

Making a plot

Data Visualisation mini-course

Gavin L. Simpson

Institute of Environmental Change and Society

07/04/2019

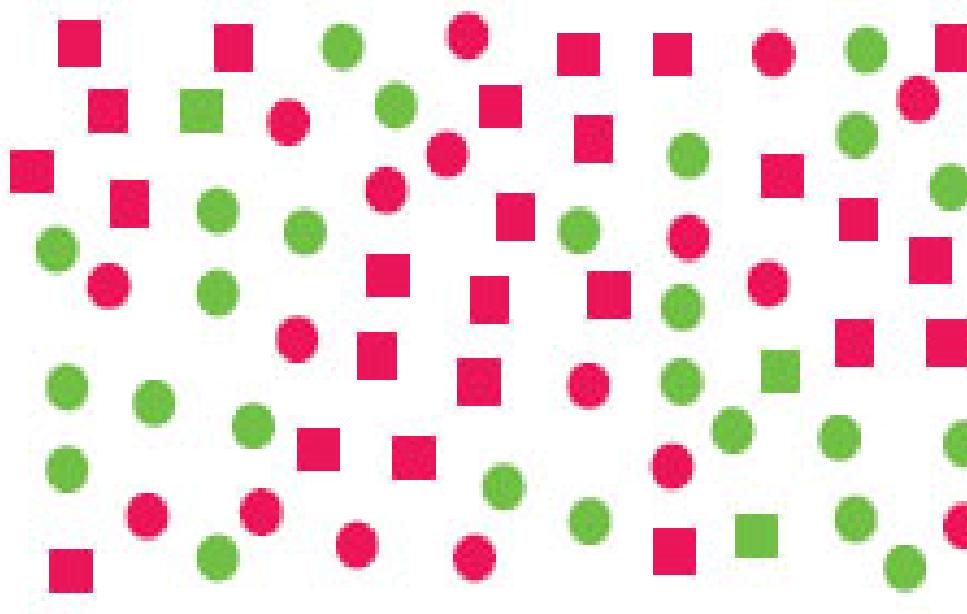
Pop



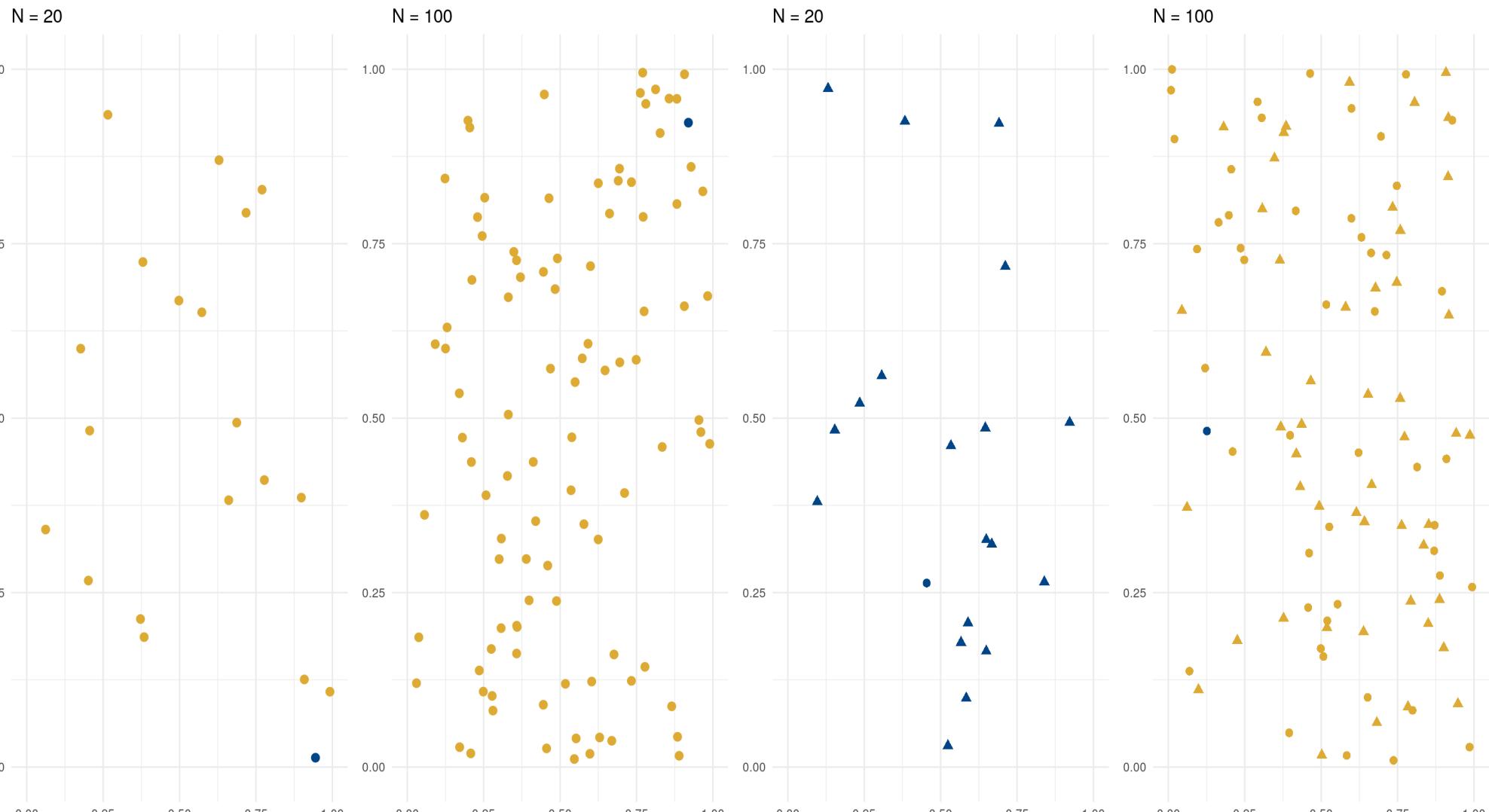
Cody Davis

Pop

Can you see the green squares?



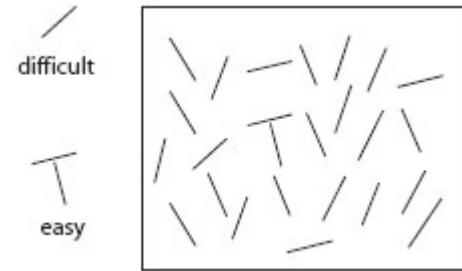
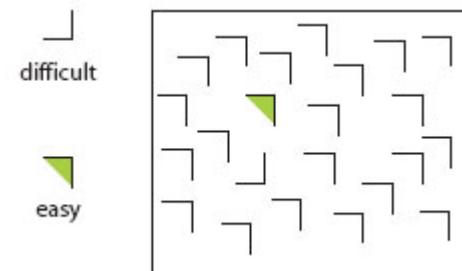
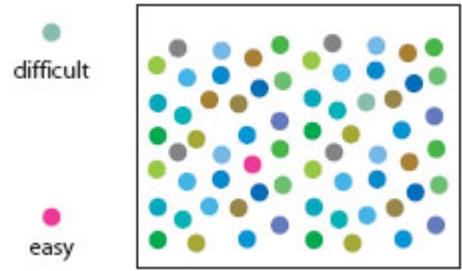
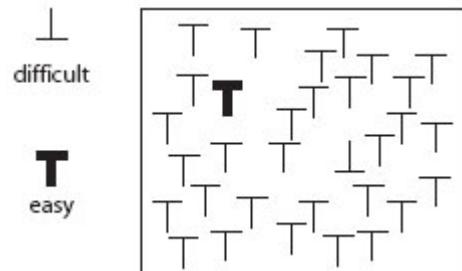
Pop



