Engaging and Effective ggplot (1)

VCS, Rice 2025

Kieran Healy
Duke University

March 2025

Get Set Up

Load our packages

```
library(here)      # manage file paths
library(socviz)    # data and some useful functions
library(tidyverse) # your friend and mine
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.5
✔ forcats   1.0.0     ✔ stringr   1.5.1
✔ ggplot2   3.5.1     ✔ tibble    3.2.1
✔ lubridate 1.9.4     ✔ tidyr     1.3.1
✔ purrr     1.0.4
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(gapminder) # some data
```

Make a plot

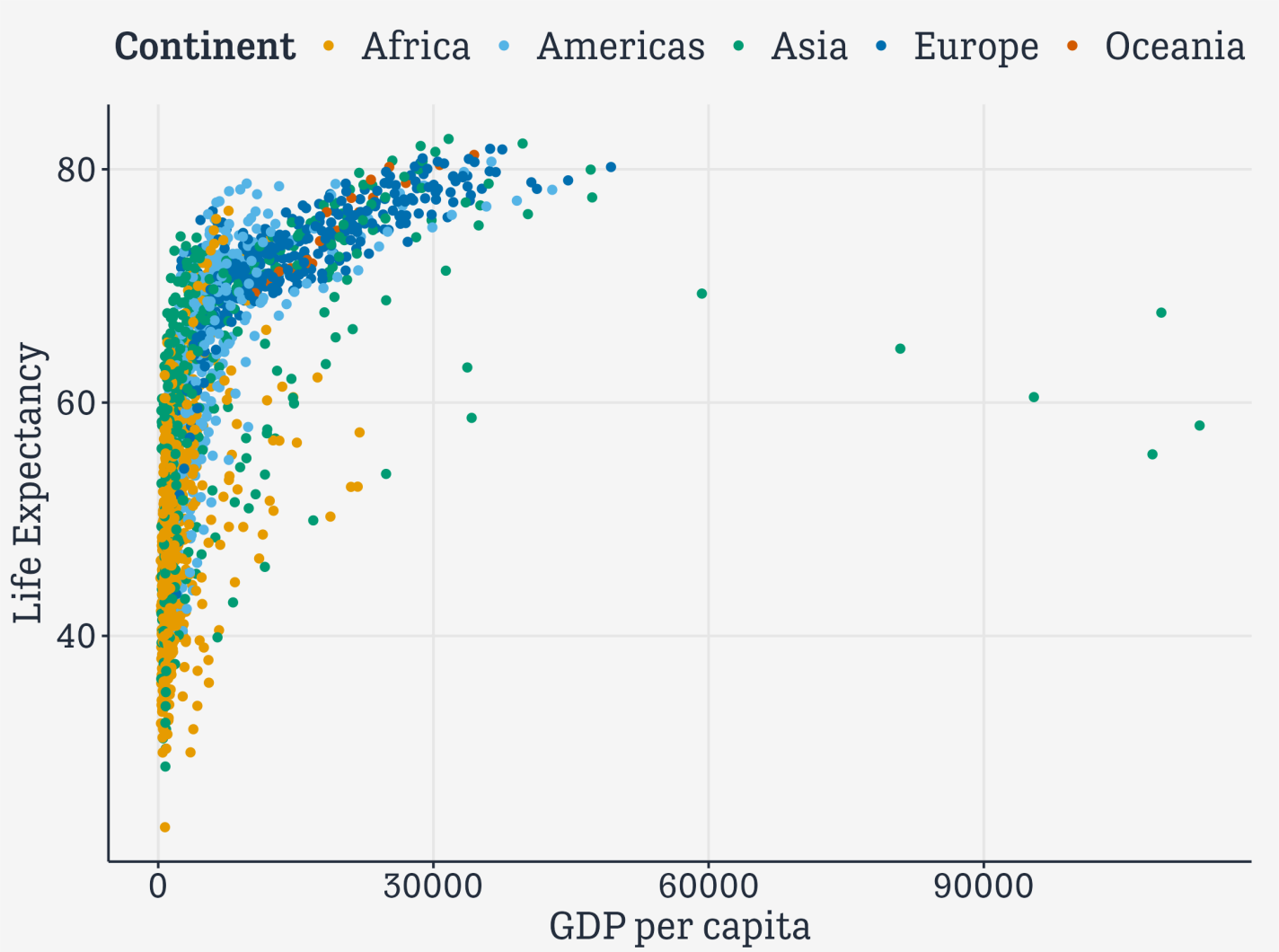
```
gapminder
```

```
# A tibble: 1,704 × 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.
 7 Afghanistan Asia       1982    39.9 12881816      978.
 8 Afghanistan Asia       1987    40.8 13867957      852.
 9 Afghanistan Asia       1992    41.7 16317921      649.
10 Afghanistan Asia       1997    41.8 22227415      635.
# ℹ 1,694 more rows
```

Make a plot

```
library(tidyverse)
library(gapminder)

p ← ggplot(data = gapminder,
           mapping = aes(x = gdpPercap,
                         y = lifeExp,
                         color = continent))

p + geom_point() +
   labs(x = "GDP per capita",
        y = "Life Expectancy",
        color = "Continent")
```

Make a plot


Continent
Africa
Americas
Asia
Europe
Oceania
Life Expectancy
80
60
40
0
30000
60000
90000
GDP per capita

What we did

```
library(gapminder)
```

The data comes from somewhere. In this case, it's in the `gapminder` package.

What we did

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

New object named **`p`** `gets` the output of the `ggplot()` *function*, given these *arguments*

One of the arguments, `mapping`, is itself taking the output of a function named `aes()`

What we did

```
p + geom_point() +
   labs(x = "GDP per capita",
        y = "Life Expectancy",
        color = "Continent")
```

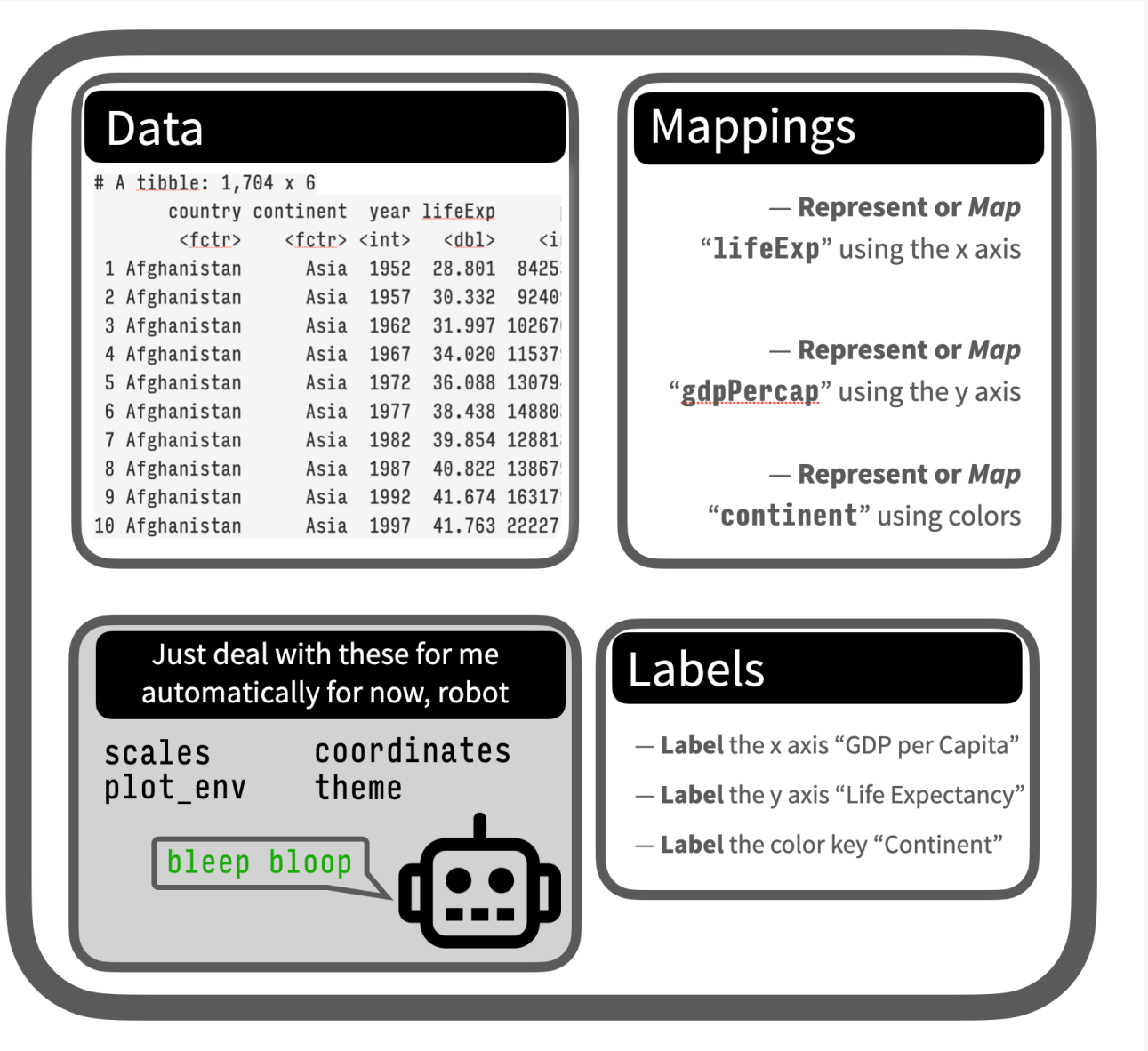
"Show me the output of the `p` object and the `geom_point()` function, and then add these labels with the `labs()` function.

The `+` here acts just like the `▷` pipe, but for ggplot functions only. (This is an accident of history.)

And what is R doing?

R objects are just lists of stuff to use or things to do

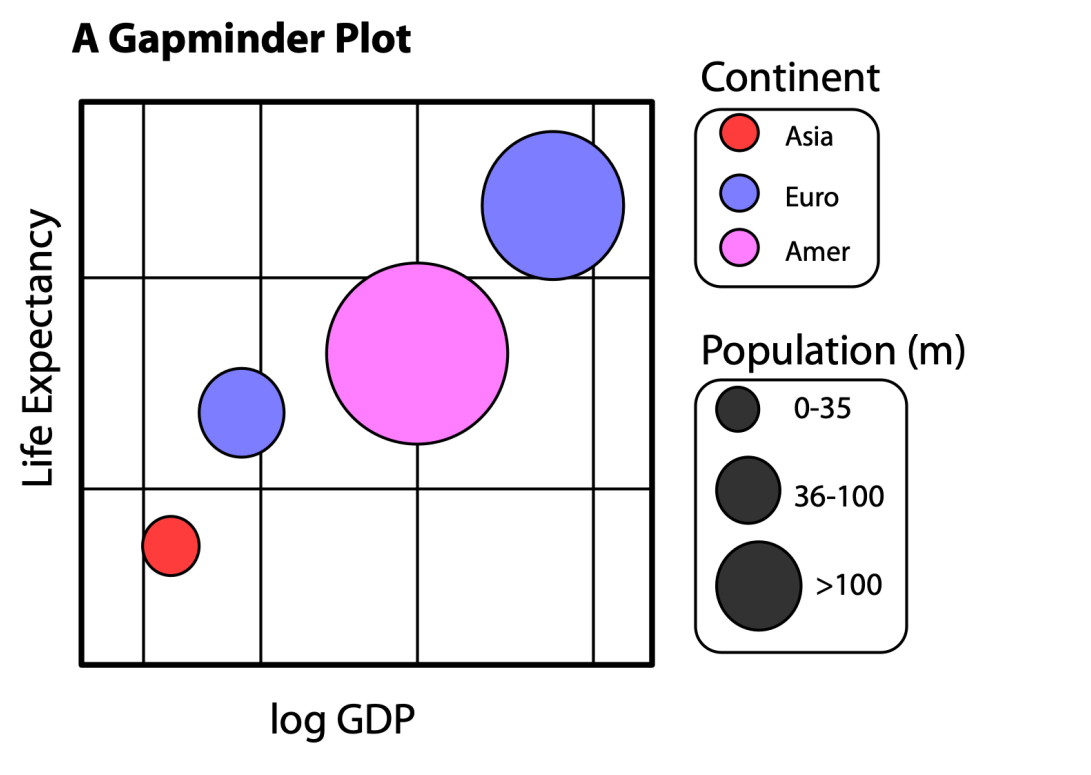
Objects are like Bento Boxes


Data
A tibble: 1,704 x 6
country continent year lifeExp
<fctr> <fctr> <int> <dbl> <i
1 Afghanistan Asia 1952 28.801 8425
2 Afghanistan Asia 1957 30.332 9240
3 Afghanistan Asia 1962 31.997 10267
4 Afghanistan Asia 1967 34.020 11537
5 Afghanistan Asia 1972 36.088 13079
6 Afghanistan Asia 1977 38.438 14880
7 Afghanistan Asia 1982 39.854 12881
8 Afghanistan Asia 1987 40.822 13867
9 Afghanistan Asia 1992 41.674 16317
10 Afghanistan Asia 1997 41.763 22227
Mappings
— Represent or Map
“lifeExp” using the x axis
— Represent or Map
“gdpPercap” using the y axis
— Represent or Map
“continent” using colors
Just deal with these for me automatically for now, robot
scales
coordinates
plot_env
theme
bleep bloop
Labels
— Label the x axis “GDP per Capita”
— Label the y axis “Life Expectancy”
— Label the color key “Continent”

The p object

A Plot's Components

What we need our code to make



Data **represented** by visual elements;

like *position*, *length*, *color*, and *size*;

Each measured on some **scale**;

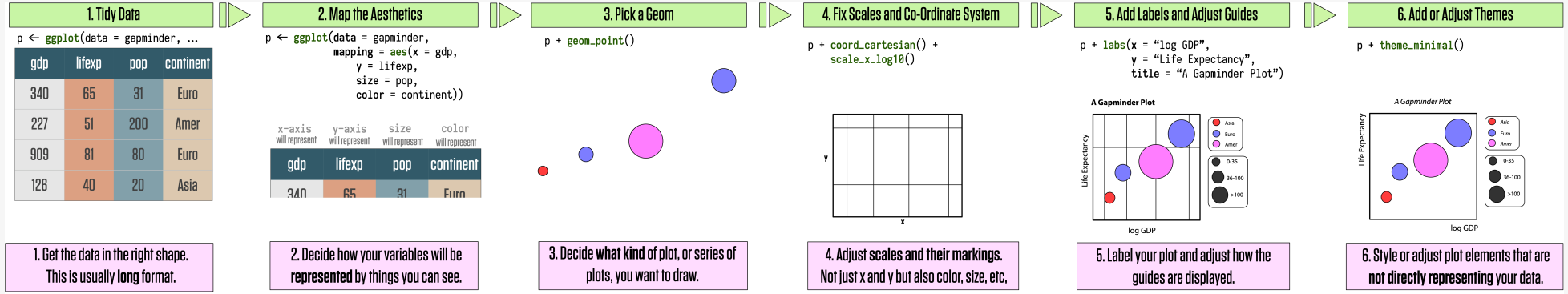
Each scale with a labeled **guide**;

With the plot itself also **titled** and labeled.

How does `ggplot` do this?

