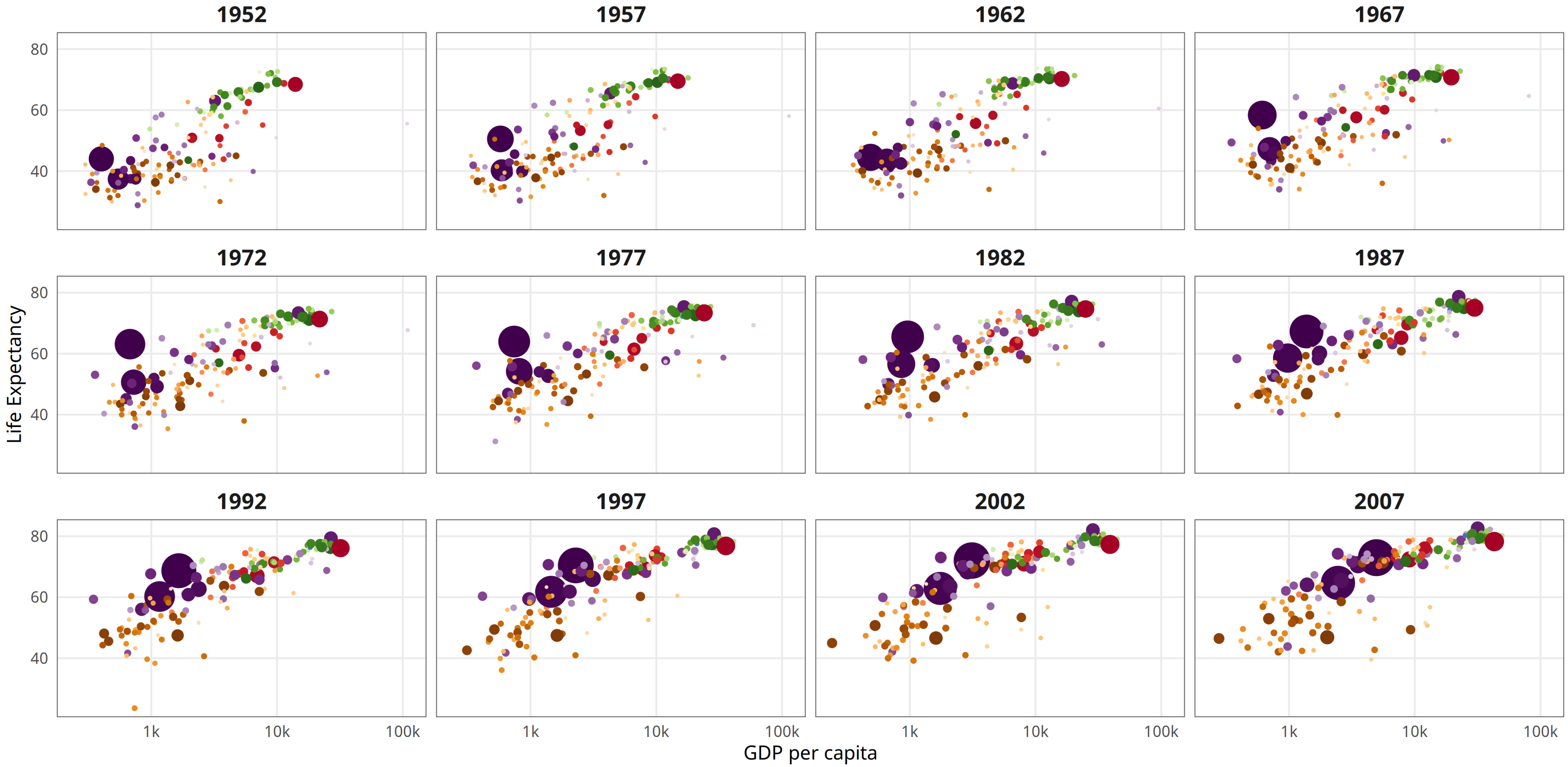
```r
g_hr <-
  g_hr +
  scale_x_log10(breaks = c(10^3, 10^4, 10^5),
                labels = c("1k", "10k", "100k")) +
  scale_color_manual(values = gapminder::country_colors) +
  scale_size(range = c(0.5, 12))
```



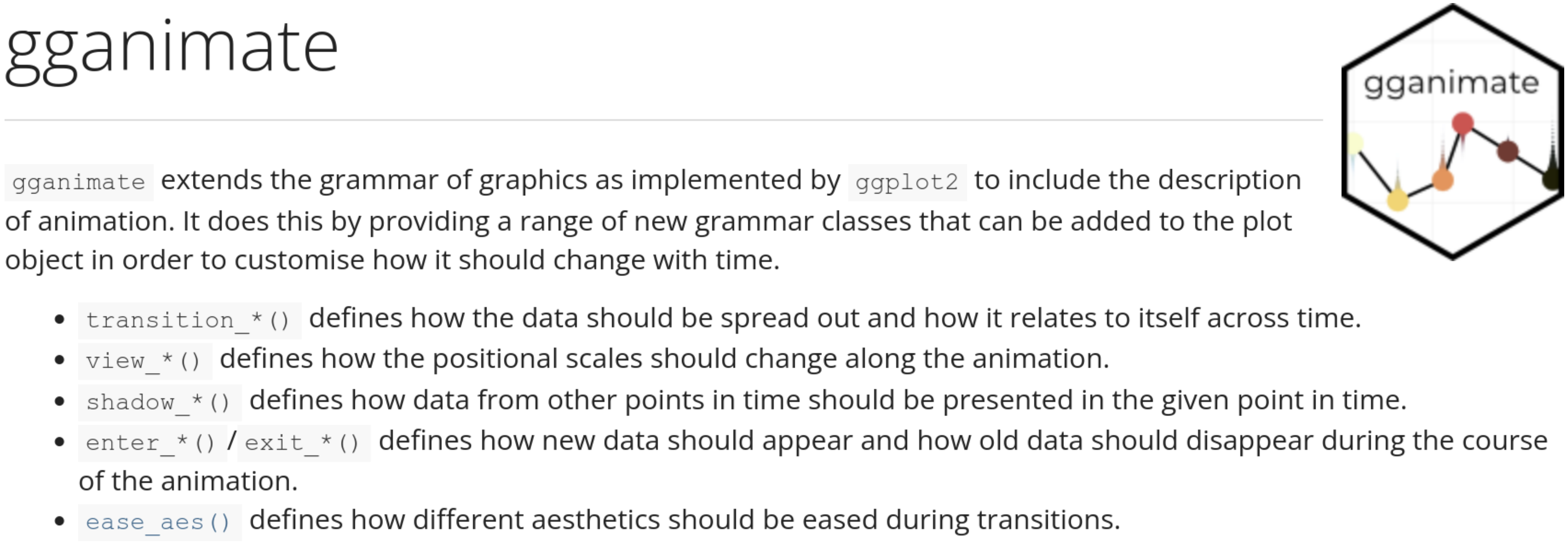
```
g_hr <- g_hr +
  labs(x = "GDP per capita", y = "Life Expectancy") +
  theme_minimal(base_family = "Fira Sans") +
  theme(strip.text = element_text(size = 16, face = "bold"),
    panel.border = element_rect(fill = NA, color = "grey40"),
    panel.grid.minor = element_blank())
```



```r
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!