Preattentive pop-out

Some shapes, colours, angles more easy to spot

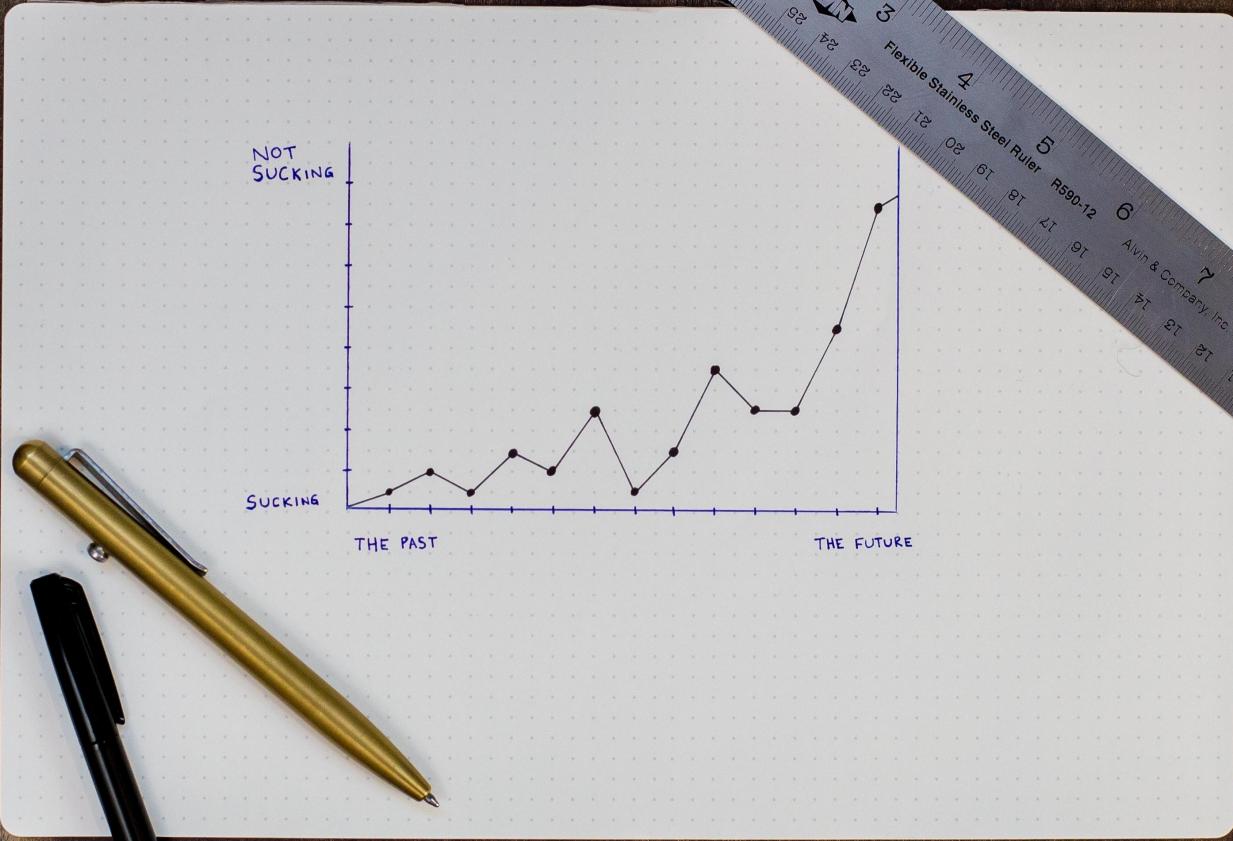
Can happen before (or almost before) before consciously looking at something



Making a plot

WEEKLY PLANNER

Isaac Smith



The anatomy of a plot

Visualisation involves representing data by lines, shapes, colours, etc.

Map data to visual channels – some channels more effective than others

ggplot provides a set of tools to

- map data to visual elements,
- specify the kind of plot, and
- control the fine details of the final plot

The anatomy of a *ggplot*

A *ggplot* comprises several main elements

1. Data
2. Aesthetic mappings
3. Geoms
4. Co-ordinates & scales
5. Labels & guides

ggplot()

Main function is `ggplot()`

- specify `data`, the data frame containing the data
- specify mappings of variables in `data` to `aesthetics` with `aes()`

Add *layers* to plot via `+`

Geoms are the main layer-types we add to influence the plot

Geoms by default inherit the `data` and `aesthetics` from the `ggplot()` call

```
ggplot(data_frame, aes(x = var1, y = var2, colour = var3)) +  
  geom_<type>(....) +  
  geom_<type>(....)
```

Data

Two main ways in which data tend to be recorded

1. *wide*-format
2. *long*-format

In *long*-format:

- every column is a variable
- every row is an observation

In *wide*-format

- some variables are spread out over multiple columns

ggplot requires data in *long* form

Wide-format

```
## # A tibble: 142 x 13
##   country `1952` `1957` `1962` `1967` `1972` `1977` `1982` `1987` `1992`
##   <fct>     <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 Afghan...  28.8    30.3    32.0    34.0    36.1    38.4    39.9    40.8    41.7
## 2 Albania   55.2    59.3    64.8    66.2    67.7    68.9    70.4    72       71.6
## 3 Algeria   43.1    45.7    48.3    51.4    54.5    58.0    61.4    65.8    67.7
## 4 Angola    30.0    32.0    34       36.0    37.9    39.5    39.9    39.9    40.6
## 5 Argent...  62.5    64.4    65.1    65.6    67.1    68.5    69.9    70.8    71.9
## 6 Austra...  69.1    70.3    70.9    71.1    71.9    73.5    74.7    76.3    77.6
## 7 Austria   66.8    67.5    69.5    70.1    70.6    72.2    73.2    74.9    76.0
## 8 Bahrain   50.9    53.8    56.9    59.9    63.3    65.6    69.1    70.8    72.6
## 9 Bangla...  37.5    39.3    41.2    43.5    45.3    46.9    50.0    52.8    56.0
## 10 Belgium  68       69.2    70.2    70.9    71.4    72.8    73.9    75.4    76.5
## # ... with 132 more rows, and 3 more variables: `1997` <dbl>, `2002` <dbl>,
## #       `2007` <dbl>
```

Long-format

```
## # A tibble: 1,704 x 3
##   country     year lifeExp
##   <fct>       <int>   <dbl>
## 1 Afghanistan 1952    28.8
## 2 Afghanistan 1957    30.3
## 3 Afghanistan 1962    32.0
## 4 Afghanistan 1967    34.0
## 5 Afghanistan 1972    36.1
## 6 Afghanistan 1977    38.4
## 7 Afghanistan 1982    39.9
## 8 Afghanistan 1987    40.8
## 9 Afghanistan 1992    41.7
## 10 Afghanistan 1997   41.8
## # ... with 1,694 more rows
```

Long-format

Data arranged in *key* and *value* pairs

`year` is the *key*

`lifeExp` is the *value*

```
## # A tibble: 5 x 3
##   country     year lifeExp
##   <fct>     <int>   <dbl>
## 1 Afghanistan 1952     28.8
## 2 Afghanistan 1957     30.3
## 3 Afghanistan 1962     32.0
## 4 Afghanistan 1967     34.0
## 5 Afghanistan 1972     36.1
```

Long-format

Can have multiple *key* and *value* columns

```
## # A tibble: 5 x 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.
```

RStudio tour



▪ Jeffrey Buchbinder

gapminder data set

```
library('gapminder') # load package  
gapminder # print data frame
```

```
## # A tibble: 1,704 x 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.  
## # ... with 1,694 more rows
```

Our first plot

Say we want to plot life expectancy (`lifeExp`) against per capita GDP `gdpPercap`

```
p <- ggplot(data = gapminder)
```

We tell `ggplot()` *where* to look for variables, but haven't specified any mappings yet

Assigned the output of the `ggplot()` call to the object `p` (could call `p` anything)

Alt + -

or

Option + -

types the *assignment operator* `<-`

Our first plot

We specify mappings between variables and aesthetics via the `mapping` argument

Use the `aes()` function to specify the mappings

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

This sets up a mapping between our two variables and the `x` and `y` aesthetics

The `x` and `y` aesthetics are the *x* and *y* coordinates of the plot

Our first plot

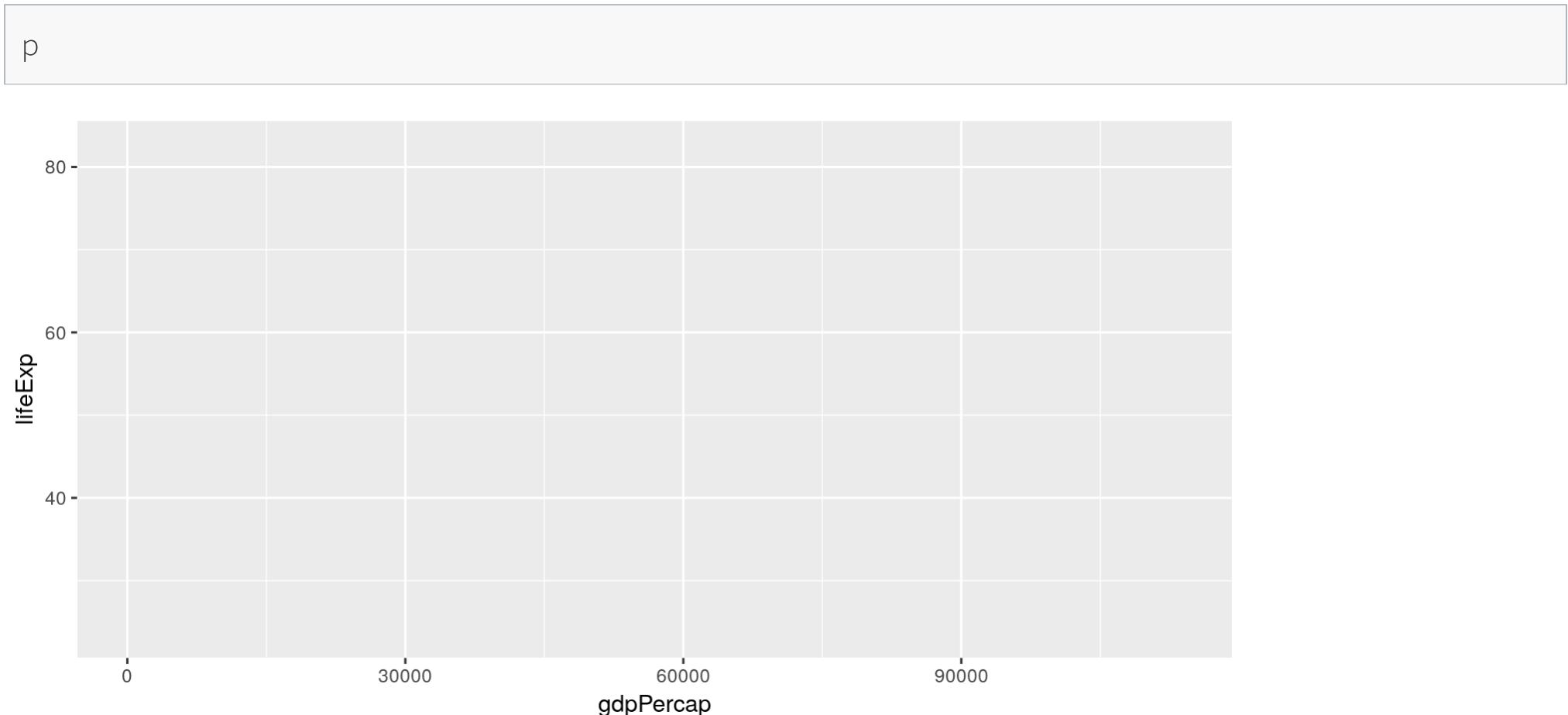
We can draw the plot by `print()`ing the object `p`