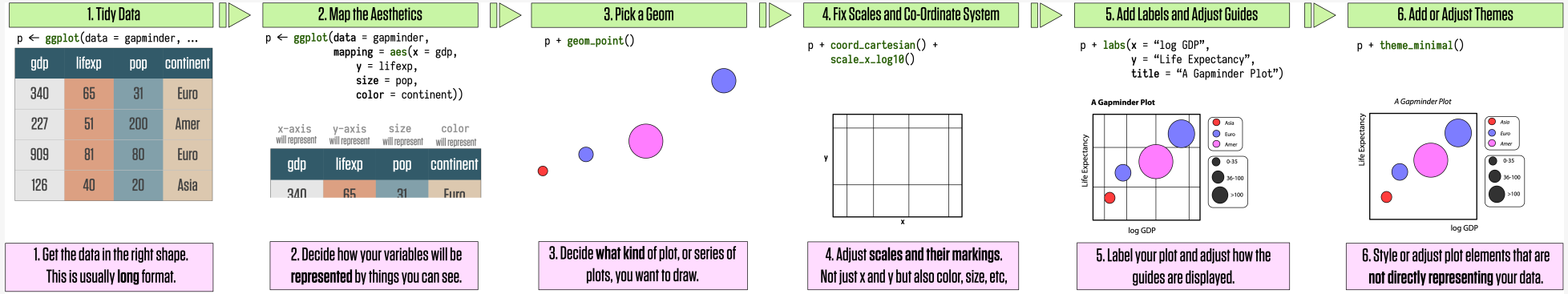
ggplot's flow of action

Here's the whole thing, start to finish



Flow of action

We'll go through it step by step



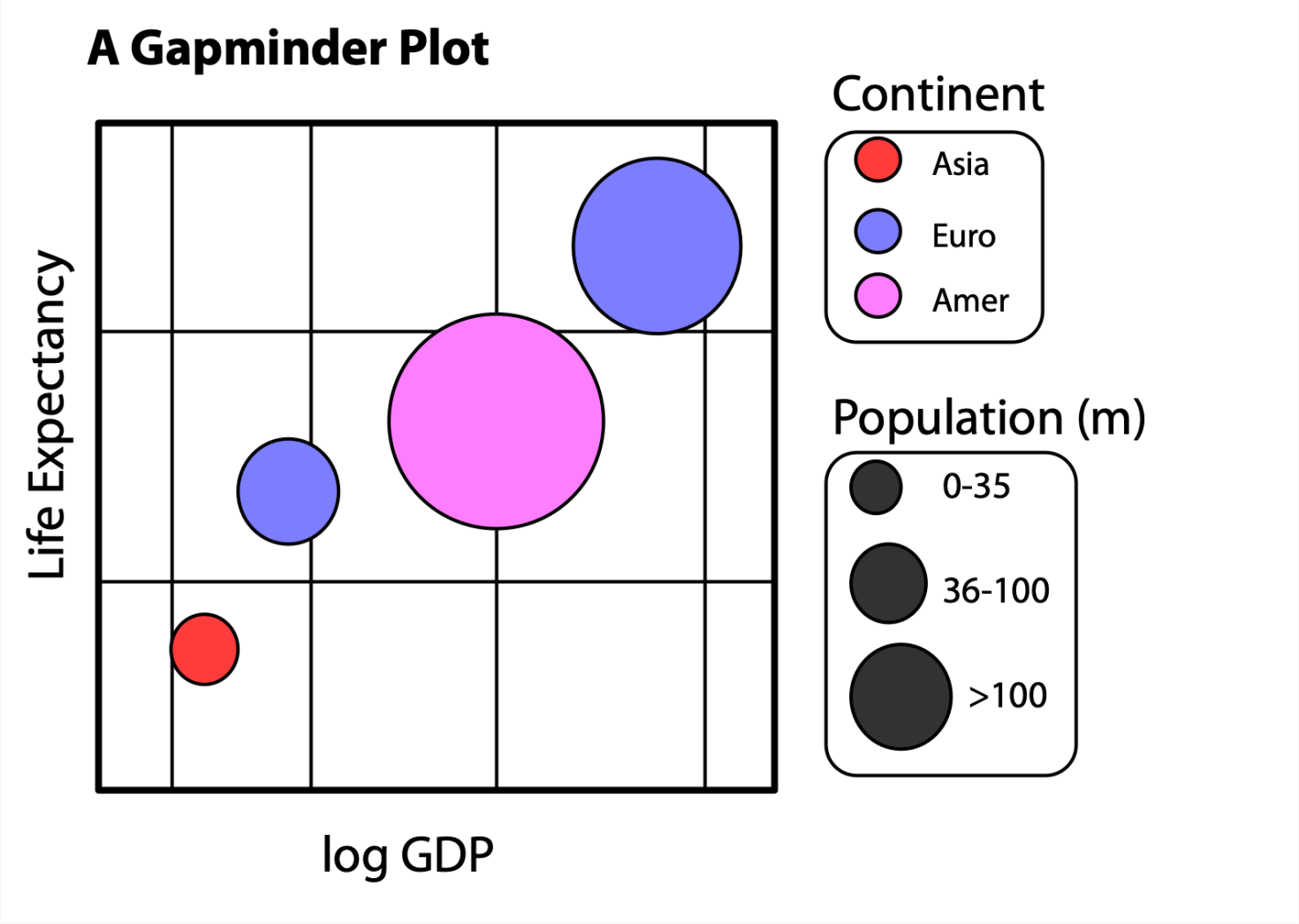
Flow of action

ggplot's flow of action

gdp	lifexp	pop	continent
340	65	31	Euro
227	51	200	Amer
909	81	80	Euro
126	40	20	Asia

What we start with

ggplot's flow of action



Where we're going

ggplot's flow of action

1. Tidy Data

```
p ← ggplot(data = gapminder, ...
```

gdp	lifexp	pop	continent
340	65	31	Euro
227	51	200	Amer
909	81	80	Euro
126	40	20	Asia

1. Get the data in the right shape. This is usually **long** format.

2. Map the Aesthetics

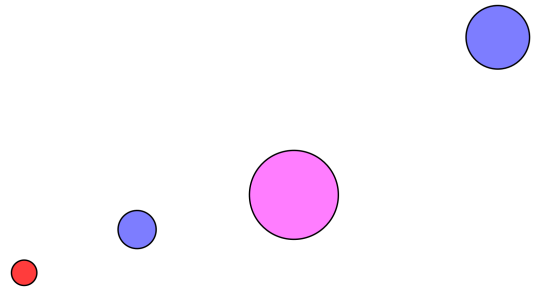
```
p ← ggplot(data = gapminder,
           mapping = aes(x = gdp,
               y = lifexp,
               size = pop,
               color = continent))
```

x-axis will represent	y-axis will represent	size will represent	color will represent
gdp	lifexp	pop	continent
340	65	31	Euro

2. Decide how your variables will be **represented** by things you can see.

3. Pick a Geom

```
p + geom_point()
```



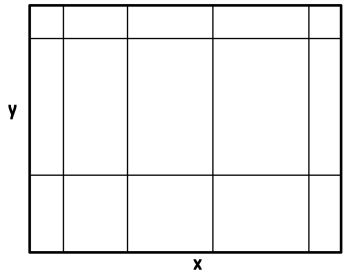
3. Decide **what kind** of plot, or series of plots, you want to draw.

Core steps

ggplot's flow of action

4. Fix Scales and Co-Ordinate System

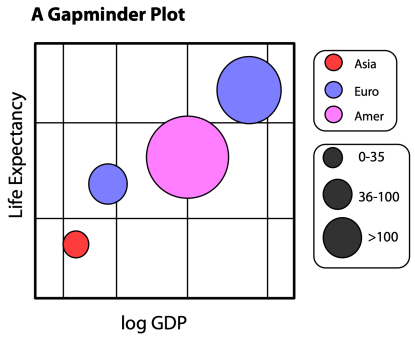
```
p + coord_cartesian() +
    scale_x_log10()
```



4. Adjust **scales and their markings.** Not just x and y but also color, size, etc,

5. Add Labels and Adjust Guides

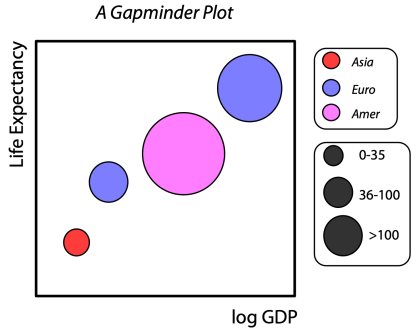
```
p + labs(x = "log GDP",
         y = "Life Expectancy",
         title = "A Gapminder Plot")
```



5. Label your plot and adjust how the guides are displayed.

6. Add or Adjust Themes

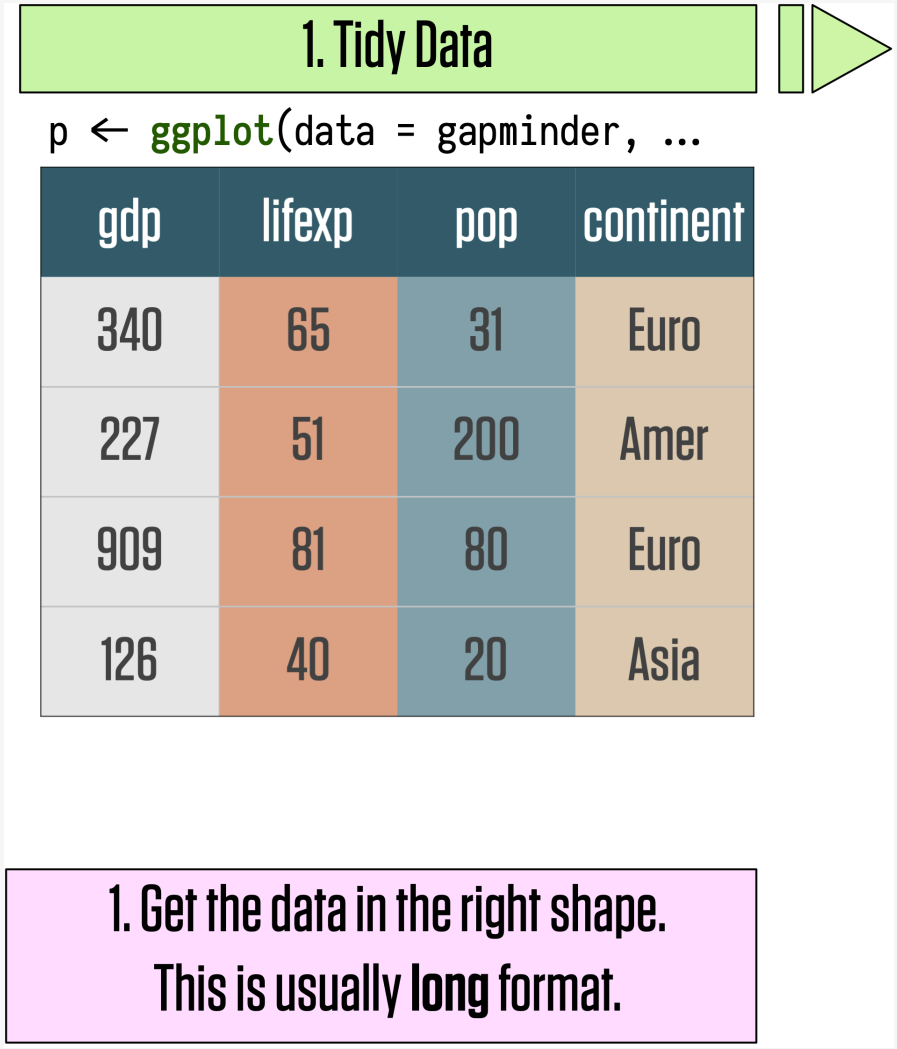
```
p + theme_minimal()
```



6. Style or adjust plot elements that are **not directly representing** your data.

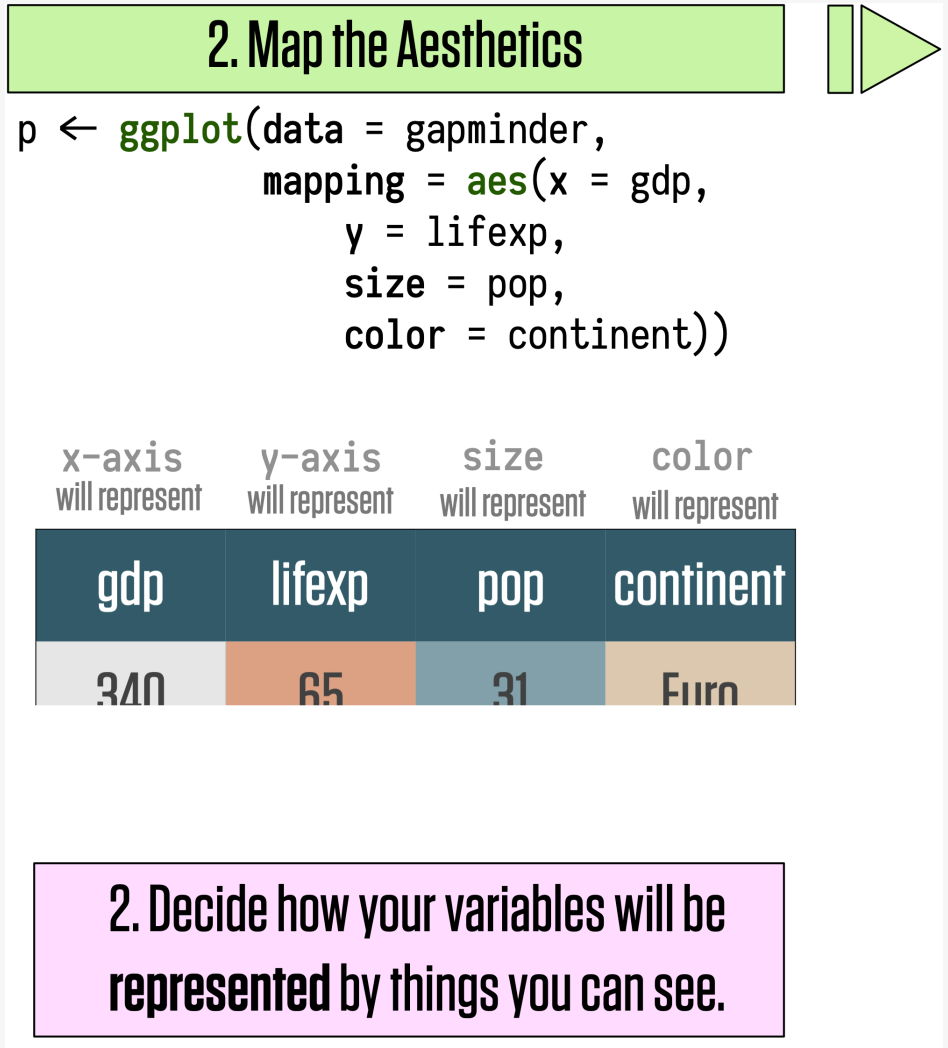
Optional steps

ggplot's flow of action: required



Tidy data

ggplot's flow of action: required

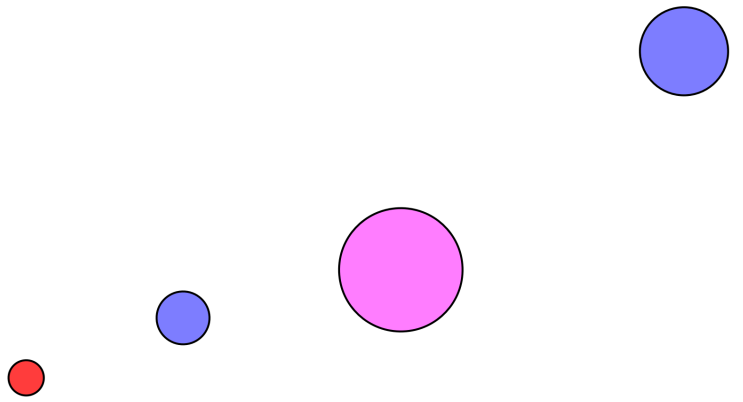


Aesthetic mappings

ggplot's flow of action: required

3. Pick a Geom

```
p + geom_point()
```



3. Decide **what kind** of plot, or series of plots, you want to draw.

A composable pipeline

Start with the data.

Do a sequence of things to it until you get a plot.

Each class of thing you do is done by **applying** some special-purpose function.

The result of each function can be passed along to the next one.

"**And then, and then, and then**, …" with ▷ outside of ggplot and + inside it.

A series of transformations yielding a plot with a layered structure.

Let’s go piece by piece

Start with the data

```
gapminder
```

```
# A tibble: 1,704 × 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.
 7 Afghanistan Asia       1982    39.9 12881816      978.
 8 Afghanistan Asia       1987    40.8 13867957      852.
 9 Afghanistan Asia       1992    41.7 16317921      649.
10 Afghanistan Asia       1997    41.8 22227415      635.
# ℹ 1,694 more rows
```

```
dim(gapminder)
```

```
[1] 1704    6
```

Create a plot object

Data is the `gapminder` tibble.

```
p ← ggplot(data = gapminder)
```

Map variables to aesthetics

Tell `ggplot` the variables you want represented by visual elements on the plot

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

Map variables to aesthetics

The `mapping = aes( ... )` call links variables to things you will see on the plot.

