R you ready: Making the switch to R

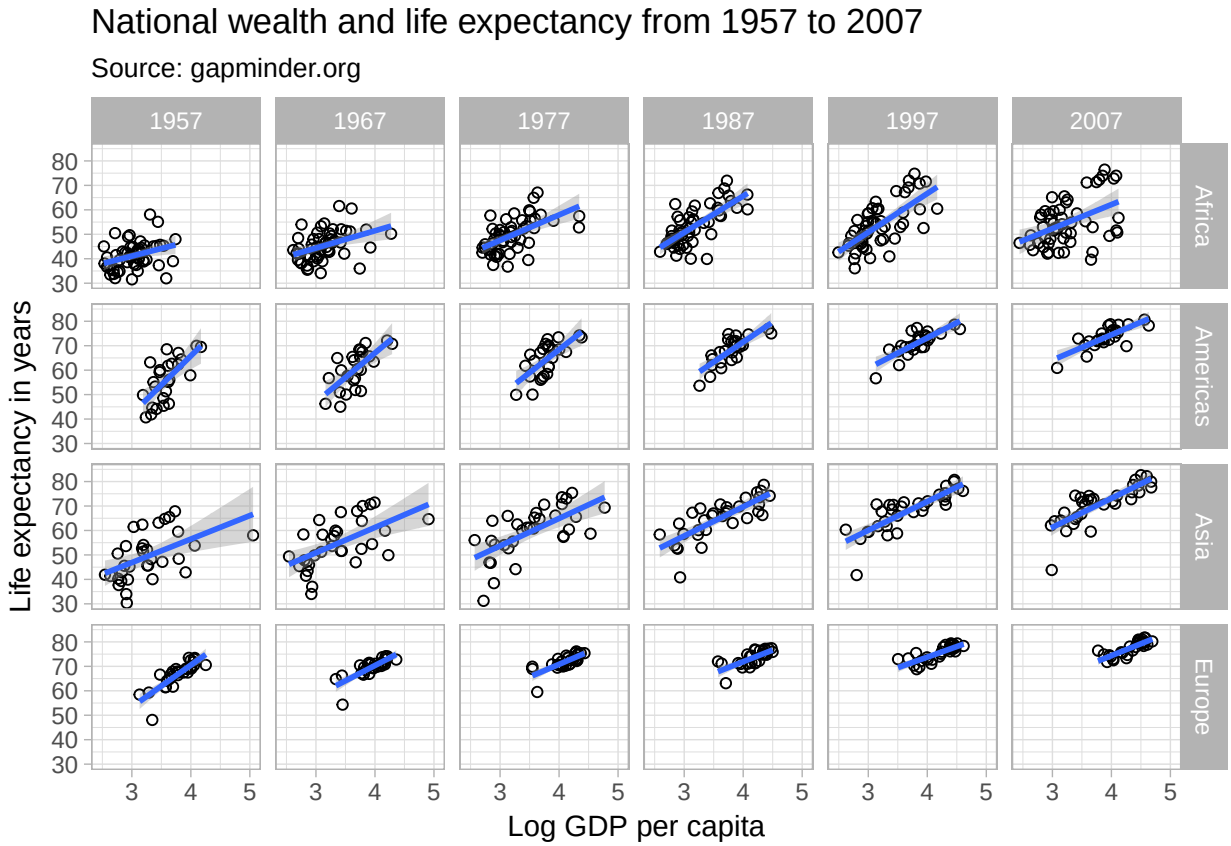
Data visualization with ggplot2

James Uanhoro

July 21, 2017

Beginning with the end

We will try to build this plot:



First we project, then we ggplot2

Keeps all of our work within a project folder

In RStudio: `File -> New Project -> New Directory`

Use `Existing Directory` if you have some work in a folder already.