What do you think you'll get if you print `p`?

```
p
```

Our first plot

Only the *scale* for the x and y aesthetics is drawn



Adding a layer

Need to tell `ggplot()` *how* we want the data drawn

Need to choose a *geometric object* or *geom*

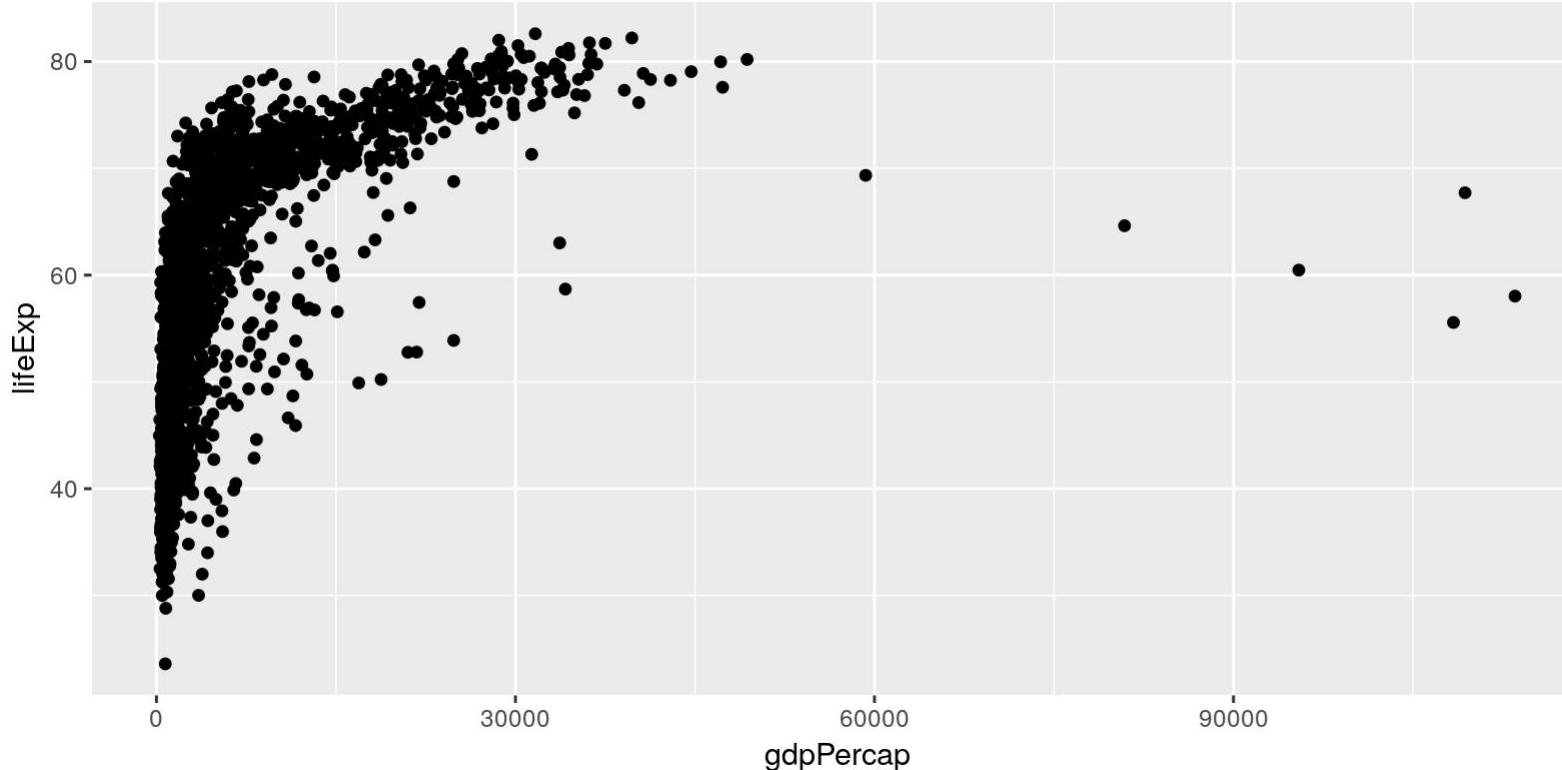
geoms are functions with names `geom_<type>()`

A *geom* adds a layer to an existing plot

For a scatterplot, we represent the x, y pairs via points `geom_point()`

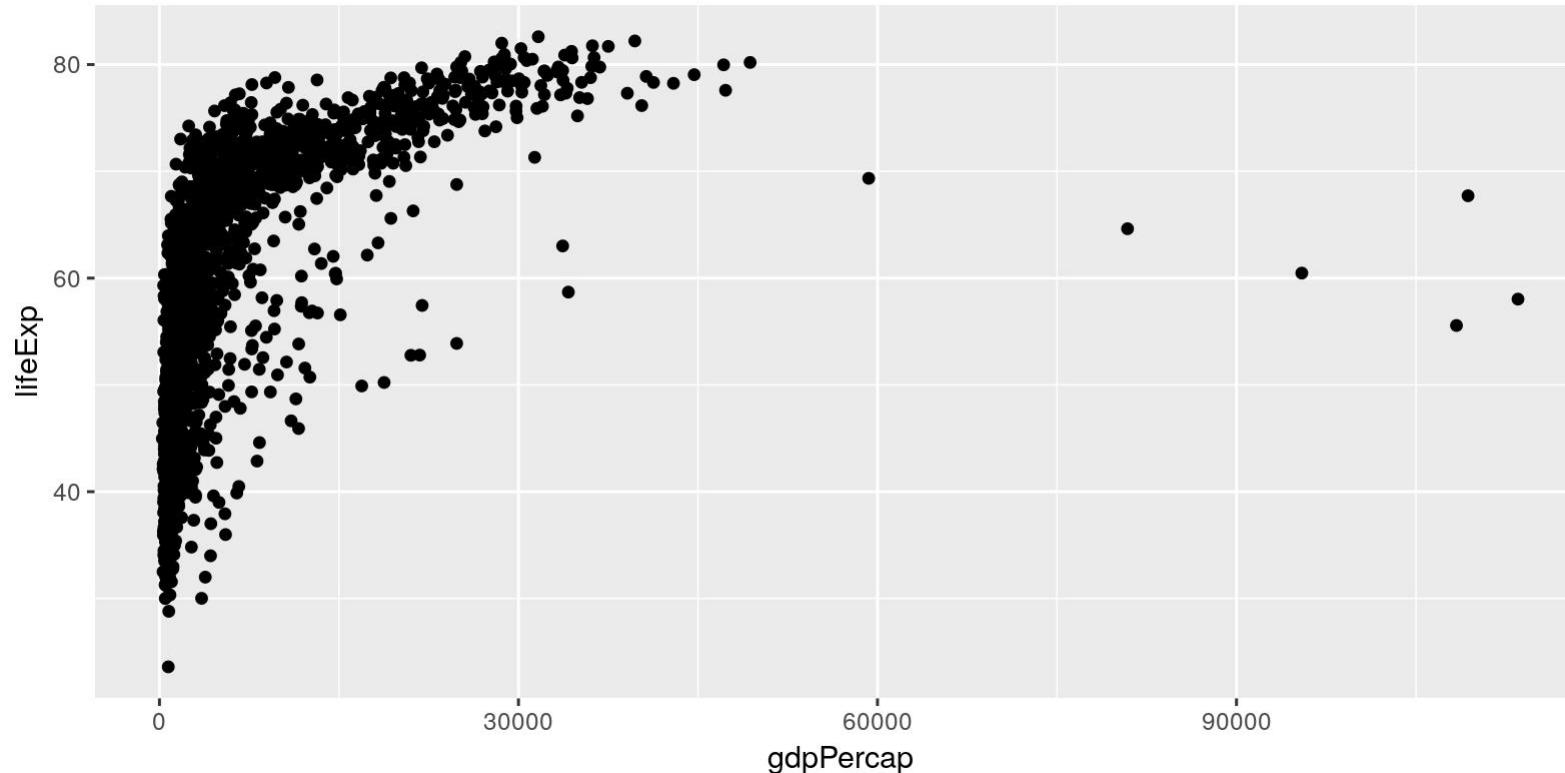
Adding a layer

```
p + geom_point()
```



Putting it all together

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



A photograph of a person climbing an indoor rock wall. The climber is wearing a blue t-shirt, green shorts, and a blue harness. They are positioned in the upper right quadrant, facing towards the left. The wall is covered in various colored holds (blue, red, green, yellow) of different shapes. A bright beam of light from a projector illuminates the climber's path. The background is dark.