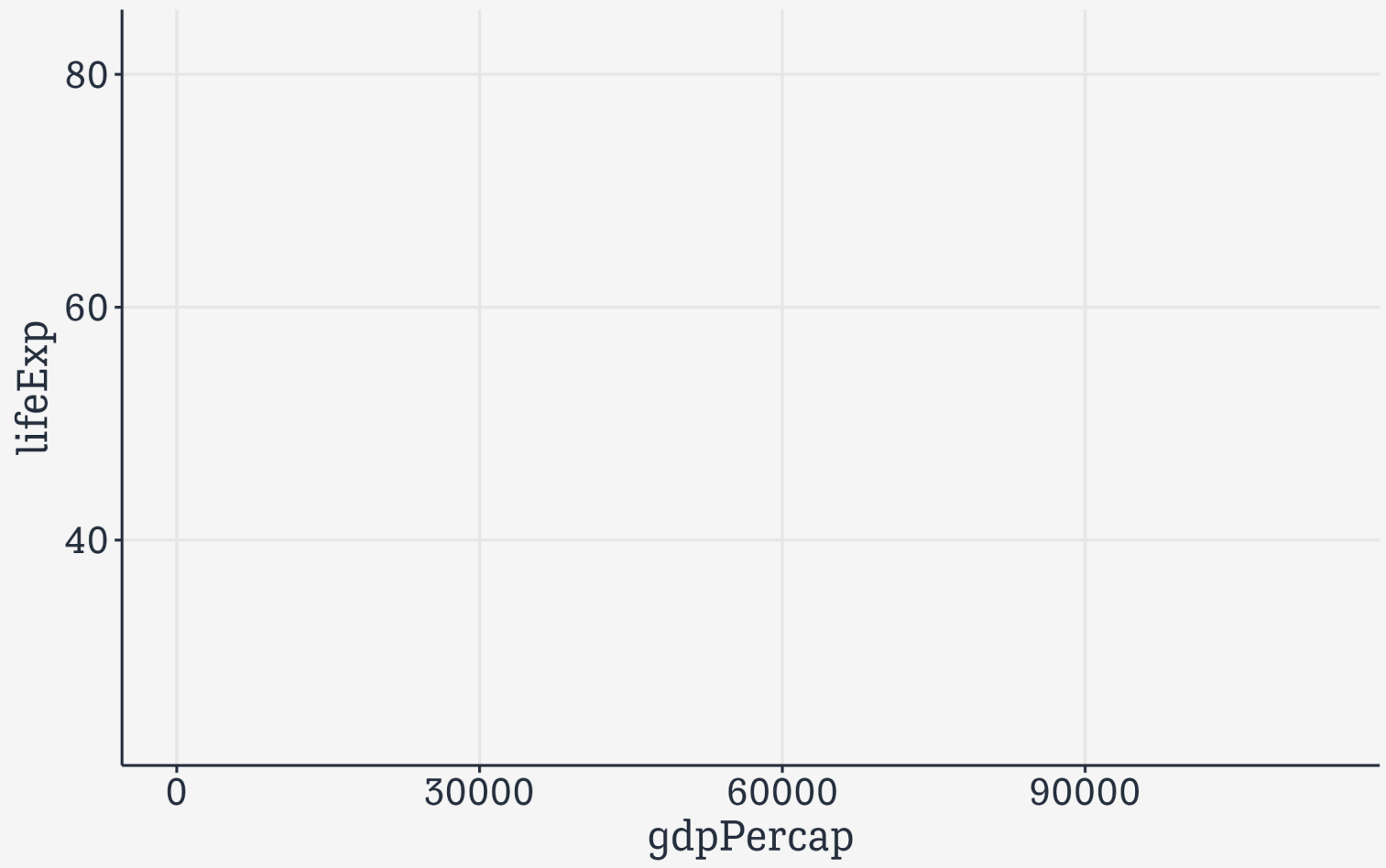
`x` and `y` represent the quantities determining position on the x and y axes.

Other aesthetic mappings can include, e.g., `color`, `shape`, `size`, and `fill`.

Mappings do not *directly* specify the particular, e.g., colors, shapes, or line styles that will appear on the plot. They say *which variables* in the data will be represented by *which visible elements* on the plot.

p has data and mappings but no geom

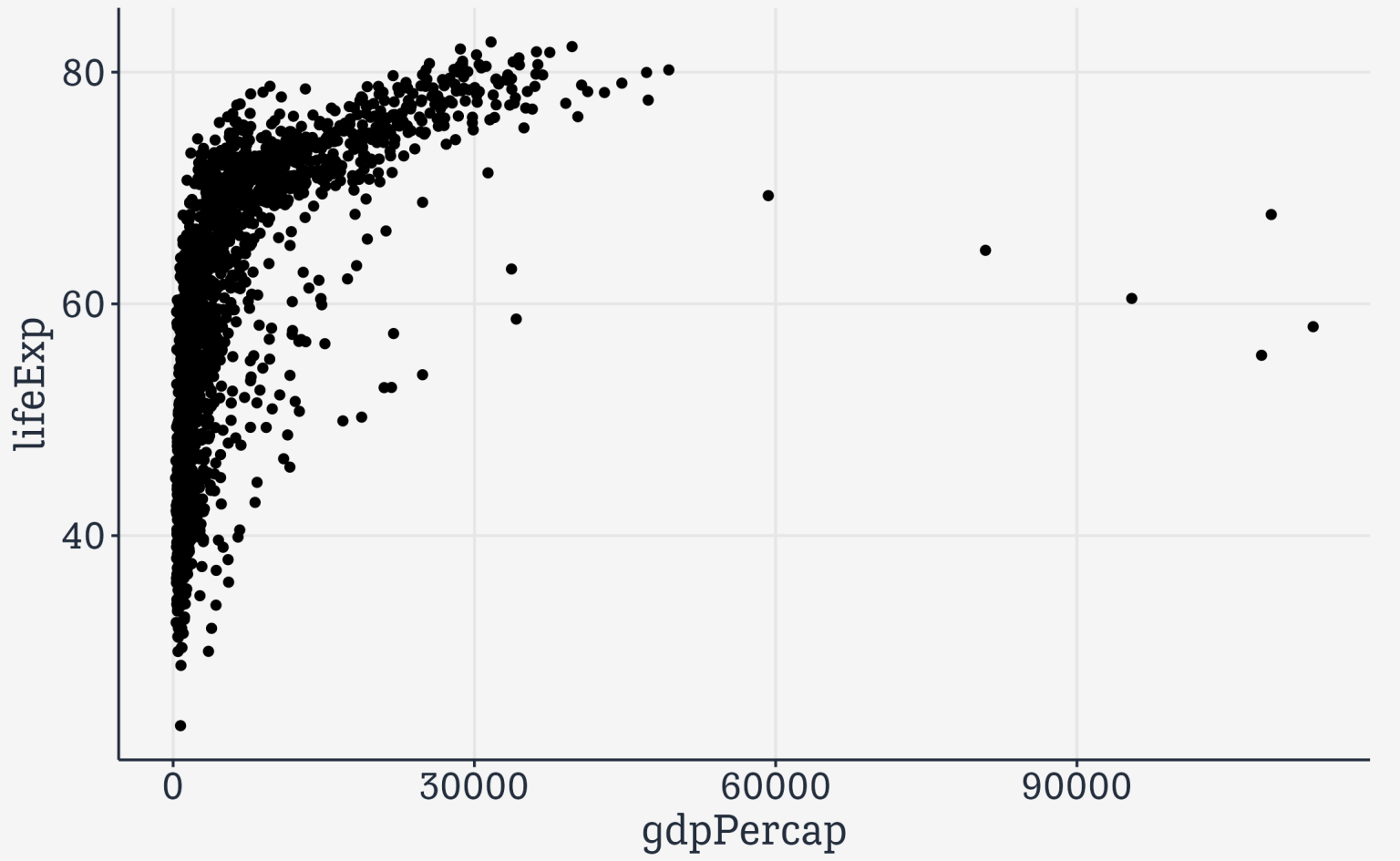
```
p
```



This empty plot has no geoms.

Add a geom

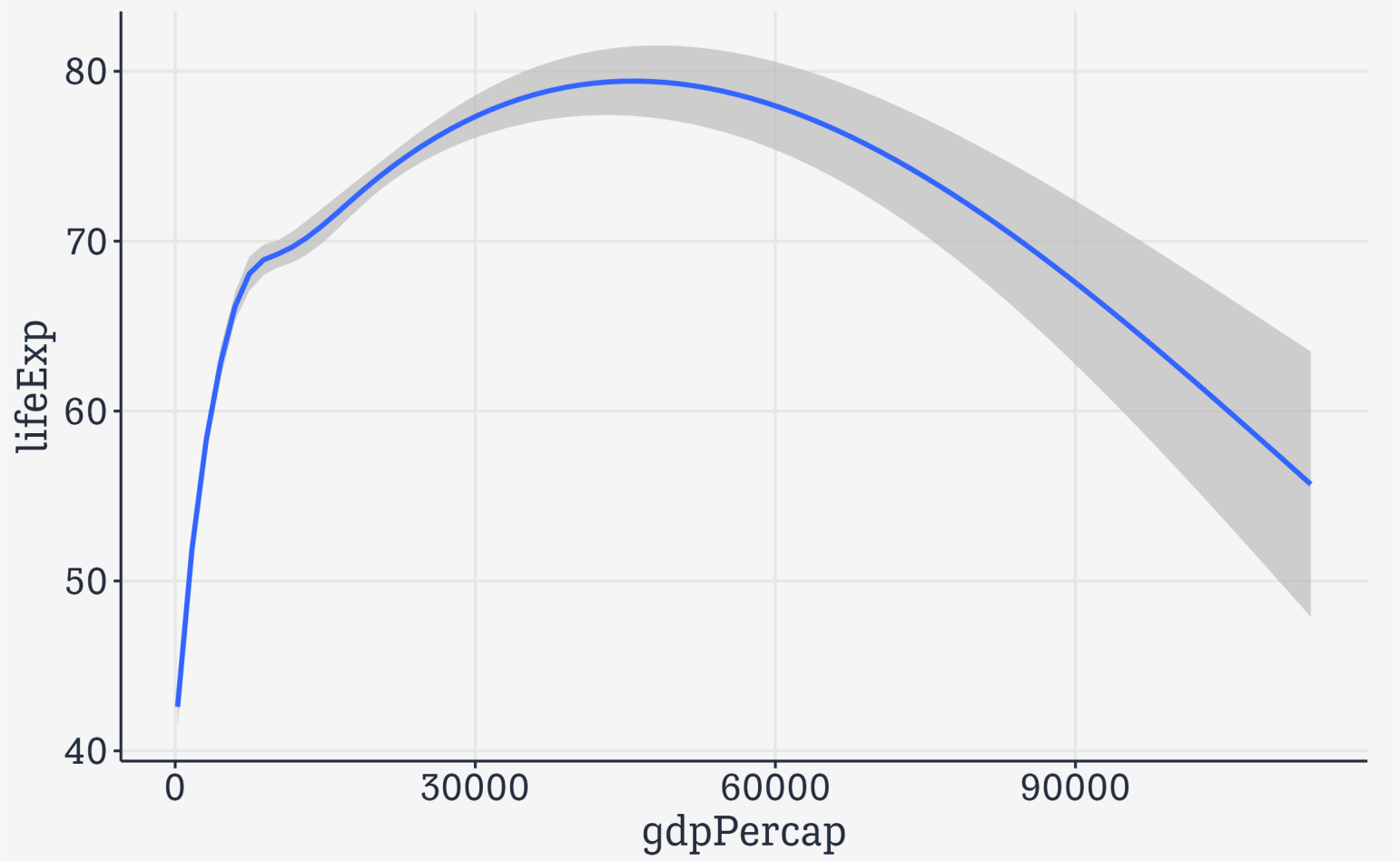
```
p + geom_point()
```



A scatterplot of Life Expectancy vs GDP

Try a different geom

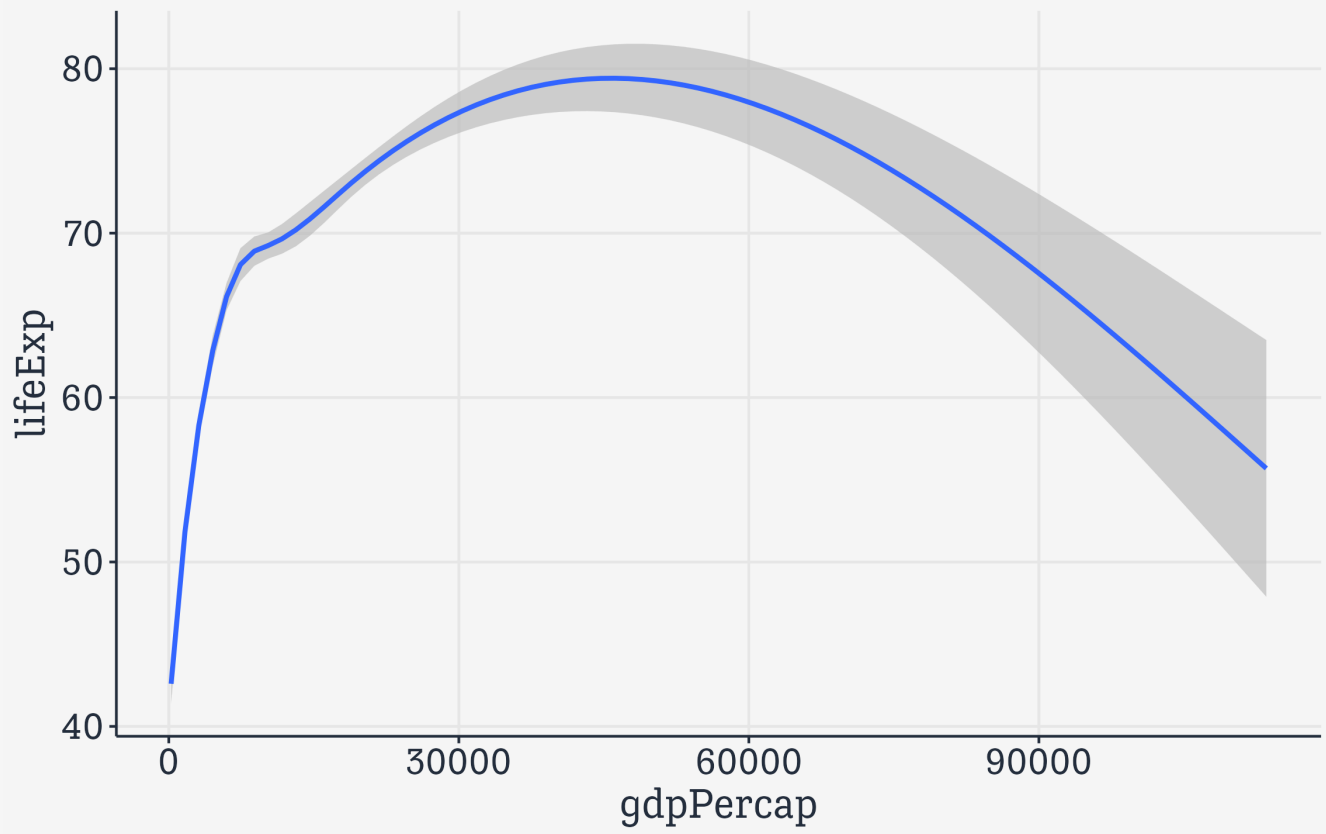
```
p + geom_smooth()
```



A scatterplot of Life Expectancy vs GDP

Build your plots layer by layer

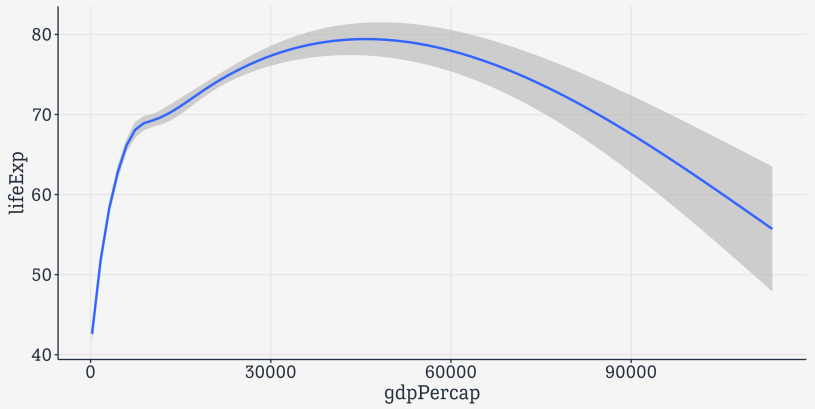
```
p ← ggplot(data = gapminder,
           mapping = aes(x = gdpPercap,
                         y=lifeExp))
p + geom_smooth()
```



Life Expectancy vs GDP, using a smoother.