Copy the `gapminder.csv` dataset into the project folder

Let's start work in a new script: `Ctrl + Shift + N`

Let's load in the data frame

```
# When you want to type the filename, press tab after typing the first quotation mark
gap <- read.csv("gapminder.csv")
str(gap) # I would also use head(gap)
```

```
## 'data.frame':    1704 obs. of  8 variables:
##  $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ year     : int  1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
##  $ lifeExp  : num  28.8 30.3 32 34 36.1 ...
##  $ pop      : int  8425333 9240934 10267083 11537966 13079460 14880372 12881816 13867957 16317921 22
##  $ gdpPercap: num  779 821 853 836 740 ...
##  $ logPop   : num  6.93 6.97 7.01 7.06 7.12 ...
##  $ logGdp   : num  2.89 2.91 2.93 2.92 2.87 ...
```

Let's ggplot2

```
# Loading the package
library(ggplot2)
```

Major elements to ggplot2:

- Data: `ggplot()`
- Map variable(s) to aesthetics (view): `aes()`
- Define type of plot: `geom()`

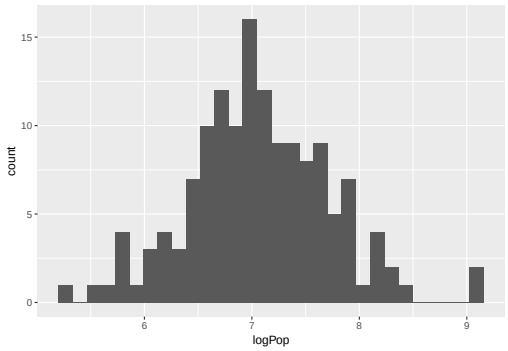
Before we ride

We'll work with data from 2007 first, use the filter function you learned earlier to select only 2007 data. Save it to a new variable named `gap.2007`.

```
library(dplyr)
gap.2007 <- filter(gap, year == 2007)
```

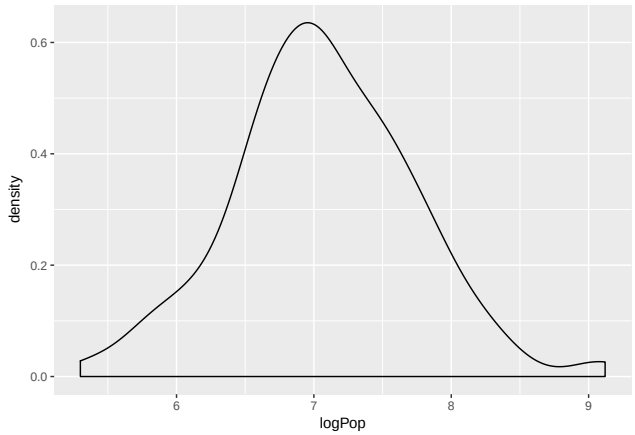
Let's visualize some continuous variables

```
p <- ggplot(data = gap.2007, aes(x = logPop))
p + geom_histogram()
```



Same variable, different visualization

```
p + geom_density()
```



See the structure?

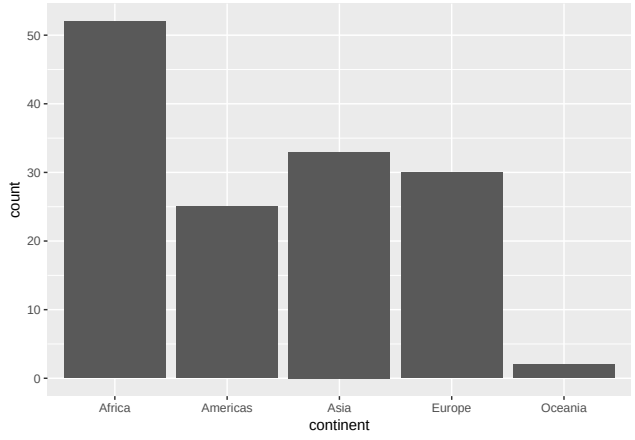
Once we give `ggplot()` a data frame, then map a variable to a graph element using `aes()` and save it as a variable, we can add different `geom()`'s to it.