Your turn

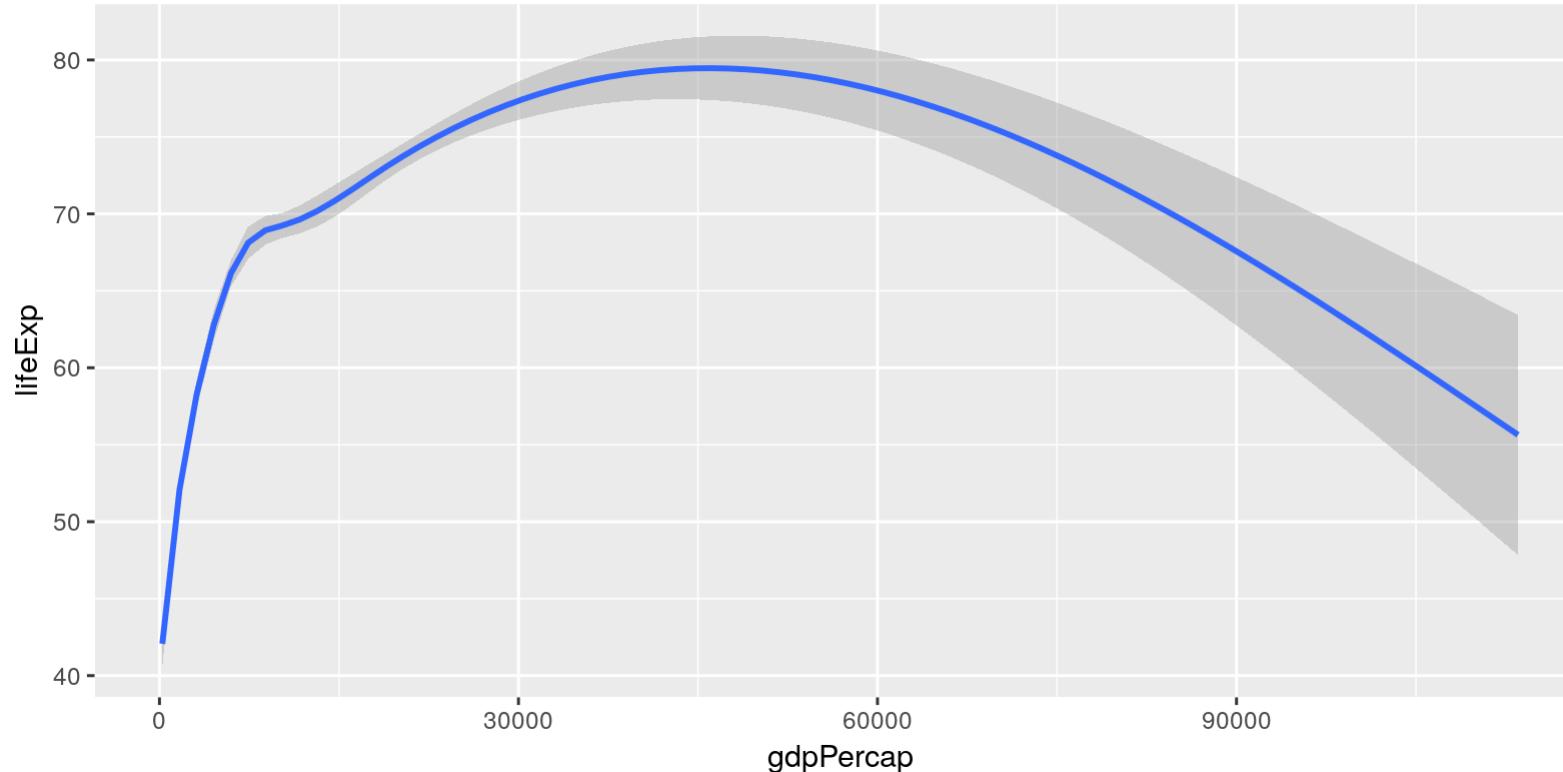
Create a scatterplot of *population* and *per capita GDP*

FancyCrave

geoms don't always draw the data

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

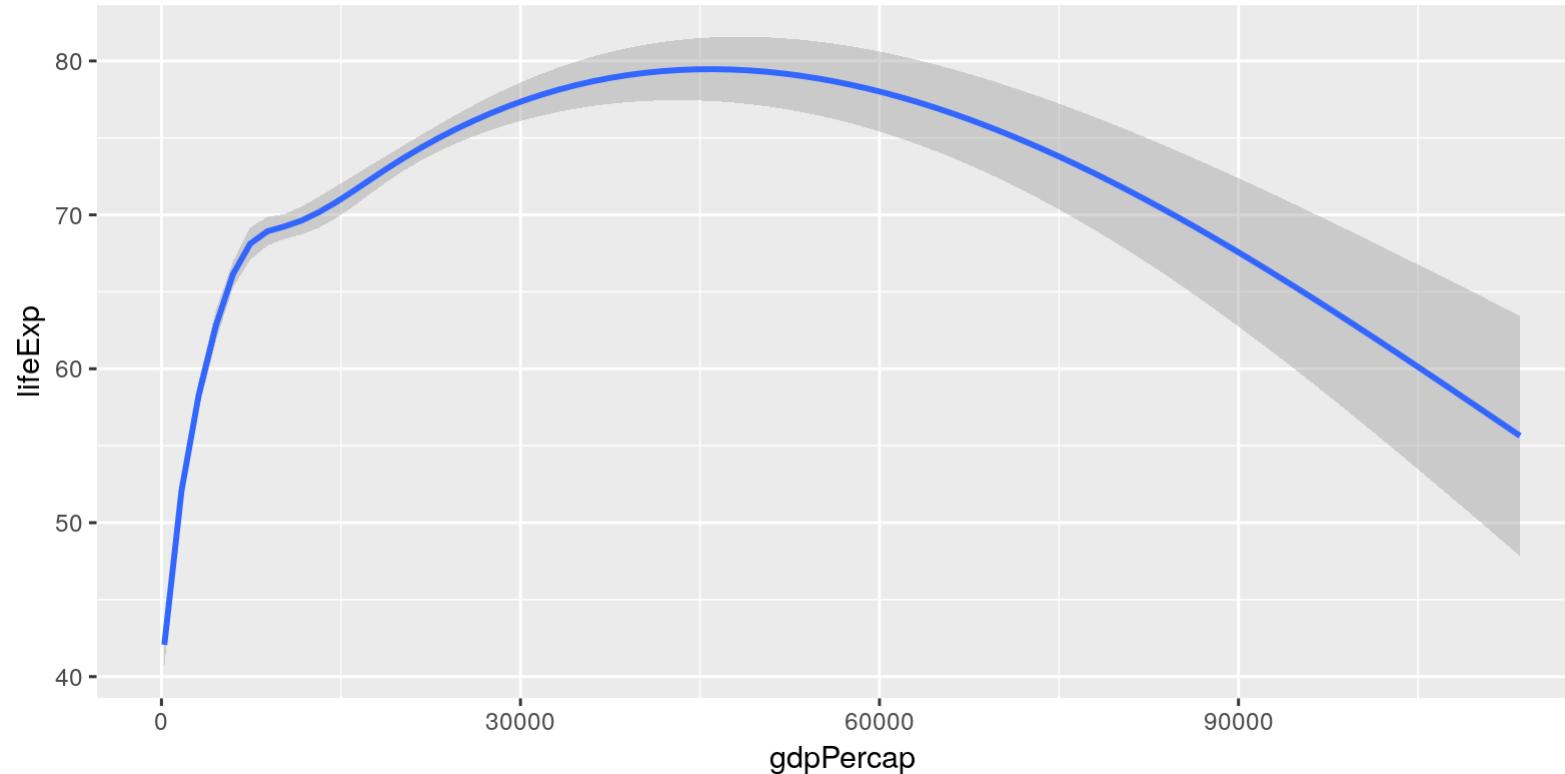
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