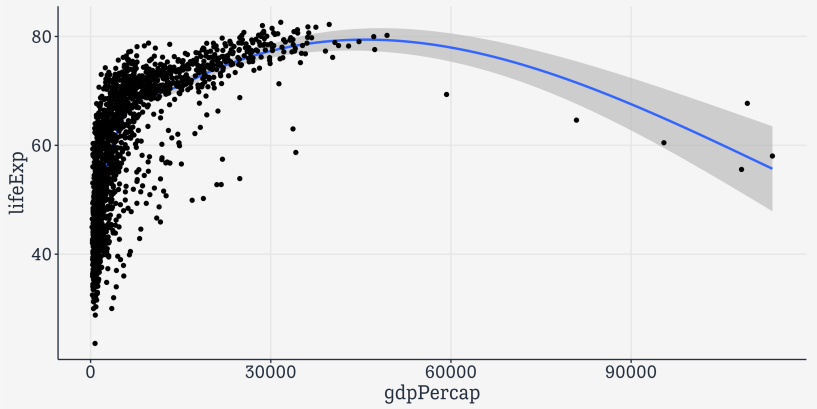
This process is additive

```
p + geom_smooth()
```



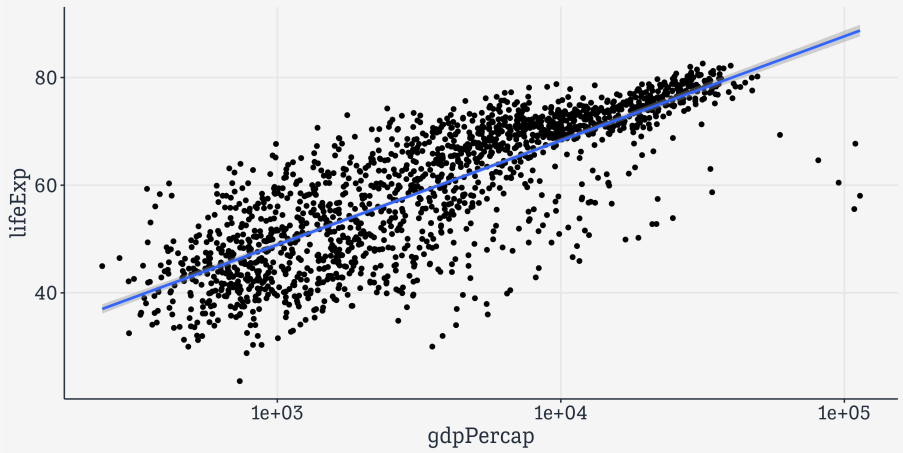
This process is additive

```
p + geom_smooth() +
    geom_point()
```



Every geom is a function

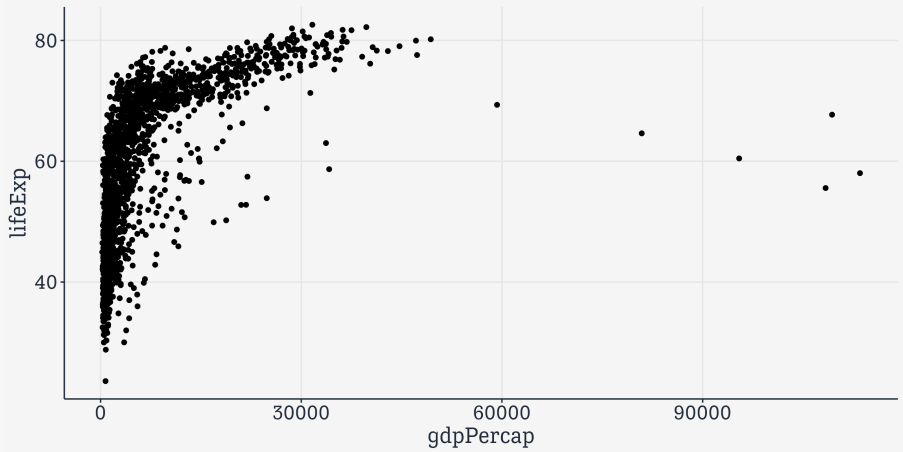
```
p + geom_point() +
    geom_smooth(method = "lm") +
    scale_x_log10()
```



Functions take arguments

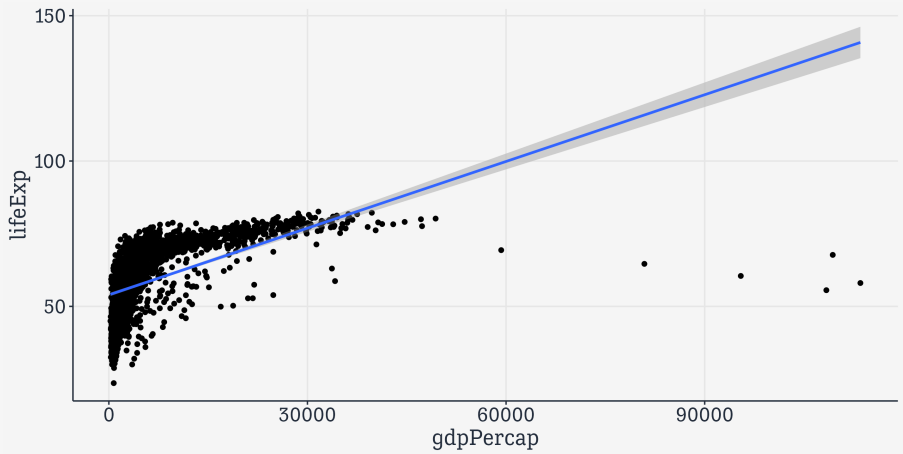
Fix the labels

```
p + geom_point()
```



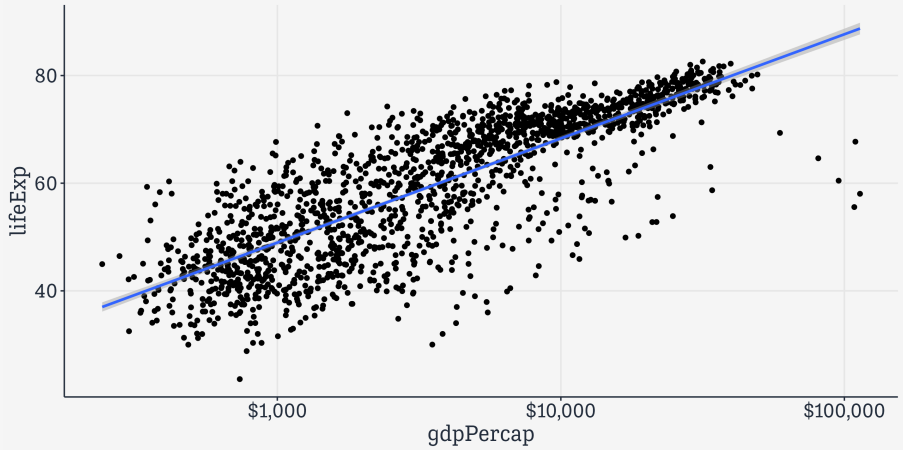
Fix the labels

```
p + geom_point() +
    geom_smooth(method = "lm")
```



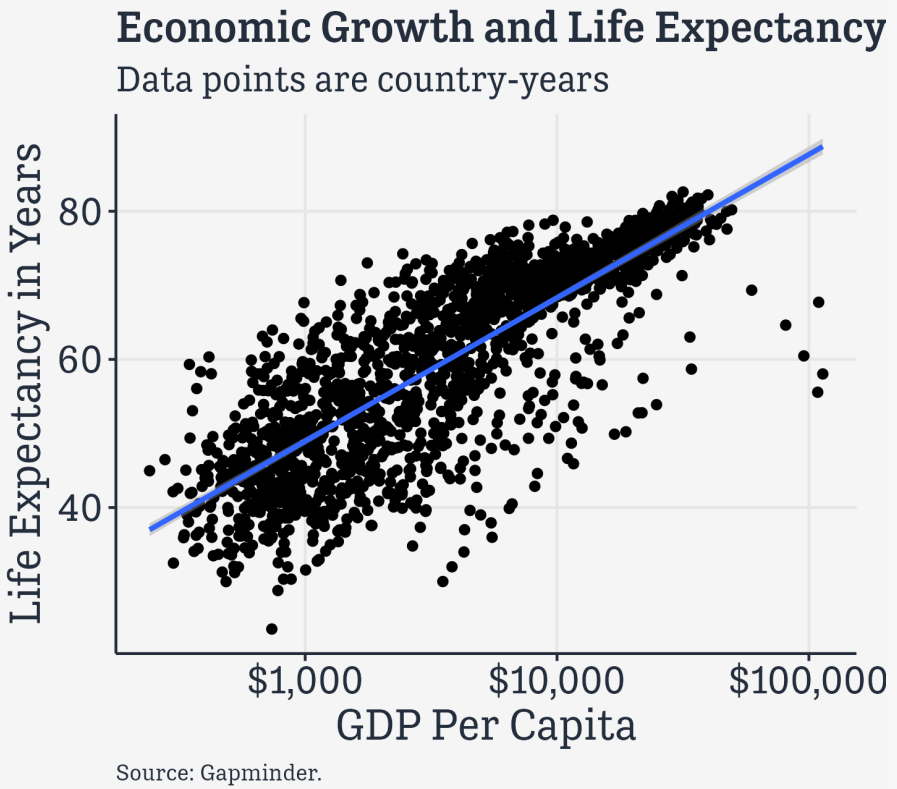
Fix the labels

```
p + geom_point() +
    geom_smooth(method = "lm") +
    scale_x_log10(labels = scales::label_dollar())
```



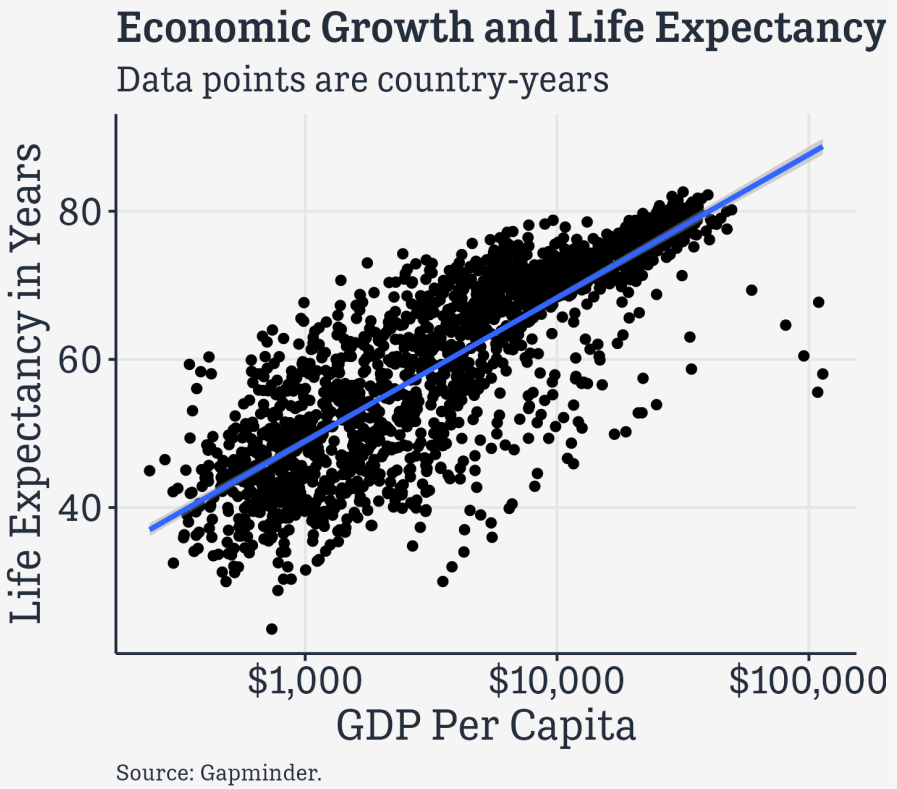
Add labels, title, and caption

```
p + geom_point() +
  geom_smooth(method = "lm") +
    scale_x_log10(labels = scales::label_dollar()) +
    labs(x = "GDP Per Capita",
         y = "Life Expectancy in Years",
         title = "Economic Growth and Life Expectancy",
         subtitle = "Data points are country-years",
         caption = "Source: Gapminder.")
```



We can also pipe the data to ggplot

```
gapminder ▷
  ggplot(mapping = aes(x = gdpPercap,
                       y = lifeExp)) +
  geom_point() +
  geom_smooth(method = "lm") +
    scale_x_log10(labels = scales::label_dollar()) +
    labs(x = "GDP Per Capita",
         y = "Life Expectancy in Years",
         title = "Economic Growth and Life Expectancy",
         subtitle = "Data points are country-years",
         caption = "Source: Gapminder.")
```



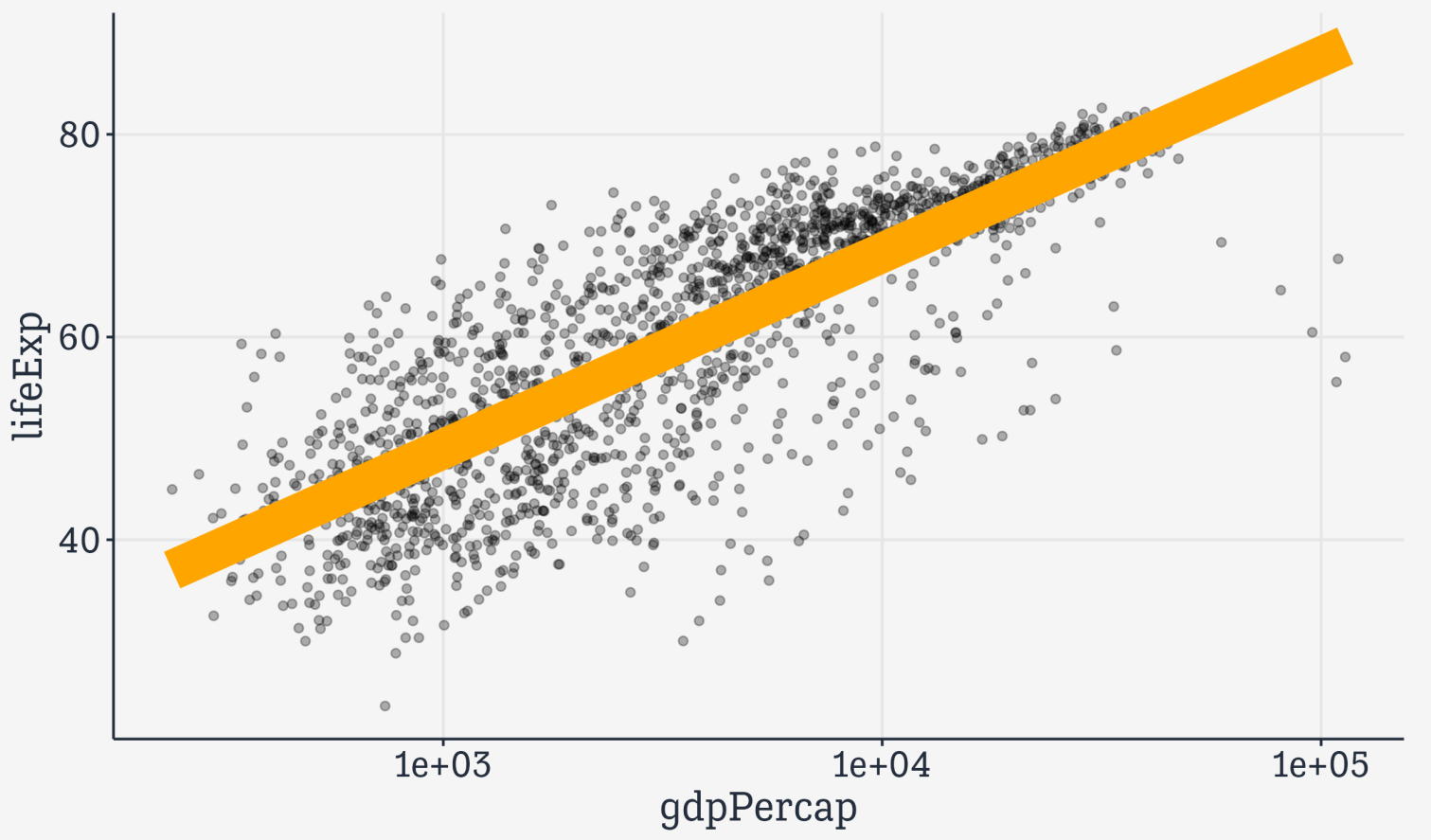
Geoms can take many arguments

Here we set `color`, `linewidth`, and `alpha`. Meanwhile `x` and `y` are mapped.