Practice with the life expectancy (`lifeExp`) variable (or any other continuous variable) then produce a histogram and a density plot of it.

Anyone willing to create a barchart of a discrete variable? - `continent`? The `geom()` you want is `geom_bar()`

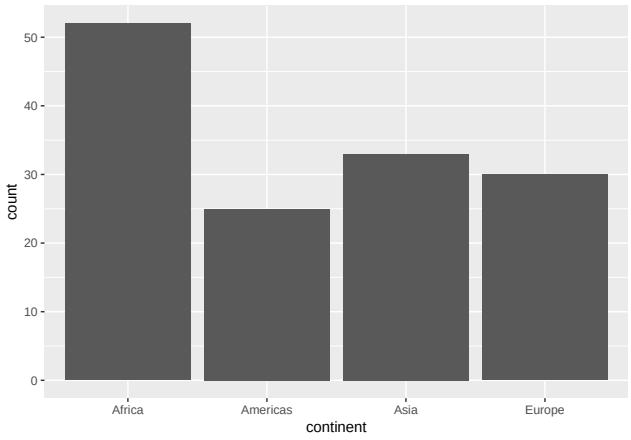
Bar chart of continent - little modification

```
p <- ggplot(data = gap.2007, aes(x = continent))
p + geom_bar()
```



Bar chart of continent - No Oceania!

Oceania has so few countries, filter it out and continue saving to `gap.2007`.



Two variables - one categorical and one continuous

Let's try to see the relation between `continent` and `lifeExp` in 2007

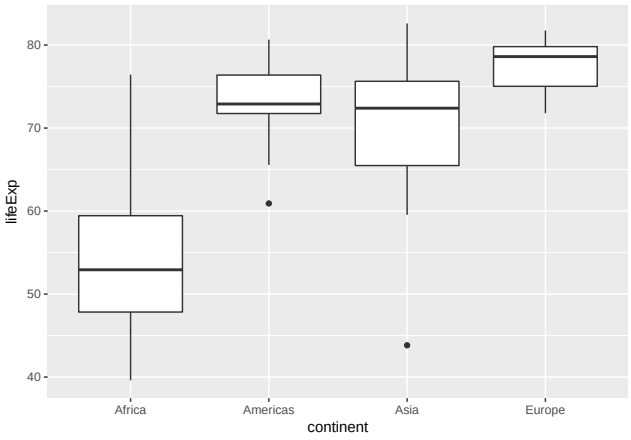
```
p <- ggplot(data = gap.2007, aes(x = continent, y = lifeExp))
```

Does this basic structure make sense?

Given this basic structure, we can make boxplots

Boxplot

```
# Making a boxplot
p + geom_boxplot()
```



Try the boxplot with `continent` and `logGdp`

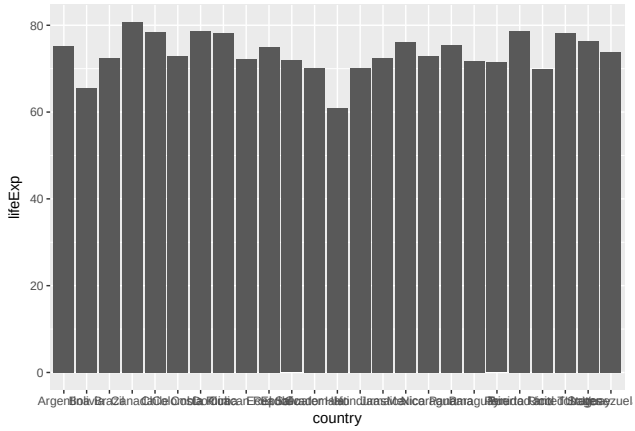
Let's zoom in on the Americas

Filter down to America:

```
gap.2007.amr <- filter(gap.2007, continent == "Americas")
```