`geom_smooth()` adds a smooth via a GAM or LOESS

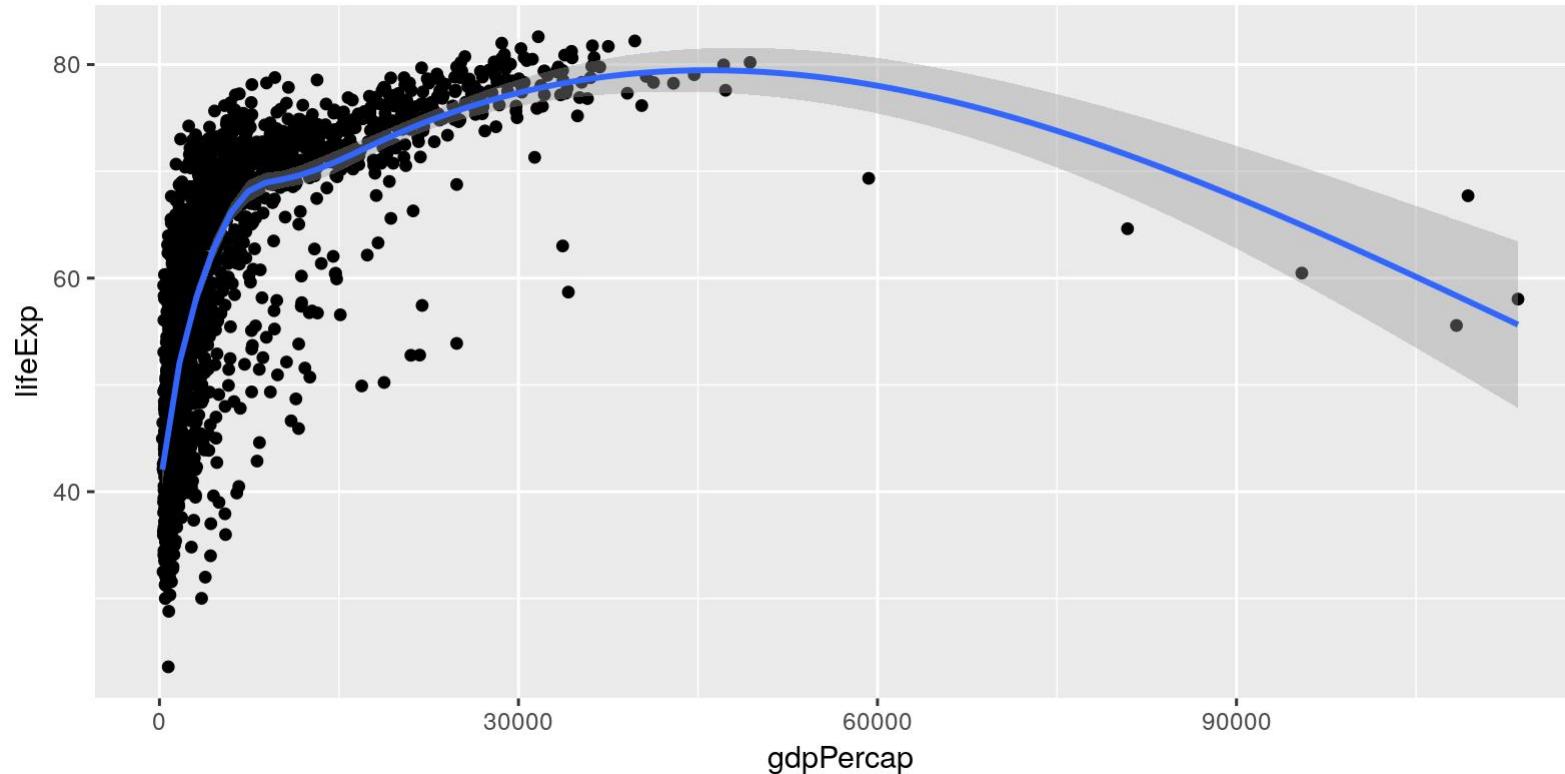
```
p <- ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp))  
p + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Plots with multiple layers

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



Geoms inherit data and mappings

Didn't need to tell each geom what data or mappings to use

Information is *inherited* from the main `ggplot()` object

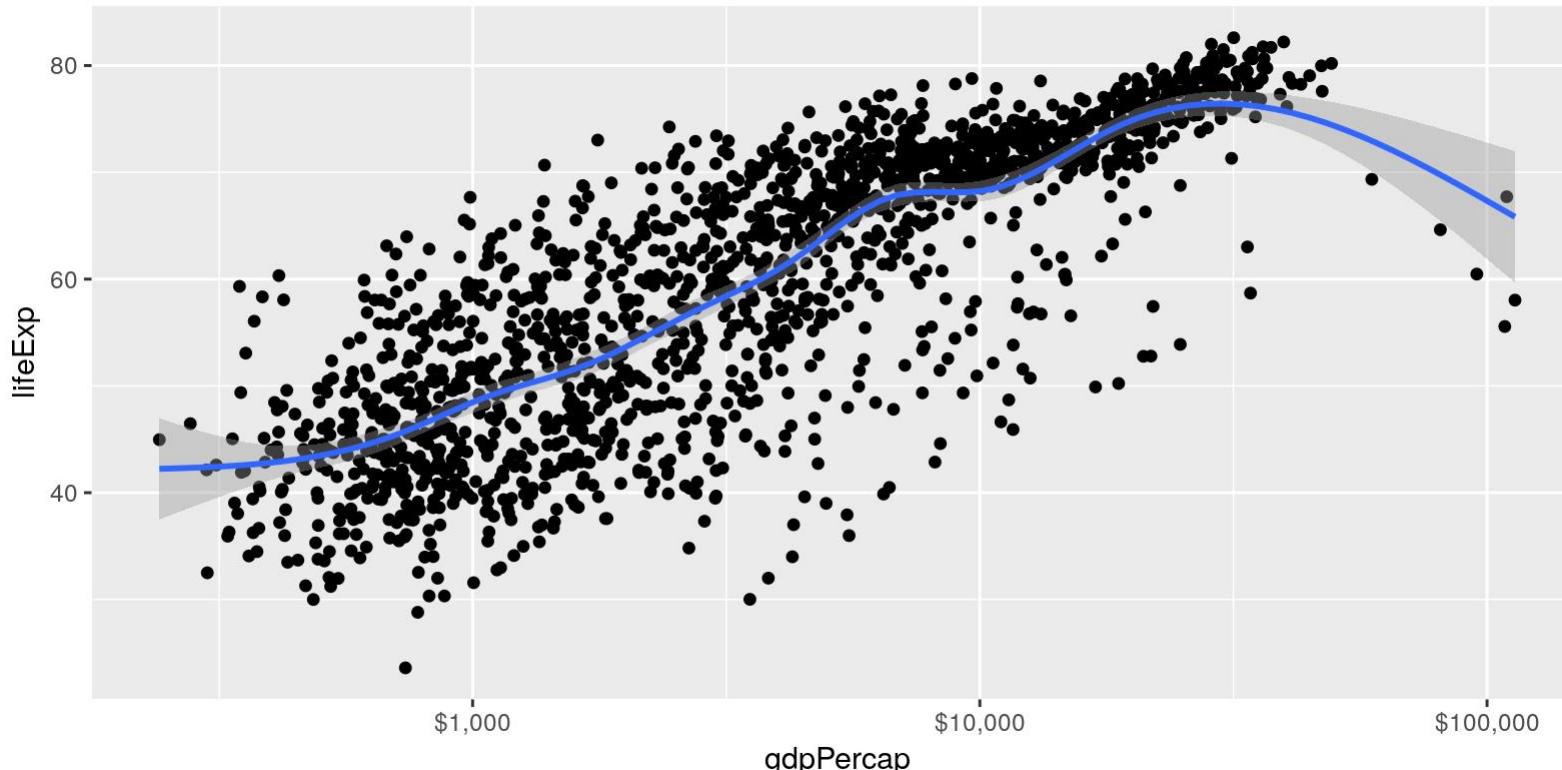
Can override this

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

scale_<aes>_<type>() control scales

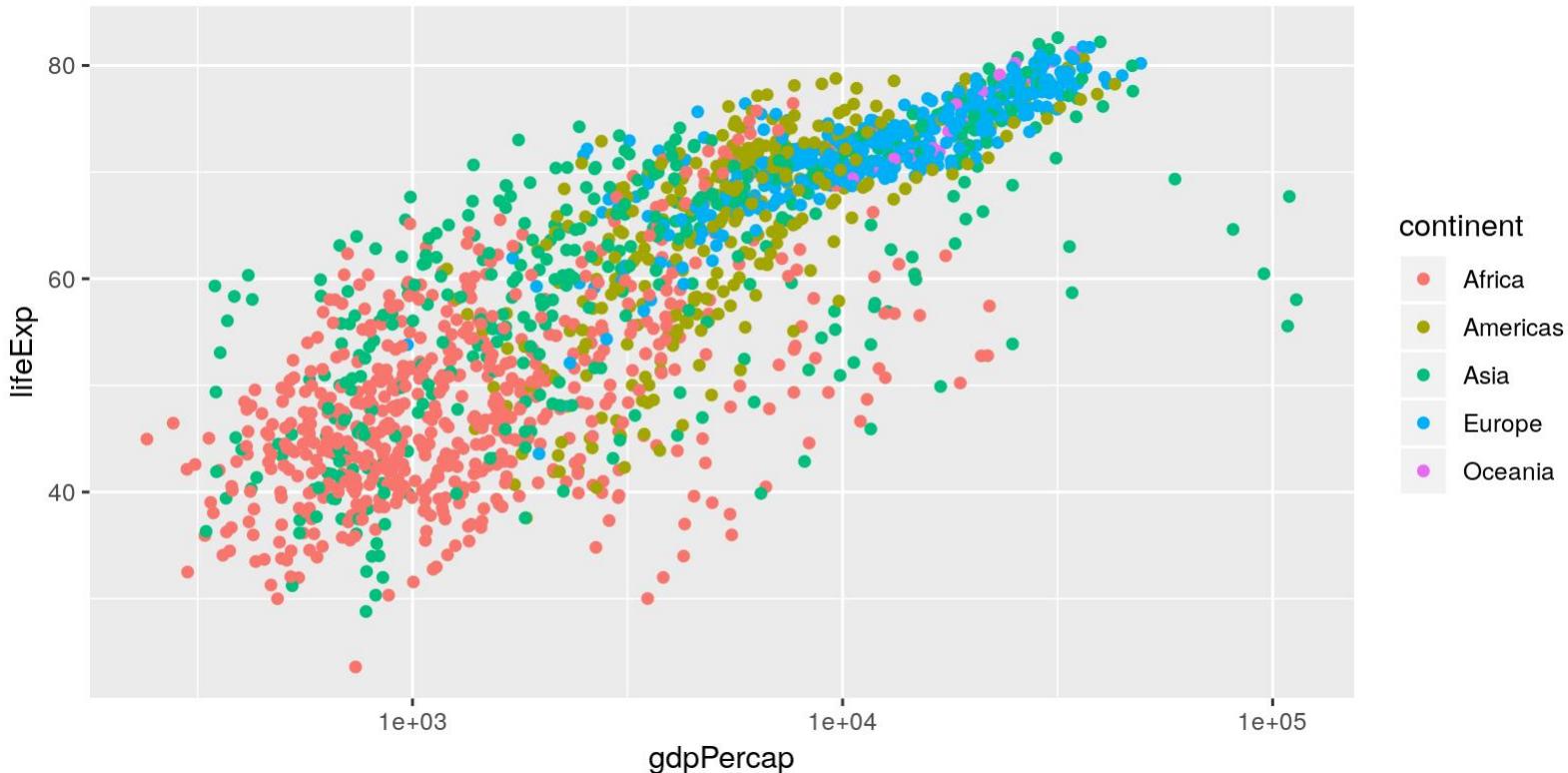
Tick labels modified via `labels` – convenient functions in *scales* package