We also give non-default values to some other arguments

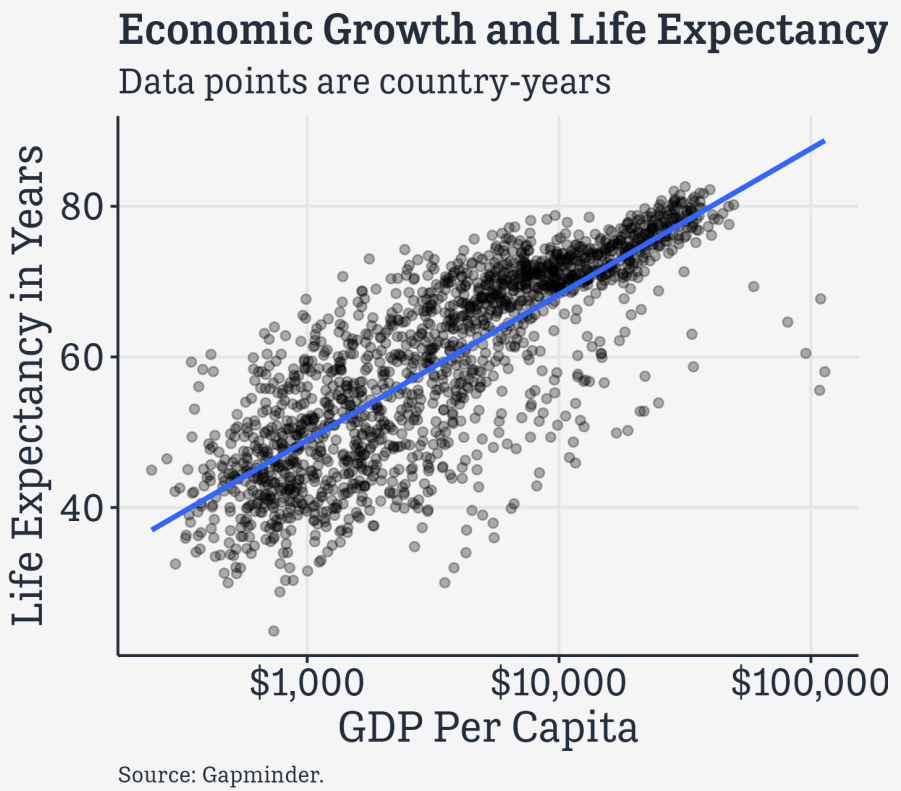
```
gapminder ▷
  ggplot(aes(x = gdpPercap,
             y = lifeExp)) +
  geom_point(alpha = 0.3) +
  geom_smooth(color = "orange",
              se = FALSE,
              linewidth = 8,
              method = "lm") +
  scale_x_log10()
```

Geoms can take many arguments



alpha for overplotting

```
gapminder ▷
  ggplot(aes(x = gdpPercap,
             y = lifeExp)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = FALSE) +
    scale_x_log10(labels = scales::label_dollar()) +
    labs(x = "GDP Per Capita",
         y = "Life Expectancy in Years",
         title = "Economic Growth and Life Expectancy",
         subtitle = "Data points are country-years",
         caption = "Source: Gapminder.")
```



Map or Set values per geom

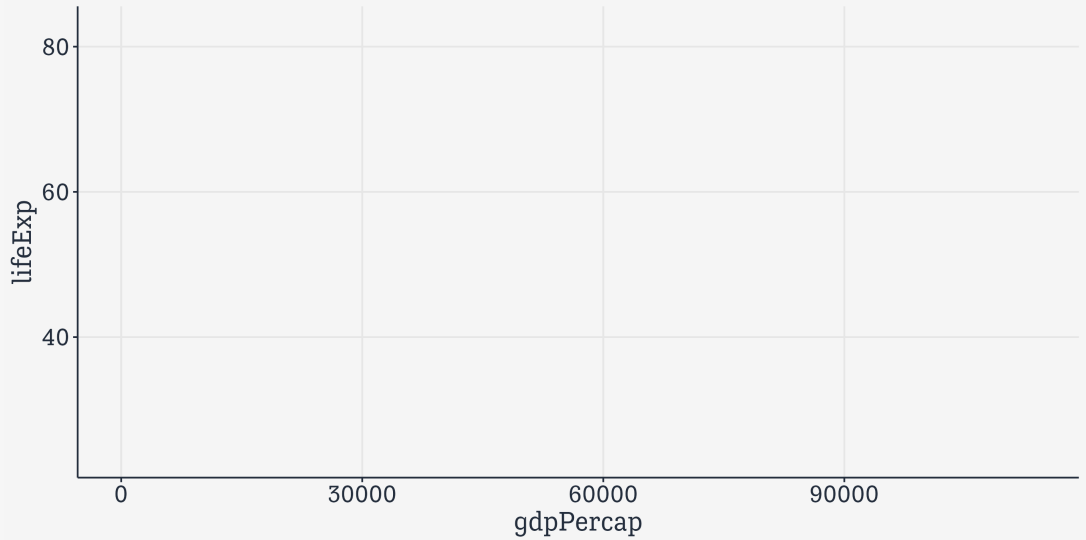
Geoms can take their own mappings

```
gapminder
```

```
# A tibble: 1,704 × 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.
 7 Afghanistan Asia       1982    39.9 12881816      978.
 8 Afghanistan Asia       1987    40.8 13867957      852.
 9 Afghanistan Asia       1992    41.7 16317921      649.
10 Afghanistan Asia       1997    41.8 22227415      635.
# ℹ 1,694 more rows
```

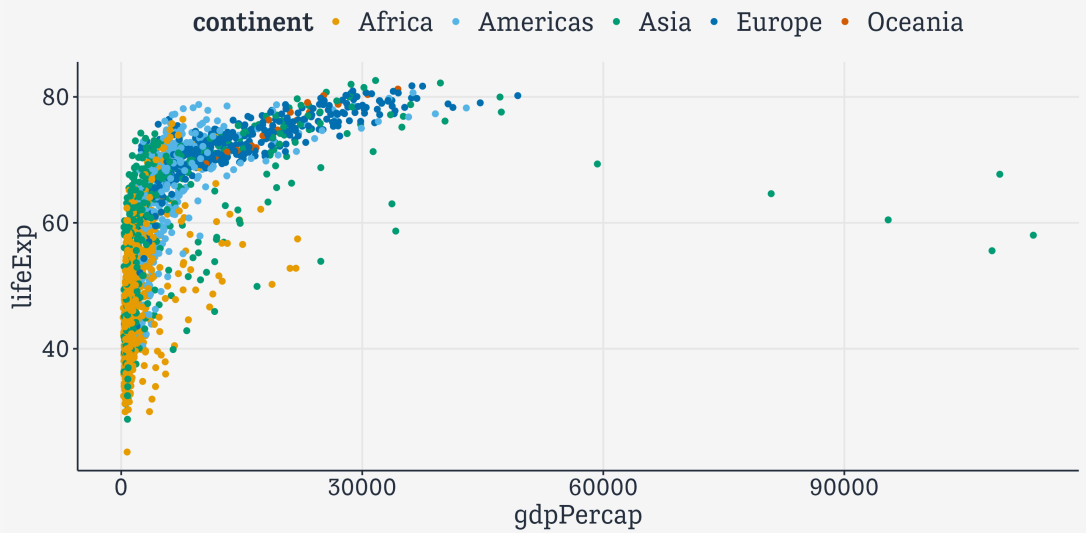
Geoms can take their own mappings

```
gapminder ▷
  ggplot(aes(x = gdpPercap,
             y = lifeExp,
             color = continent,
             fill = continent))
```



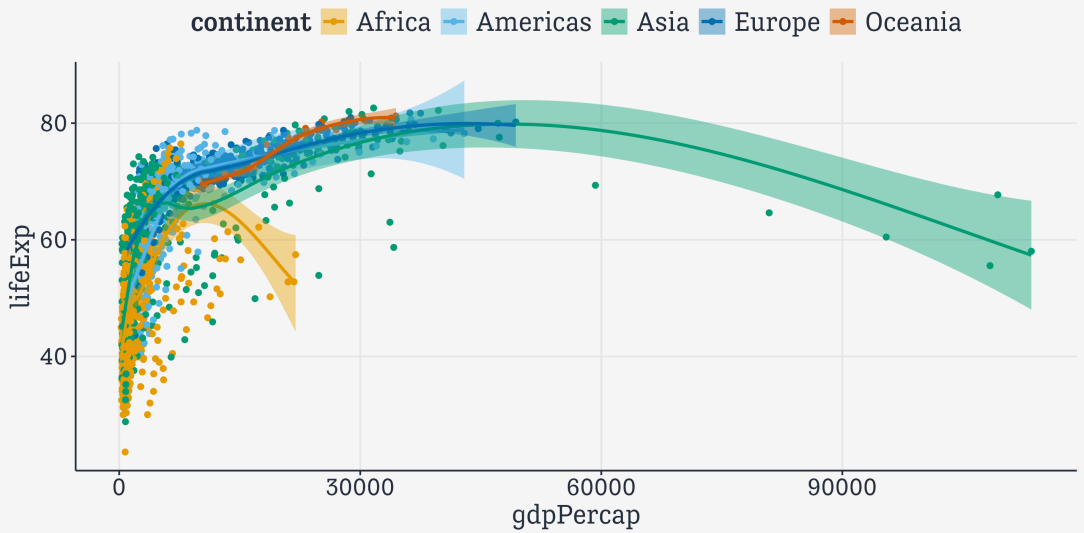
Geoms can take their own mappings

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



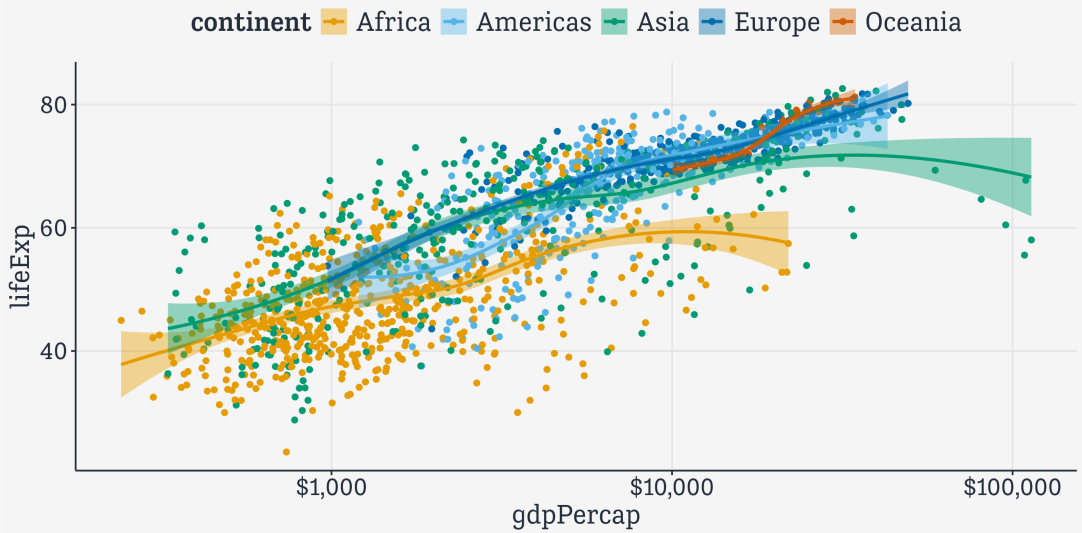
Geoms can take their own mappings

```
gapminder ▷
  ggplot(aes(x = gdpPercap,
             y = lifeExp,
             color = continent,
             fill = continent)) +
  geom_point() +
  geom_smooth(method = "loess")
```



Geoms can take their own mappings

```
gapminder ▷
  ggplot(aes(x = gdpPercap,
             y = lifeExp,
             color = continent,
             fill = continent)) +
  geom_point() +
  geom_smooth(method = "loess") +
  scale_x_log10(labels = scales::label_dollar())
```



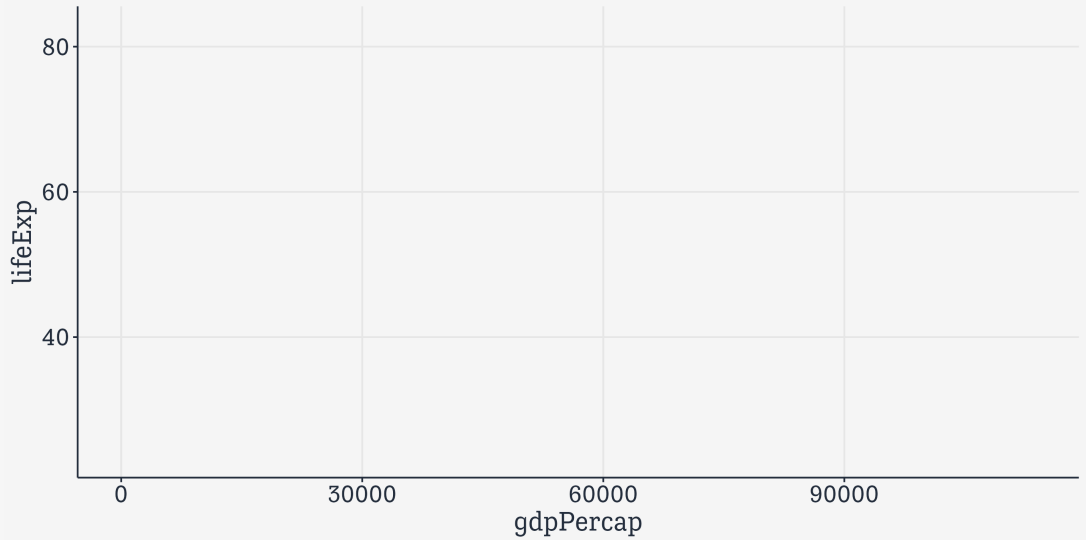
Geoms can take their own mappings

```
gapminder
```

```
# A tibble: 1,704 × 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.
 7 Afghanistan Asia       1982    39.9 12881816      978.
 8 Afghanistan Asia       1987    40.8 13867957      852.
 9 Afghanistan Asia       1992    41.7 16317921      649.
10 Afghanistan Asia       1997    41.8 22227415      635.
# ℹ 1,694 more rows
```