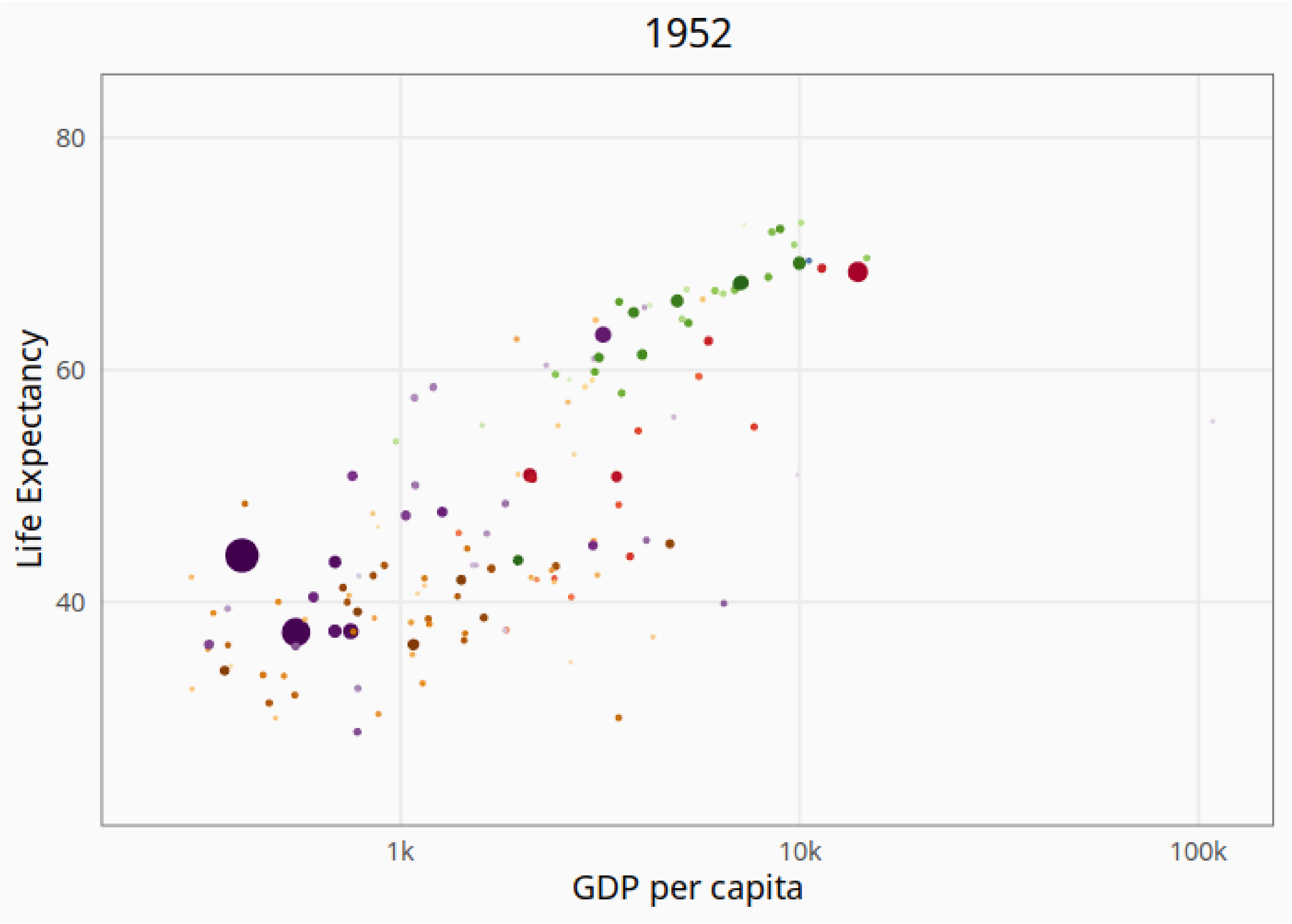
gganimate

`gganimate` extends the grammar of graphics as implemented by `ggplot2` to include the description of animation. It does this by providing a range of new grammar classes that can be added to the plot object in order to customise how it should change with time.

- `transition_*()` defines how the data should be spread out and how it relates to itself across time.
- `view_*()` defines how the positional scales should change along the animation.
- `shadow_*()` defines how data from other points in time should be presented in the given point in time.
- `enter_*()`/`exit_*()` defines how new data should appear and how old data should disappear during the course of the animation.
- `ease_aes()` defines how different aesthetics should be eased during transitions.

https://gganimate.com/

```r
# Same plot without facet_wrap()
g_hra <-
  ggplot(gapminder) +
  aes(x = gdpPercap, y = lifeExp, size = pop, color = country) +
  geom_point() +
  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(18, base_family = "Fira Sans") +
  theme(
    plot.background = element_rect("#FAFAFA", color = NA),
    strip.text = element_text(size = 16, face = "bold"),
    panel.border = element_rect(fill = NA, color = "grey40"),
```



UNIVERSITY
OF MANNHEIM

Acknowledgements

http://github.com/gadenbuie/gentle-ggplot2