```
p + geom_point() + geom_smooth() + scale_x_log10(labels = scales::dollar)
```



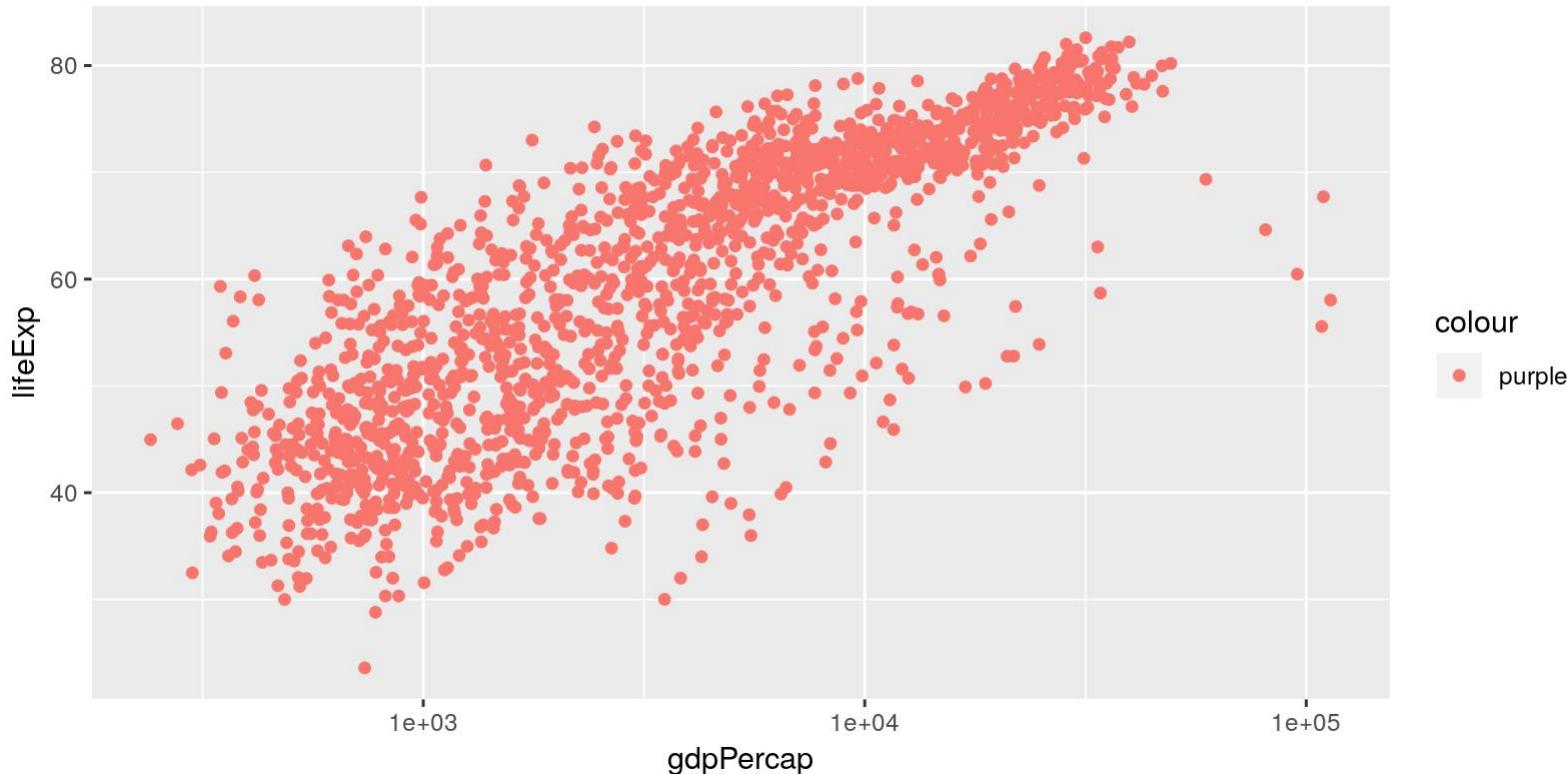
Mapping vs setting aesthetics

```
p <- ggplot(data = gapminder,  
             mapping = aes(x = gdpPercap, y = lifeExp, colour = continent))  
p + geom_point() + scale_x_log10()
```



Setting aesthetics – the wrong way

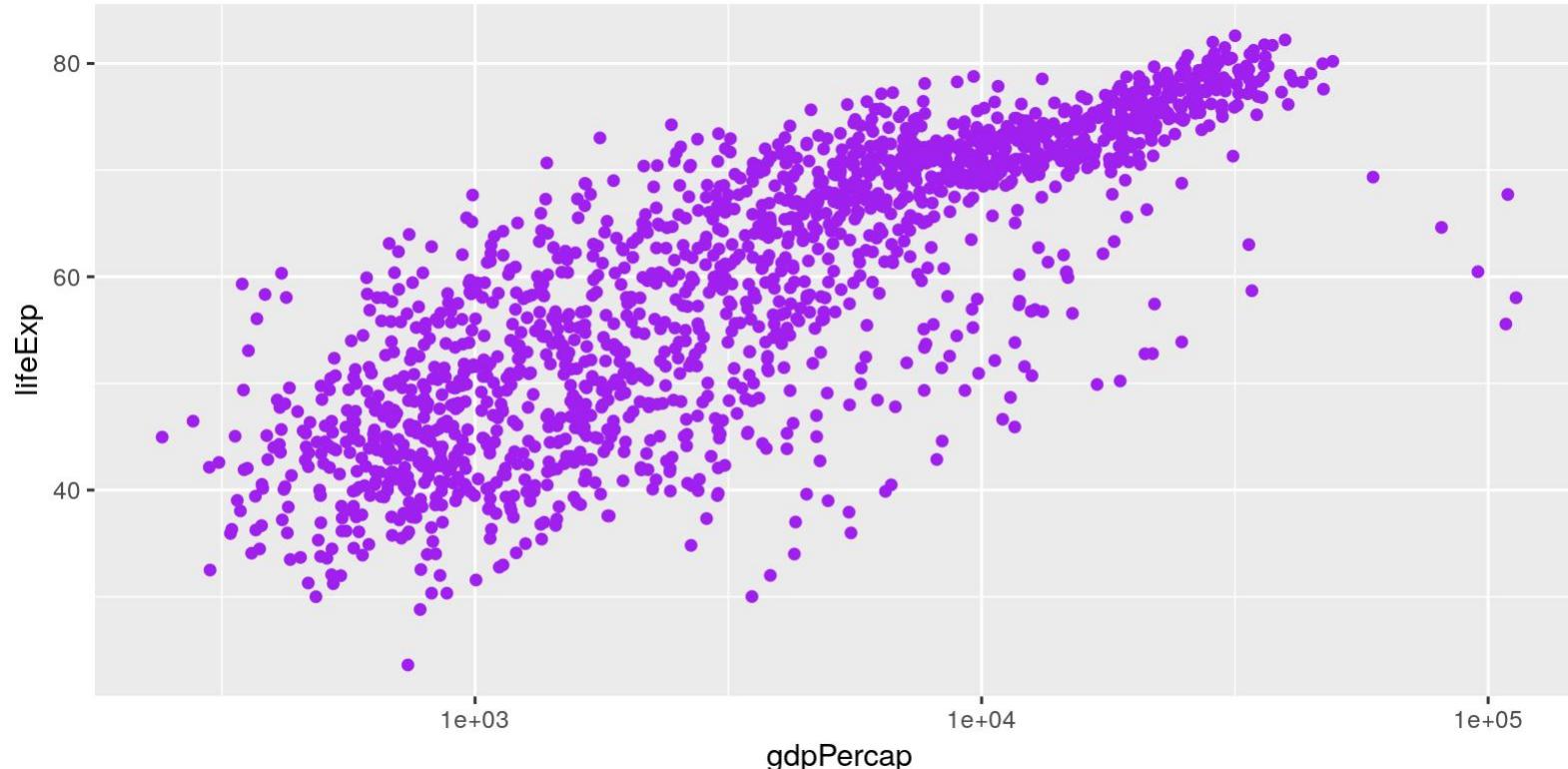
```
p <- ggplot(data = gapminder,  
             mapping = aes(x = gdpPercap, y = lifeExp, colour = "purple"))  
p + geom_point() + scale_x_log10()
```



Setting aesthetics – the right way

Mappings are in `aes()`, settings go *outside* `aes()`