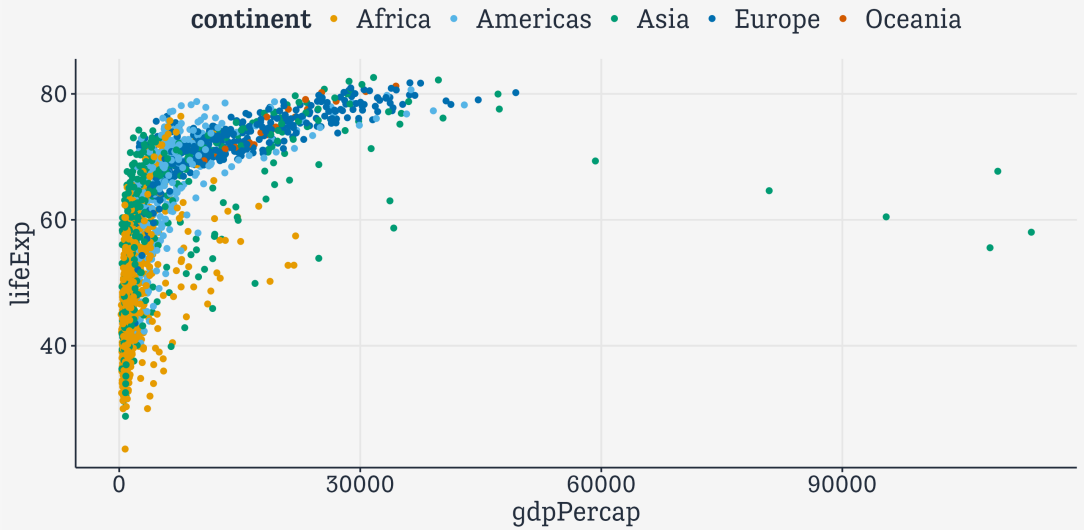
Geoms can take their own mappings

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



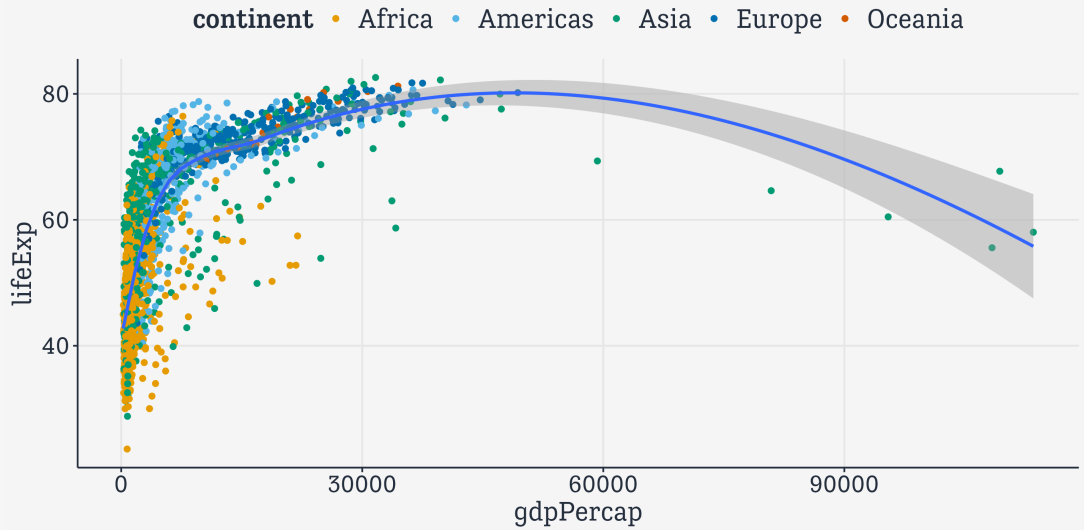
Geoms can take their own mappings

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



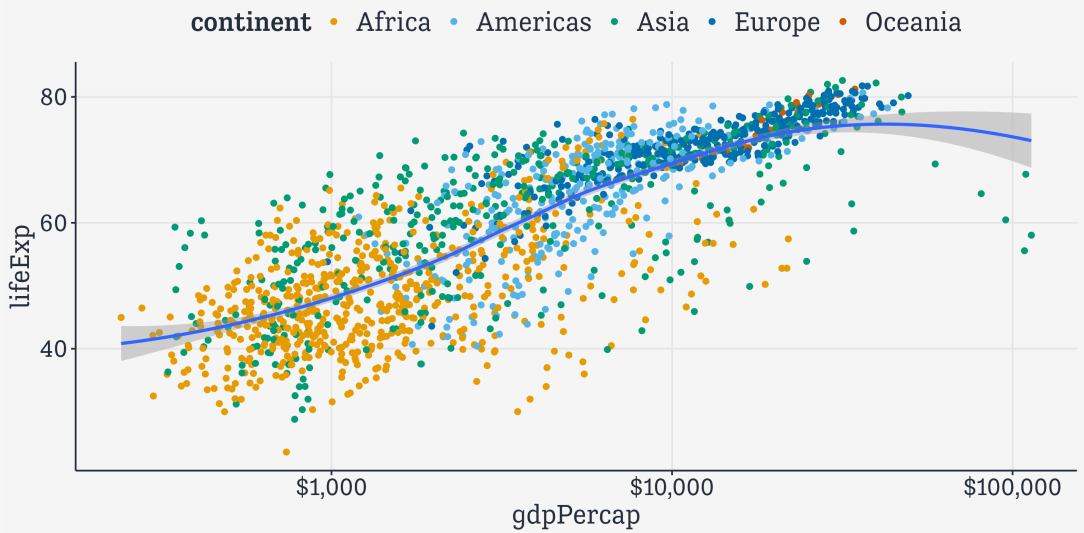
Geoms can take their own mappings

```
gapminder ▷
  ggplot(aes(x = gdpPercap,
             y = lifeExp)) +
  geom_point(mapping = aes(color = continent)) +
  geom_smooth(method = "loess")
```



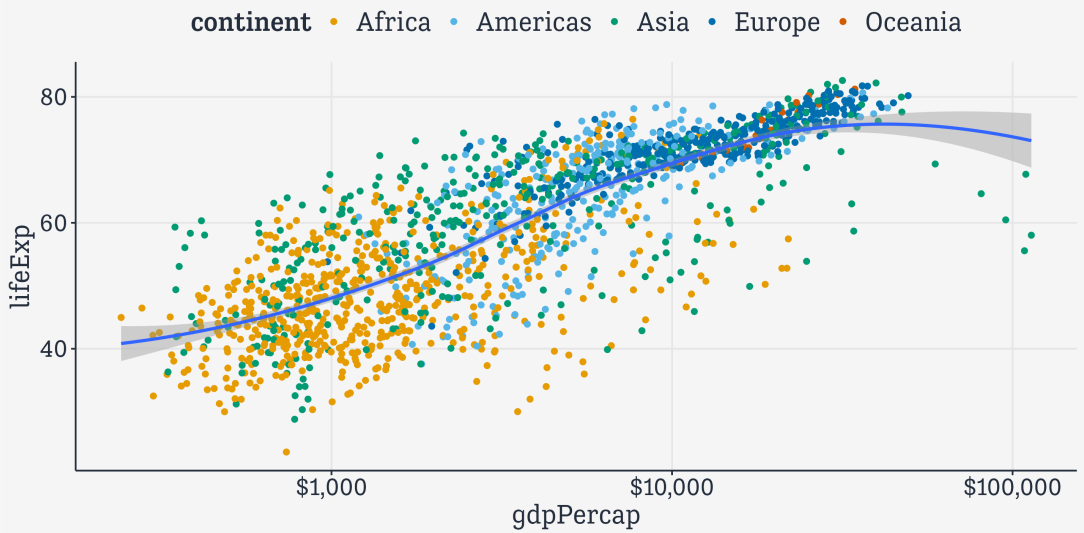
Geoms can take their own mappings

```
gapminder ▷
  ggplot(aes(x = gdpPercap,
             y = lifeExp)) +
  geom_point(mapping = aes(color = continent)) +
  geom_smooth(method = "loess") +
  scale_x_log10(labels = scales::label_dollar())
```



Geoms can take their own mappings

```
gapminder ▷
  ggplot(aes(x = gdpPercap,
             y = lifeExp)) +
  geom_point(mapping = aes(color = continent)) +
  geom_smooth(method = "loess") +
  scale_x_log10(labels = scales::label_dollar())
```



Pay attention to which scales and guides are drawn, and why

Guides and scales reflect `aes()` mappings

```
mapping = aes(color =
continent, fill = continent)
```

Guides and scales reflect `aes()` mappings

```
mapping = aes(color =
continent, fill = continent)
```


```
mapping = aes(color =
              continent)
```

Remember: Every mapped variable has a scale

Arrange geoms with facets

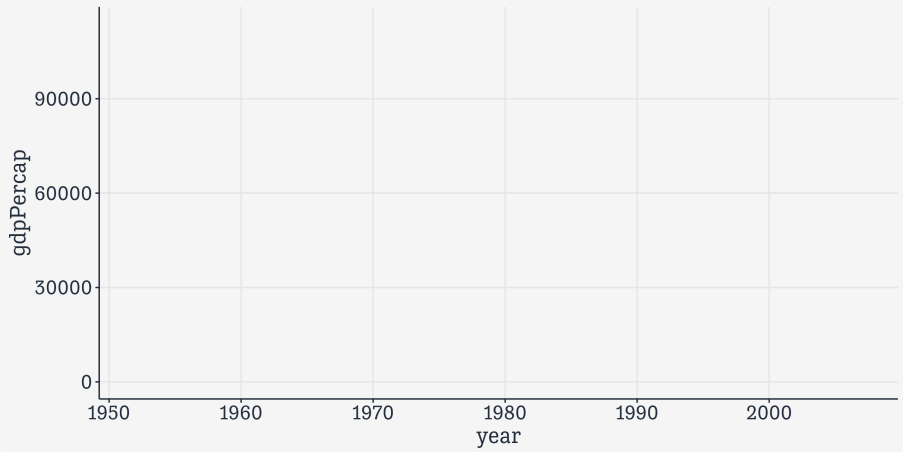
Facet a plot

```
gapminder
```

```
# A tibble: 1,704 × 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.
 7 Afghanistan Asia       1982    39.9 12881816
978.
```

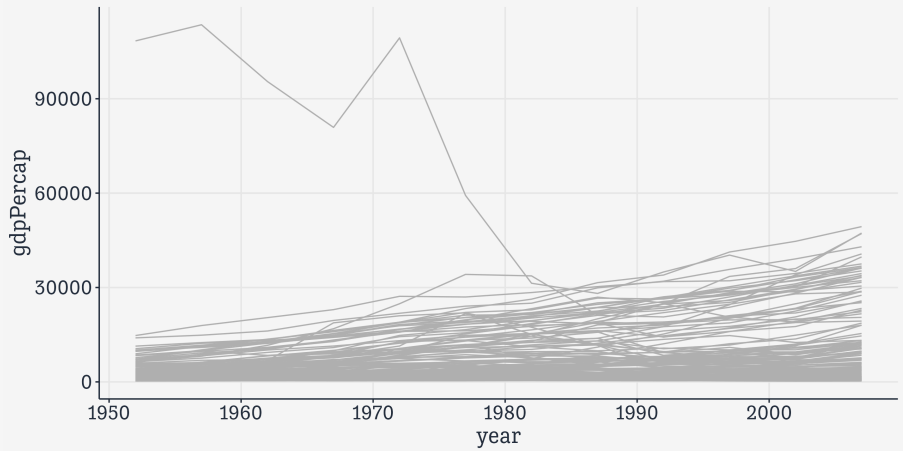
Facet a plot

```
gapminder ▷
  ggplot(aes(x = year,
             y = gdpPercap))
```



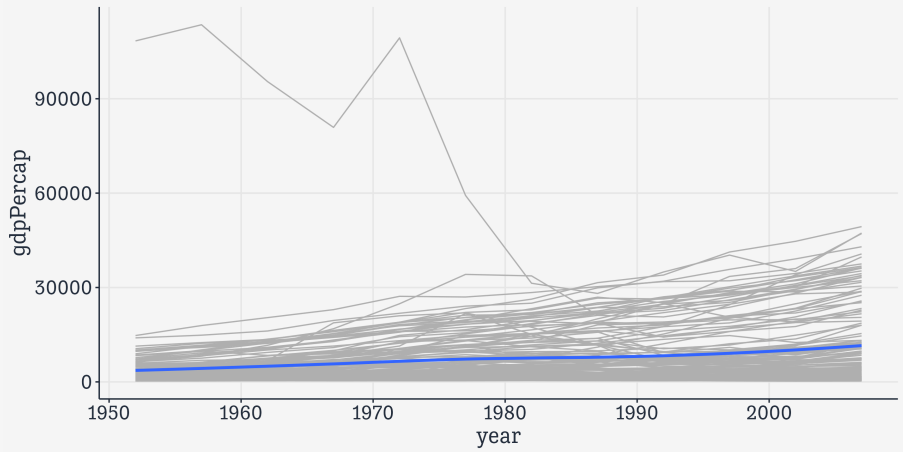
Facet a plot

```
gapminder ▷
  ggplot(aes(x = year,
             y = gdpPercap)) +
  geom_line(color="gray70",
            aes(group = country))
```



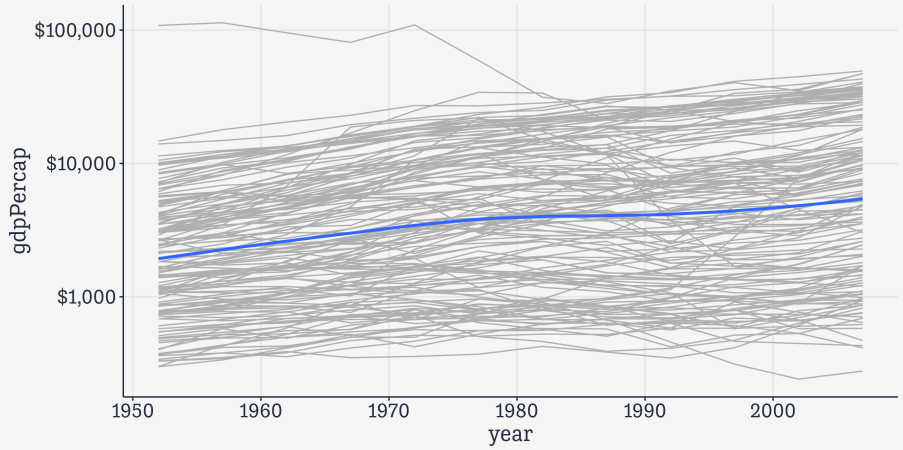
Facet a plot

```
gapminder ▷
  ggplot(aes(x = year,
             y = gdpPercap)) +
  geom_line(color="gray70",
            aes(group = country)) +
  geom_smooth(linewidth = 1.1,
              method = "loess",
              se = FALSE)
```



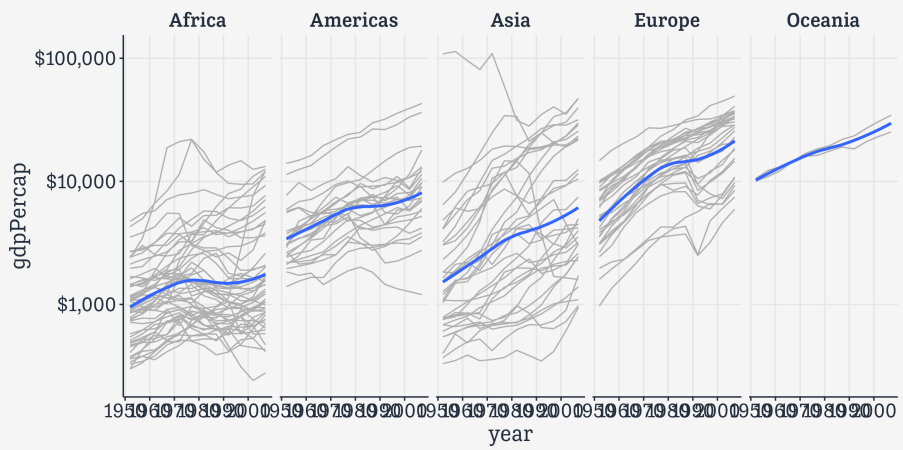
Facet a plot

```
gapminder ▷
  ggplot(aes(x = year,
             y = gdpPercap)) +
  geom_line(color="gray70",
            aes(group = country)) +
  geom_smooth(linewidth = 1.1,
              method = "loess",
              se = FALSE) +
    scale_y_log10(labels=scales::label_dollar())
```



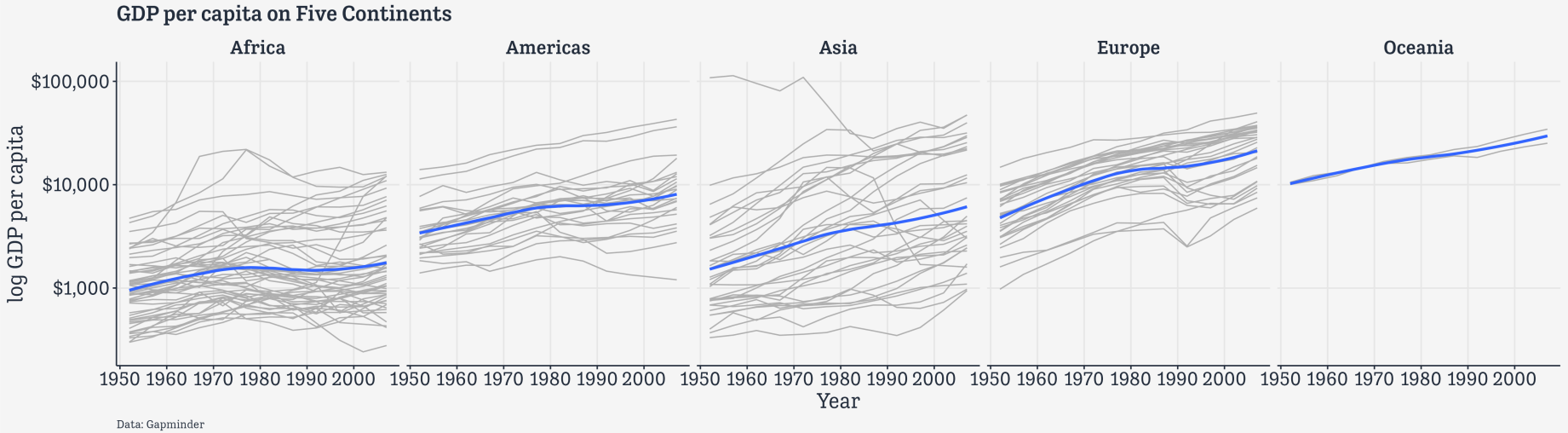
Facet a plot

```
gapminder ▷
  ggplot(aes(x = year,
             y = gdpPercap)) +
  geom_line(color="gray70",
            aes(group = country)) +
  geom_smooth(linewidth = 1.1,
              method = "loess",
              se = FALSE) +
    scale_y_log10(labels=scales::label_dollar()) +
    facet_wrap(~ continent, nrow = 1)
```



Facet a plot

```
gapminder ▷
  ggplot(aes(x = year,
             y = gdpPercap)) +
  geom_line(color="gray70",
            aes(group = country)) +
  geom_smooth(linewidth = 1.1,
              method = "loess",
              se = FALSE) +
    scale_y_log10(labels=scales::label_dollar()) +
    facet_wrap(~ continent, nrow = 1) +
    labs(x = "Year",
         y = "log GDP per capita",
         title = "GDP per capita on Five Continents",
         caption = "Data: Gapminder") → p_out
```



A more polished faceted plot.