Let's try geom_col()

```
p <- ggplot(data = gap.2007.amr, aes(x = country, y = lifeExp))
p + geom_col()
```



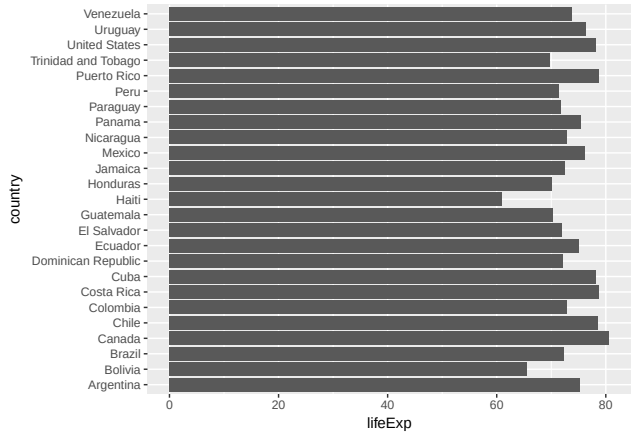
Our first snag :/

We can rotate those country names, or **we can flip the chart instead**!

And to make the plot more useful, we can order the countries from highest to lowest.

Flipping - coord_flip()

```
p + geom_col() + coord_flip()
```



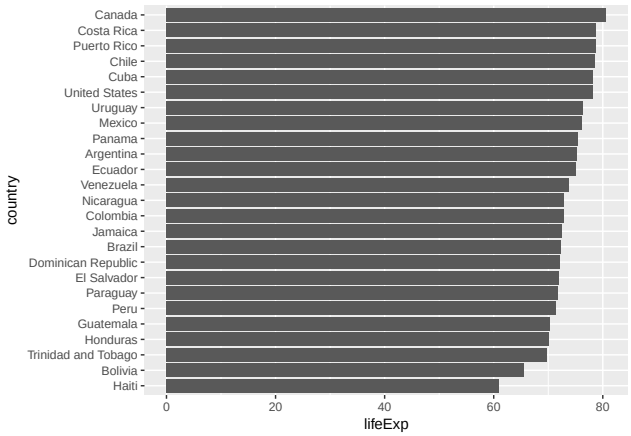
Reordering

In *R*, you can `reorder` a variable using: `reorder(variable.name, variable.to.order.by)`

If we want to reorder `country` by `lifeExp`, we can place the following code inside `geom_col()`: `aes(x = reorder(country, lifeExp)`

Essentially, we can re-define `x` and `y` using `aes()` inside any `geom()` like:

```
p + geom_col(aes(x = reorder(country, lifeExp)), data = gap.2007.amr) + coord_flip()
```



Since we did not change `y` inside this `aes()`, it stays the same from its original definition in `p` above

If you ask me, this is pretty neat.

Let's change those axes labels and save it! First:

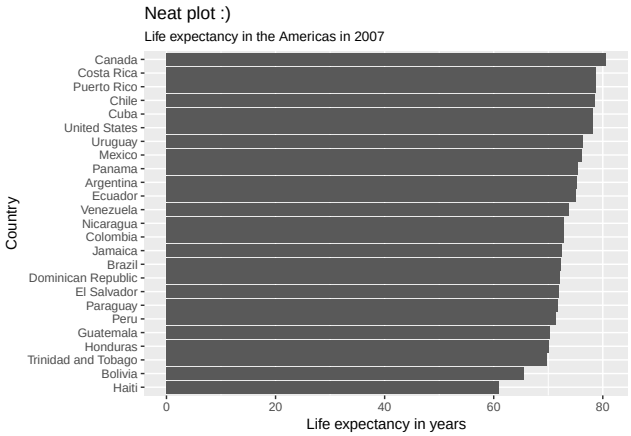
```
p <- p + geom_col(aes(x = reorder(country, lifeExp)), data = gap.2007.amr) + coord_flip()
```

By doing this, I am saving all my code into `p`. Now, `p` contains the columns, the flipping, the whole shebang!