```
p <- ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp))  
p + geom_point(colour = "purple") + scale_x_log10()
```

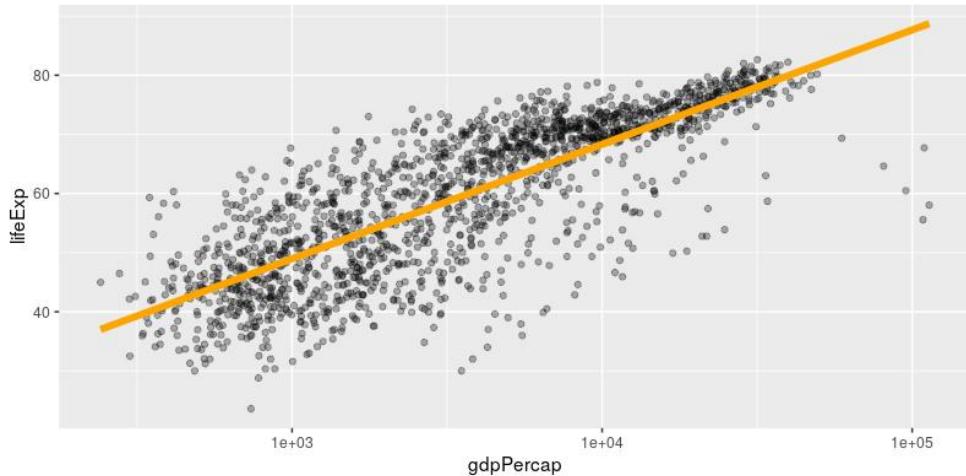


Setting aesthetics – the right way

Mappings are in `aes()`, settings go *outside* `aes()`

`alpha` controls transparency, `size` controls how big things are

```
ggplot(gapminder,  
       aes(x = gdpPercap,  
            y = lifeExp)) +  
  geom_point(alpha = 0.3) +  
  geom_smooth(method = 'lm',  
              colour = 'orange',  
              se = FALSE, size = 2)  
+  
  scale_x_log10()
```



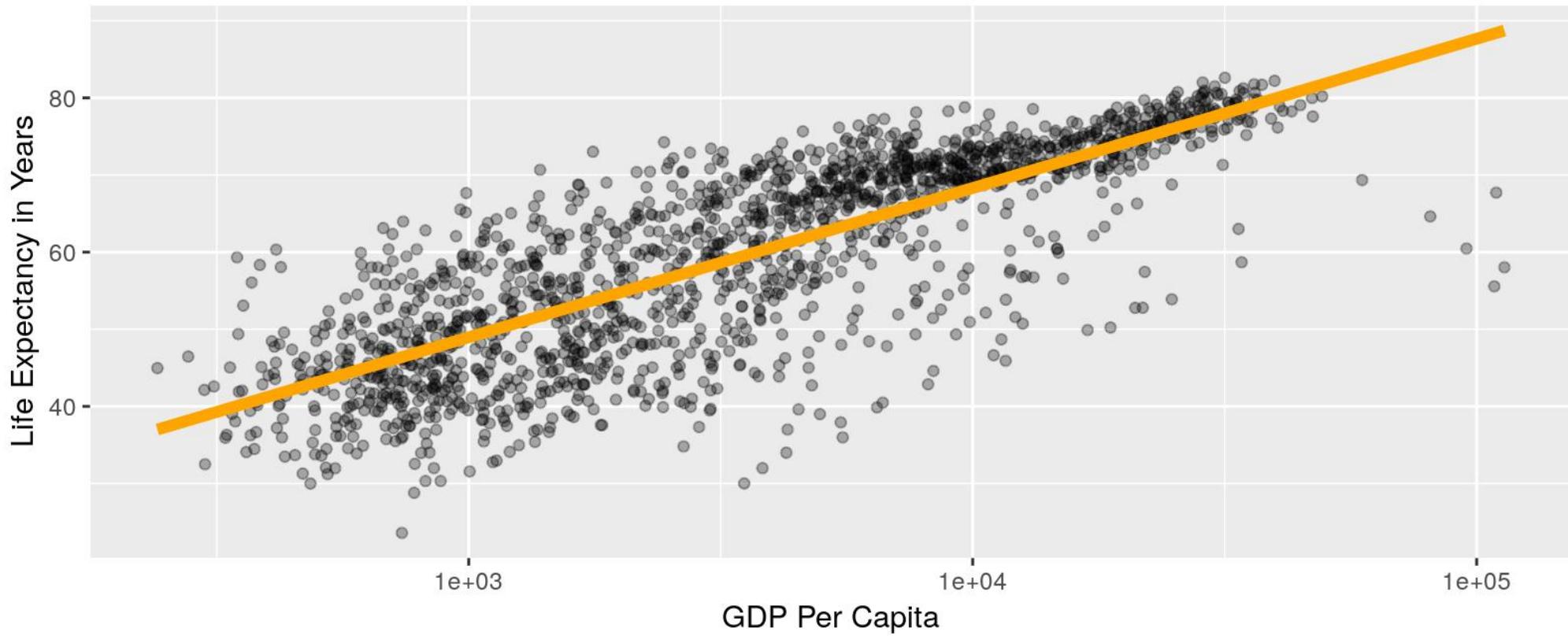
labs() – setting plot labels

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp)) +  
  geom_point(alpha = 0.3) +  
  geom_smooth(method = 'lm', colour = 'orange', se = FALSE, size = 2) +  
  scale_x_log10() +  
  labs(x = 'GDP Per Capita',  
       y = 'Life Expectancy in Years',  
       title = 'Economic growth & life expectancy',  
       subtitle = 'Data points are country-years',  
       caption = 'Source: Gapminder')
```

labs() – setting plot labels

Economic growth 7 life expectancy

Data points are country-years



Source: Gapminder

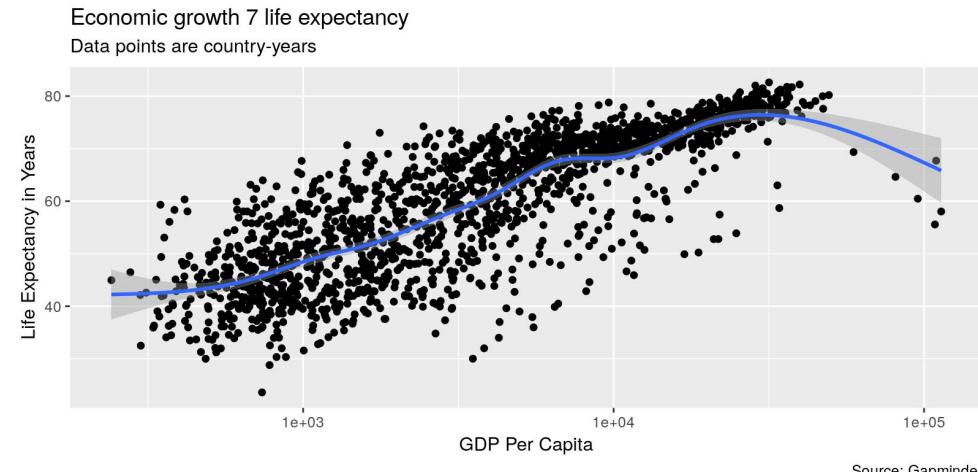
Reusing elements

You can save time and effort by reusing plot elements

```
my_labs <- labs(x = 'GDP Per Capita',  
                 y = 'Life Expectancy in Years',  
                 title = 'Economic growth 7 life expectancy',  
                 subtitle = 'Data points are country-years',  
                 caption = 'Source: Gapminder')
```

Then reuse

```
p + geom_point() + geom_smooth() +  
  scale_x_log10() + my_labs
```