Facets are often better than Guides

A GSS Crosstab

```
gss_sm ▷
   select(year, age, race, sex, bigregion, religion)
```

```
# A tibble: 2,867 × 6
    year   age race  sex    bigregion religion
   <dbl> <dbl> <fct> <fct>  <fct>     <fct>
 1  2016    47 White Male   Northeast None
 2  2016    61 White Male   Northeast None
 3  2016    72 White Male   Northeast Catholic
 4  2016    43 White Female Northeast Catholic
 5  2016    55 White Female Northeast None
 6  2016    53 White Female Northeast None
 7  2016    50 White Male   Northeast None
 8  2016    23 Other Female Northeast Catholic
 9  2016    45 Black Male   Northeast Protestant
10  2016    71 White Male   Northeast None
# ℹ 2,857 more rows
```

A GSS Crosstab

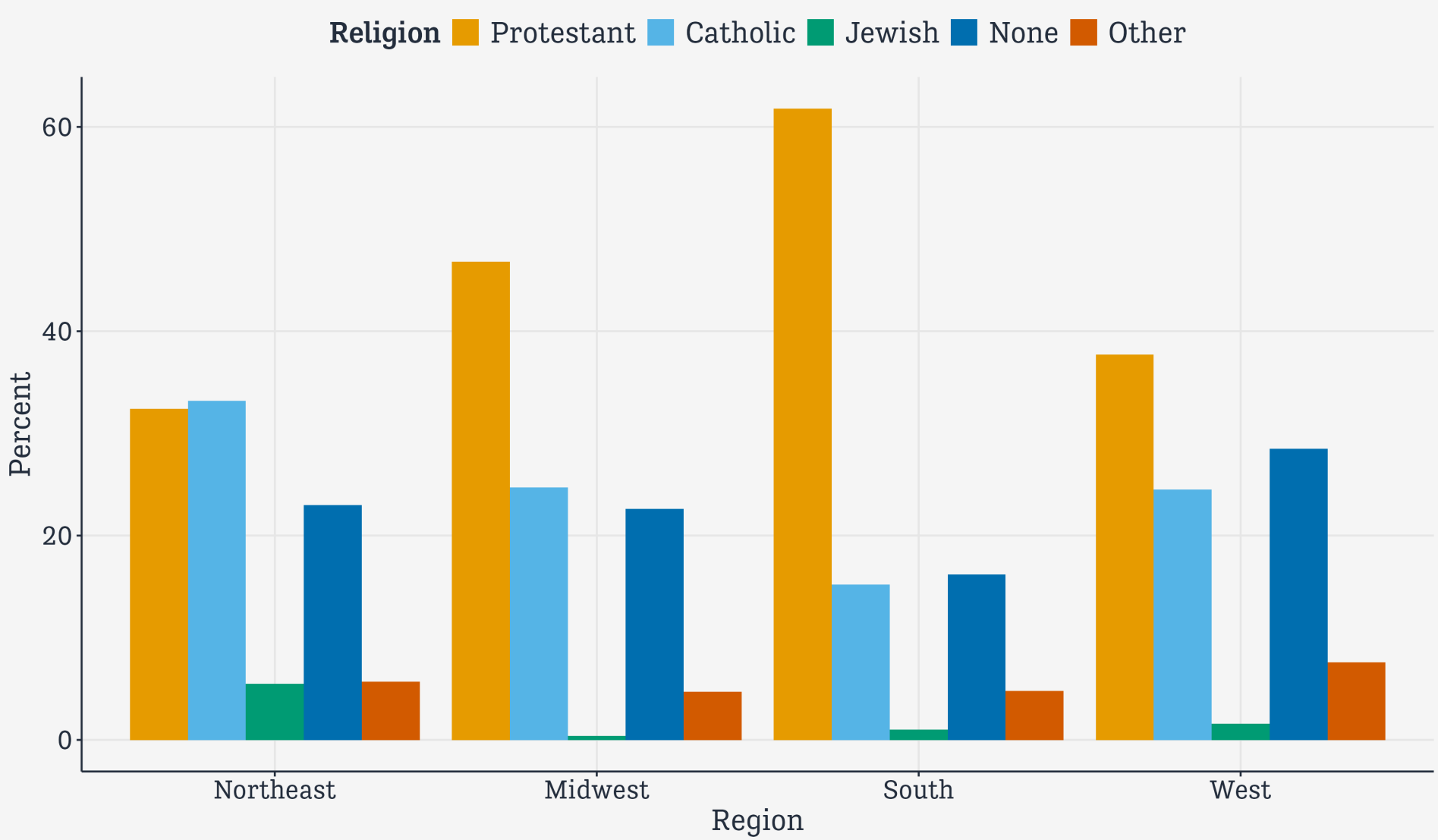
```
rel_by_region ← gss_sm ▷
  group_by(bigregion, religion) ▷
  tally() ▷
  mutate(pct = round((n/sum(n))*100, 1)) ▷
  drop_na()


rel_by_region
```

```
# A tibble: 20 × 4
# Groups:   bigregion [4]
   bigregion religion       n   pct
   <fct>     <fct>      <int> <dbl>
 1 Northeast Protestant   158  32.4
 2 Northeast Catholic     162  33.2
 3 Northeast Jewish        27   5.5
 4 Northeast None         112  23
 5 Northeast Other         28   5.7
 6 Midwest   Protestant   325  46.8
 7 Midwest   Catholic     172  24.7
 8 Midwest   Jewish         3   0.4
 9 Midwest   None         157  22.6
10 Midwest   Other         33   4.7
11 South     Protestant   650  61.8
12 South     Catholic     160  15.2
13 South     Jewish        11   1
14 South     None         170  16.2
15 South     Other         50   4.8
```

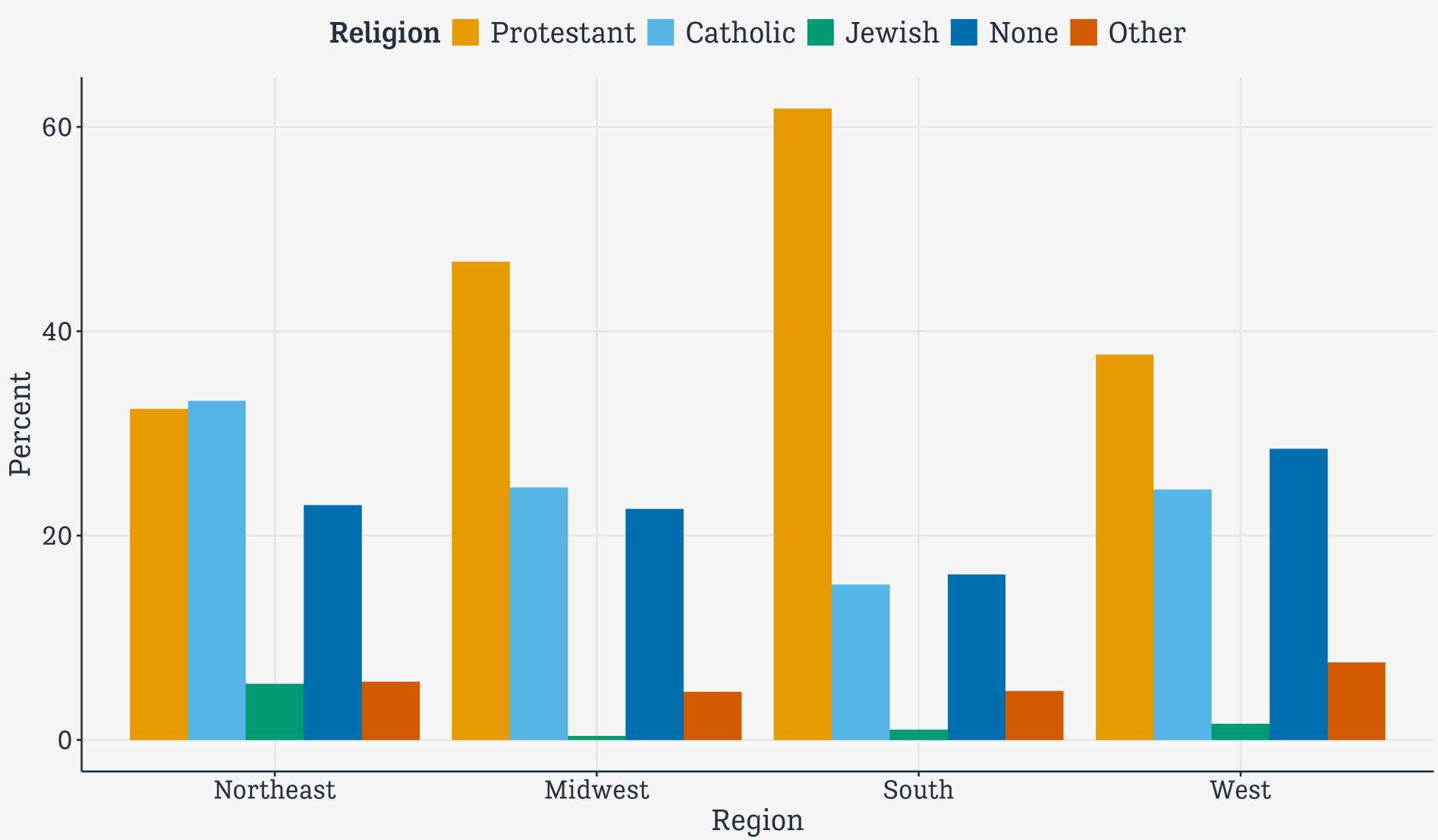
We might write ...

```
p ← ggplot(data = rel_by_region,
               mapping = aes(x = bigregion,
                             y = pct,
                             fill = religion))

p_out ← p + geom_col(position = "dodge") +
    labs(x = "Region", y = "Percent",
         fill = "Religion")
```

We might write ...

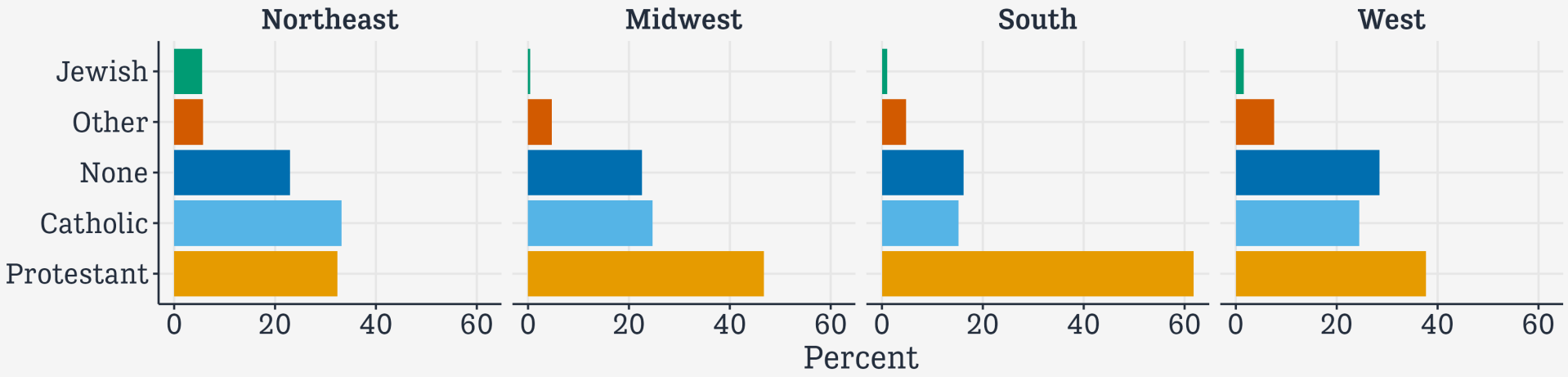


Is this an effective graph? Not really!



Try faceting instead

```
p ← ggplot(data = rel_by_region,
                mapping = aes(x = pct,
                              y = reorder(religion, -pct),
                              fill = religion))
p + geom_col() +
  guides(fill = "none") +
  facet_wrap(~ bigregion, nrow = 1) +
  labs(x = "Percent", y = NULL)
```



Putting categories on the y-axis is a very useful trick. Think of your graph as being like a table. Tabular or grid layouts like this are very compact.

Try faceting instead

Try putting categories on the y-axis. (And reorder them by x.)

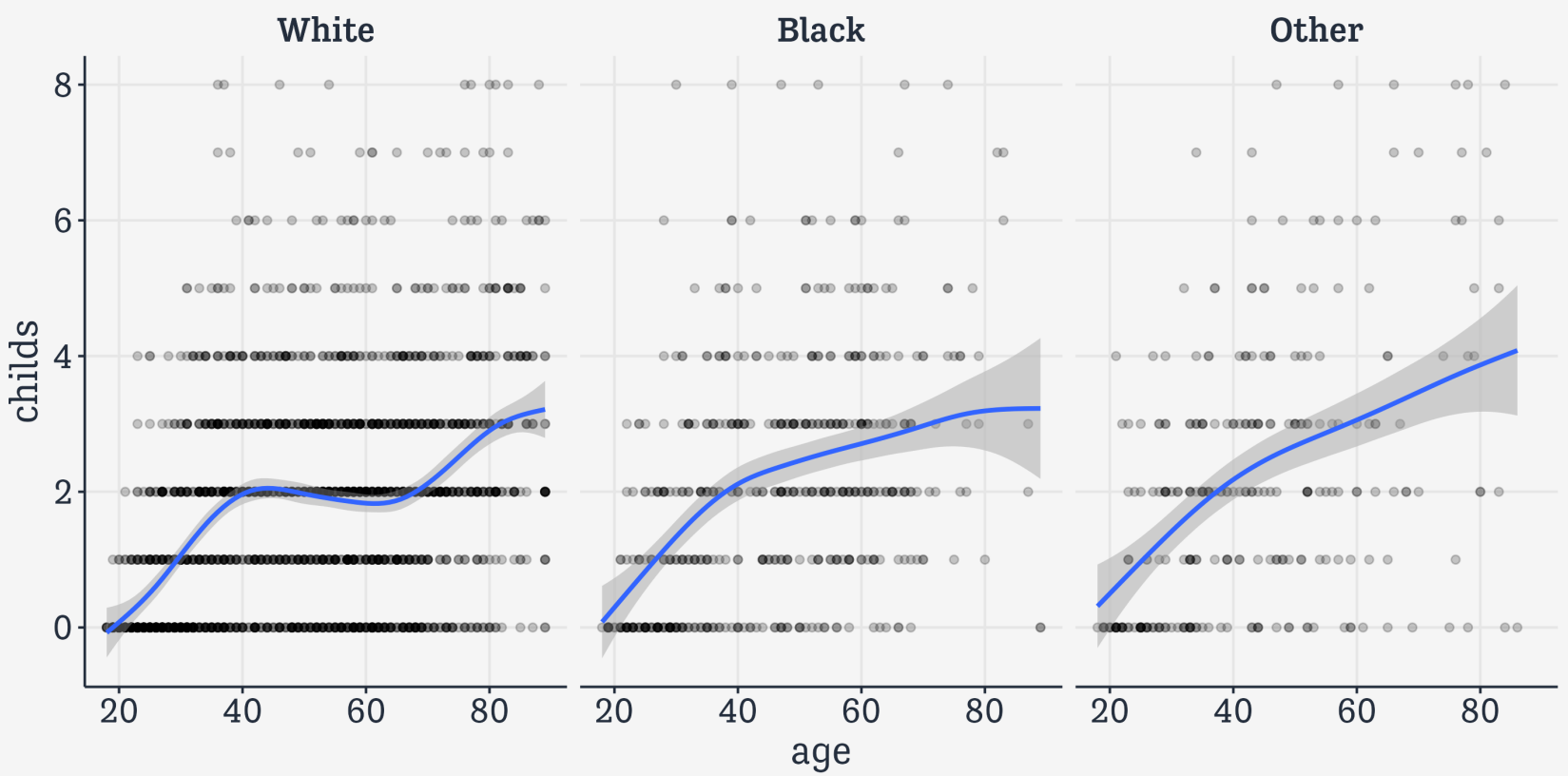
Try faceting variables instead of mapping them to color or shape.

Try to minimize the need for guides and legends.

Two kinds of facet

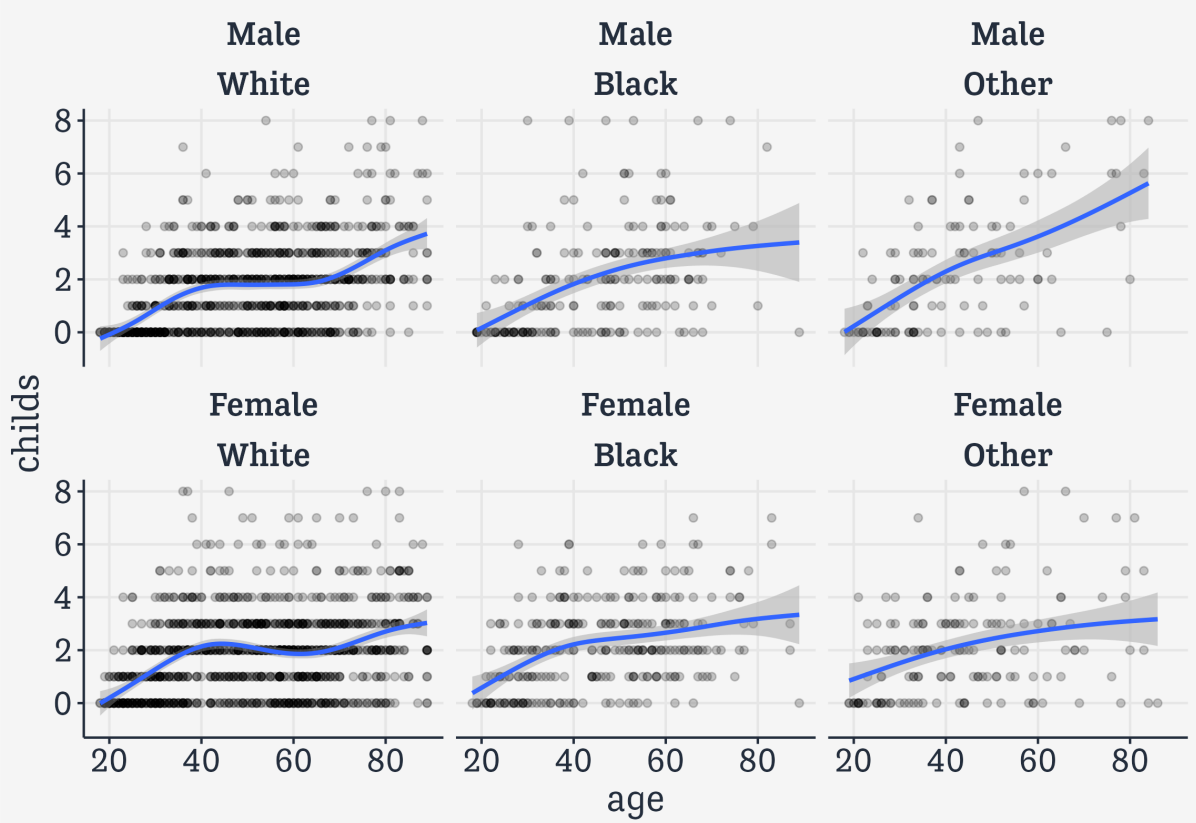
Facet Children vs Age, by Race

```
p ←  ggplot(data = gss_sm,
            mapping = aes(x = age, y = childs))

p + geom_point(alpha = 0.2) +
  geom_smooth() +
  facet_wrap(~ race)
```



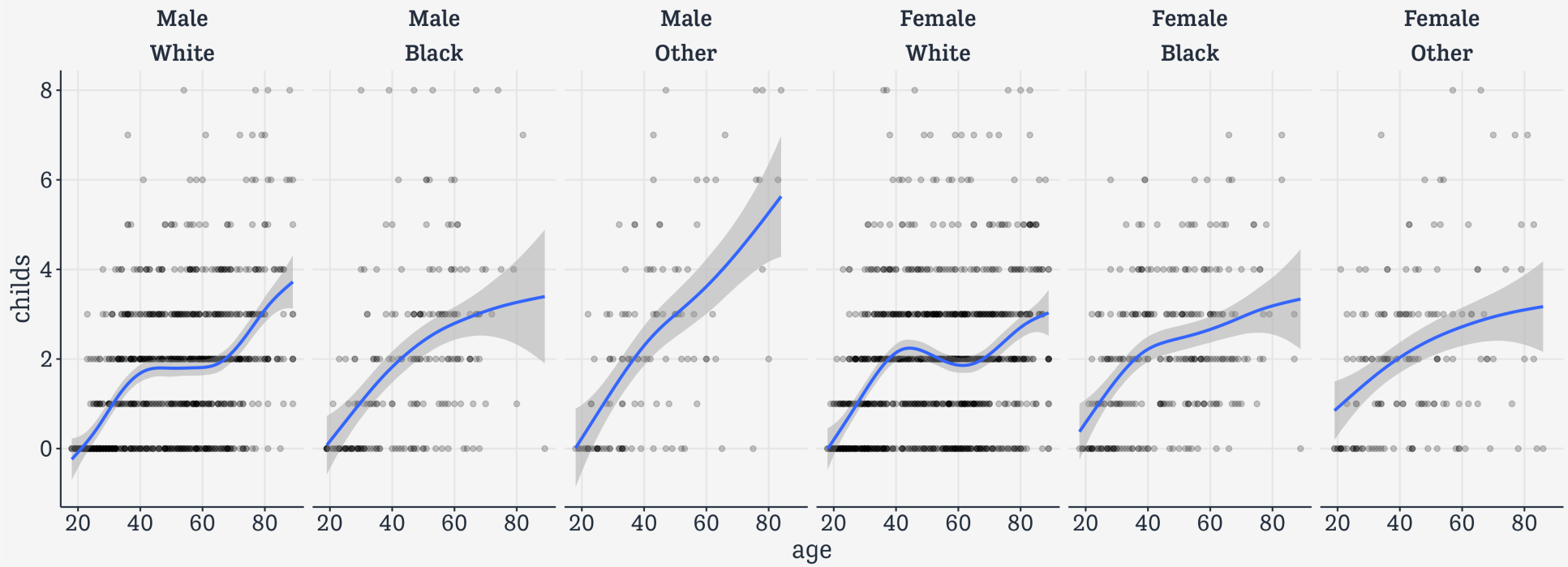
Facet by more than one variable

```
p ←  ggplot(data = gss_sm,
            mapping = aes(x = age, y = childs))

p + geom_point(alpha = 0.2) +
  geom_smooth() +
  facet_wrap(~ sex + race)
```



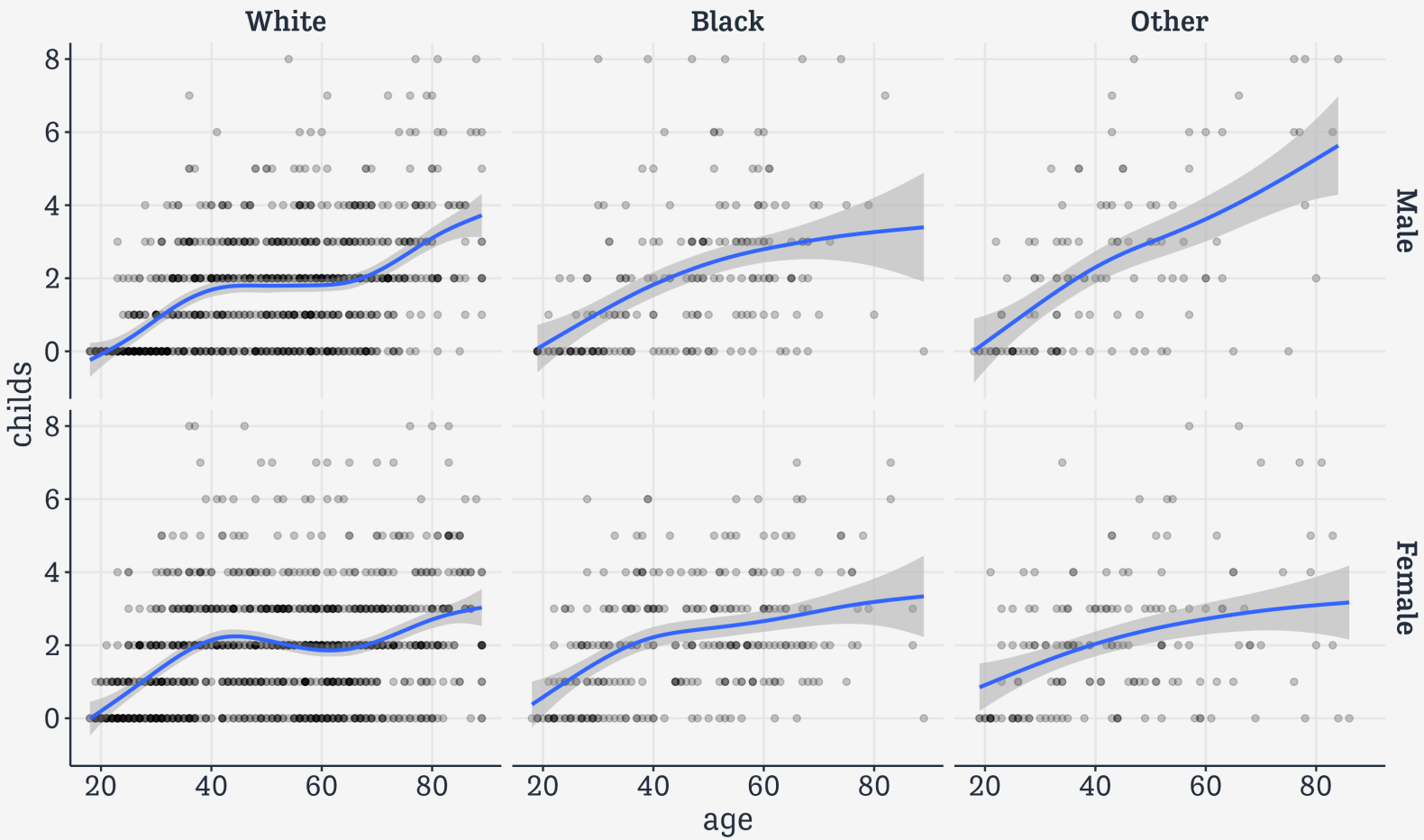
Arrange facet_wrap() quite freely

```
p ←  ggplot(data = gss_sm,
            mapping = aes(x = age, y = childs))

p + geom_point(alpha = 0.2) +
  geom_smooth() +
  facet_wrap(~ sex + race, nrow = 1)
```



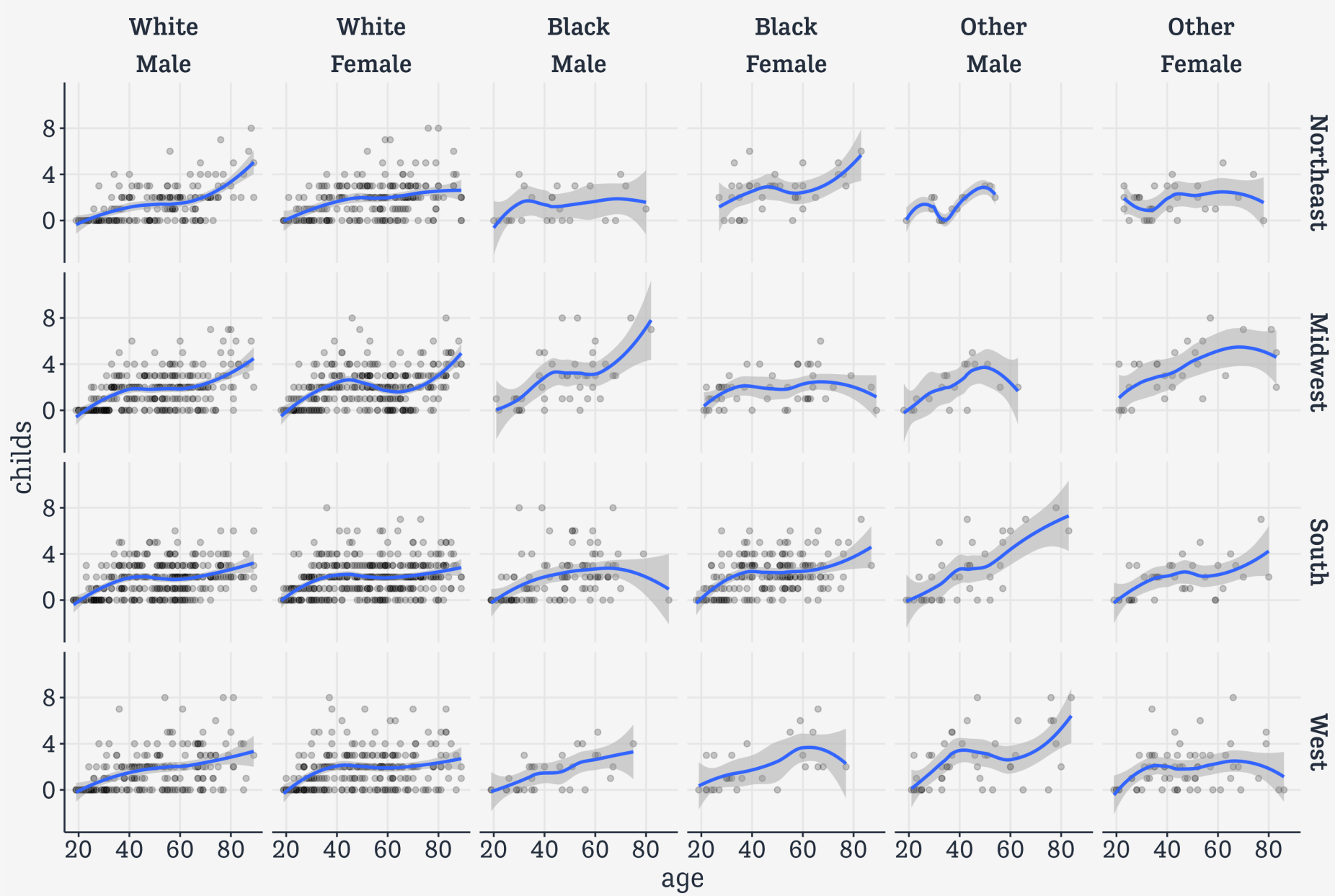
facet_grid() is more like a true crosstab

```
p + geom_point(alpha = 0.2) +
  geom_smooth() +
  facet_grid(sex ~ race)
```



Extend both to multi-way views

```
p_out ← p + geom_point(alpha = 0.2) +
  geom_smooth() +
  facet_grid(bigregion ~ race + sex)
```


White Male
White Female
Black Male
Black Female
Other Male
Other Female
Northeast
Midwest
South
West
childs
age
0
4
8
20
40
60
80