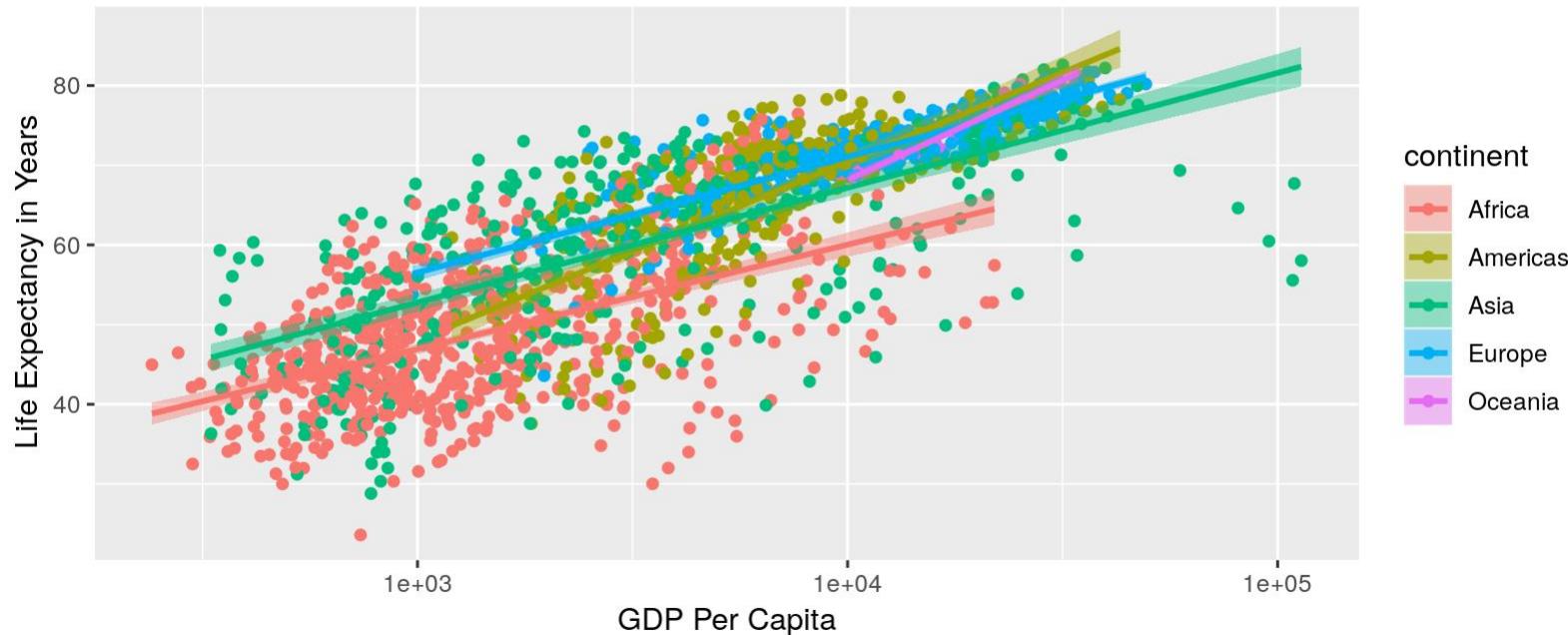


Matching aesthetics

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, colour = continent,  
fill = continent)) +  
  geom_point() + geom_smooth(method = 'gam') +  
  scale_x_log10() + my_labs
```

Economic growth 7 life expectancy

Data points are country-years



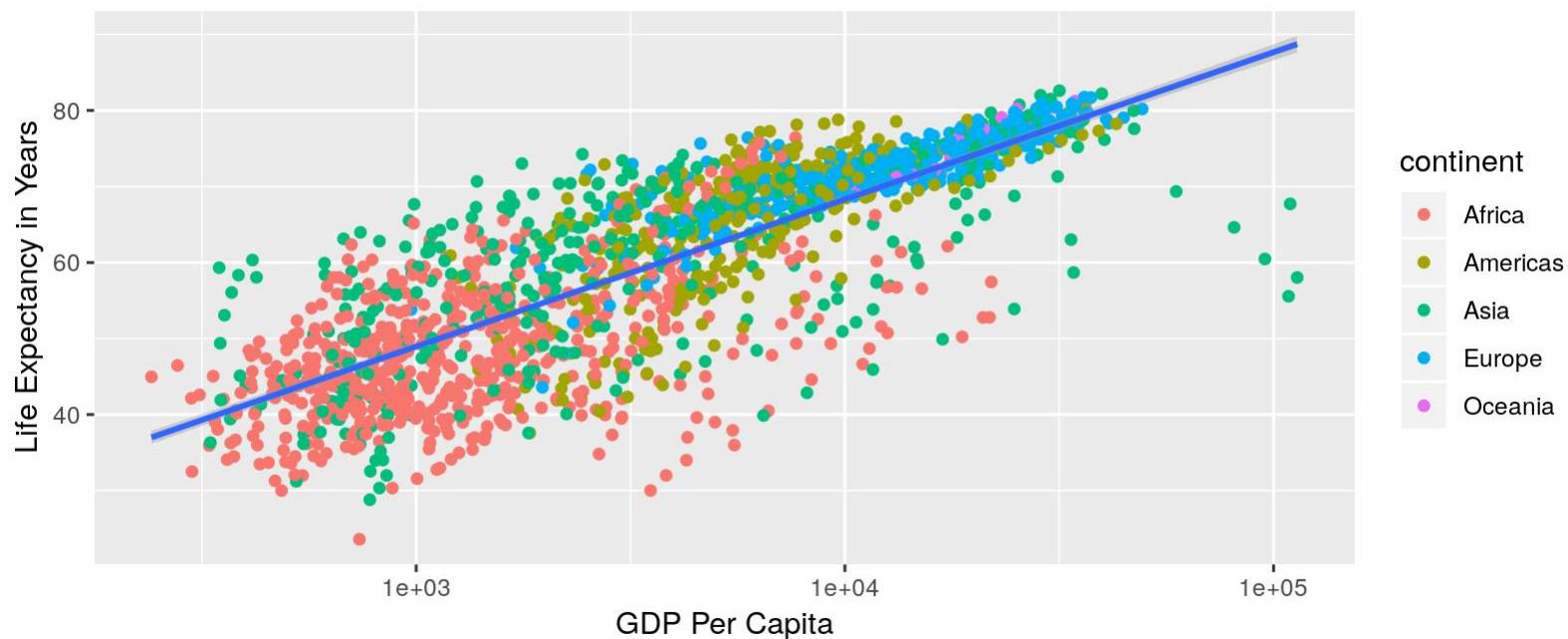
Source: Gapminder

Mapping aesthetics per geom

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp)) +  
  geom_point(mapping = aes(colour = continent)) + geom_smooth(method = 'gam') +  
  scale_x_log10() + my_labs
```

Economic growth & life expectancy

Data points are country-years



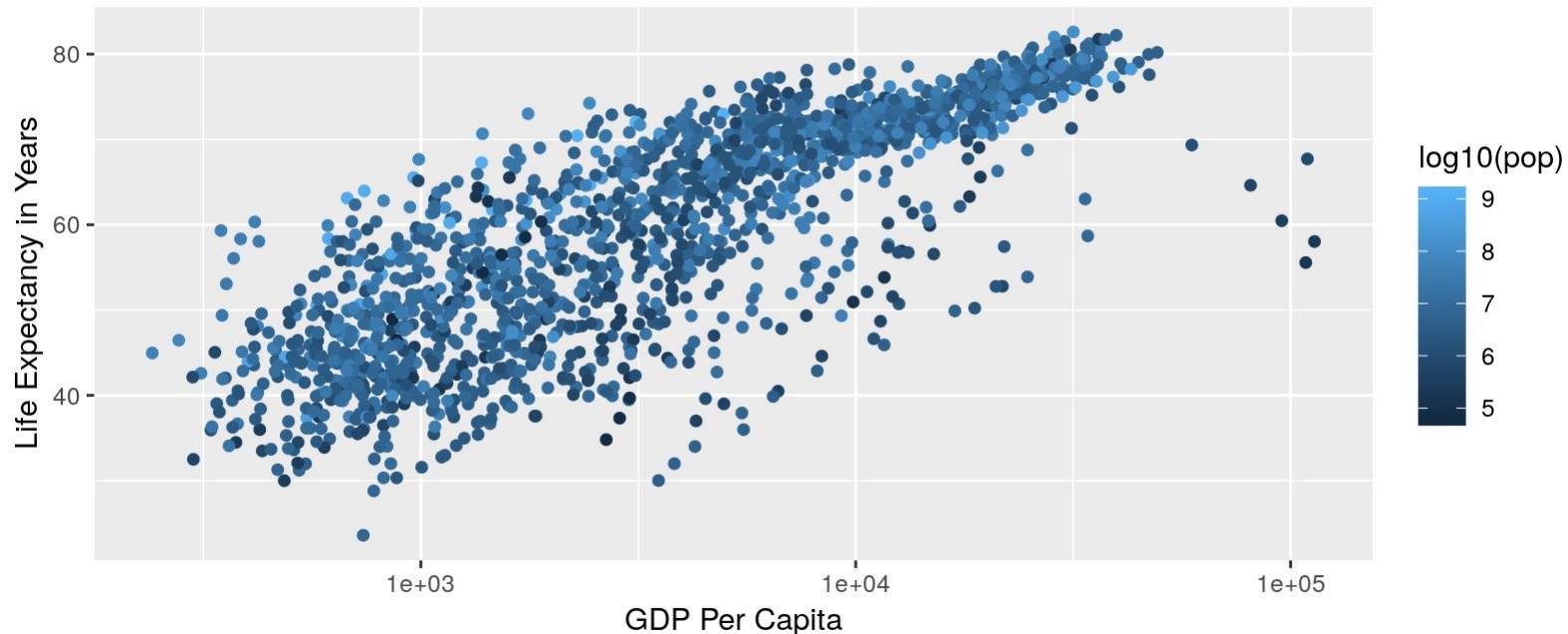
Source: Gapminder

Mapping continuous variables

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp)) +  
  geom_point(mapping = aes(colour = log10(pop))) +  
  scale_x_log10() + my_labs
```

Economic growth & life expectancy

Data points are country-years



Source: Gapminder

ggsave() – Saving your work

Plots can be rendered to disk in a range of formats – PNG, PDF, ...

Type of file depends on the extension given in `filename`

`ggsave()` saves the last `ggplot` object plotted

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp)) +  
  geom_point(mapping = aes(colour = log10(pop))) +  
  scale_x_log10() + my_labs
```

```
## save the last plot  
ggsave('my-plot.png')
```

```
## Saving 7 x 7 in image
```

ggsave() – Saving your work

Plots can be rendered to disk in a range of formats – PNG, PDF, ...

Type of file depends on the extension given in `filename`

`ggsave()` saves a specific `ggplot` object if given on

```
my_plt <- ggplot(gapminder, aes(x = gdpPercap, y = lifeExp)) +  
  geom_point(mapping = aes(colour = log10(pop))) +  
  scale_x_log10() + my_labs  
  
## save a specific plot object  
ggsave('my-plot.pdf', plot = my_plt)
```

```
## Saving 7 x 7 in image
```

ggsave() – Specifying size

ggsave() always saves objects in inches & takes the size from the device if not specified

Can set width and height to numeric values and select the units via units

```
my_plt <- ggplot(gapminder, aes(x = gdpPercap, y = lifeExp)) +  
  geom_point(mapping = aes(colour = log10(pop))) +  
  scale_x_log10() + my_labs  
  
## save a specific plot object  
ggsave('my-plot-cm.pdf', plot = my_plt, height = 10, width= 20, units = 'cm')
```