Labels

We can add title, subtitle, rename the x-axis, y-axis and more using `labs()` - pretty handy.

```
(p <- p + labs(
  title = "Neat plot :)",
  subtitle = "Life expectancy in the Americas in 2007",
  x = "Country",
  y = "Life expectancy in years"))
```



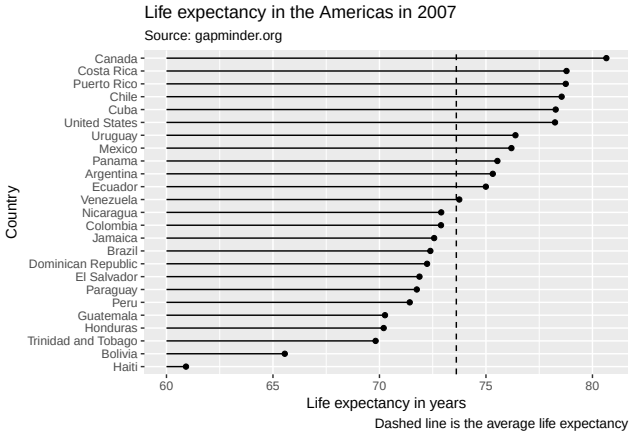
I guess the subtitle and the title should be reversed?

ggsave()

Executing the code below produces a pretty neat plot named `neat_plot.png` inside a folder named `plots` inside your project folder. Let's create this folder called `plots` from Windows/Finder, then run the code.

```
ggsave(filename = "plots/neat_plot.png", plot = p)
```

This would even be better



And you are are not too far off. It requires `geom_segment()`, check it out on your time!

Two continuous variables

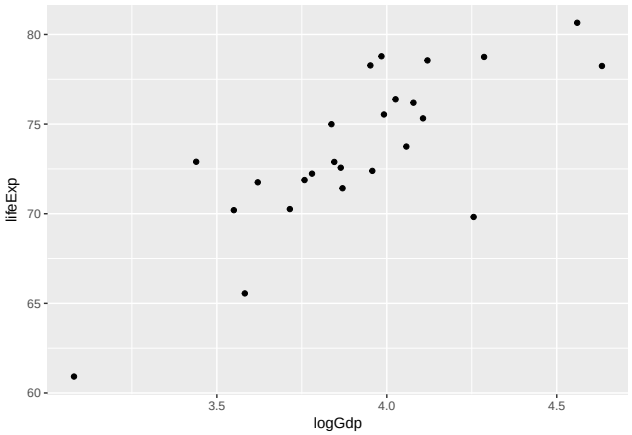
Here we can start with scatterplots. Let's go with `logGdp` and `lifeExp`

In 2007 within the Americas, was there any relationship between `logGdp` `lifeExp`?

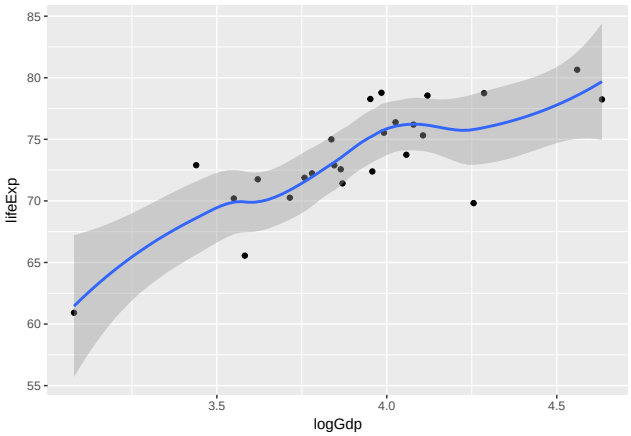
Can anyone write the first line of this with `ggplot()` and `aes()` line?

The type of `geom()` you want is `geom_point()`

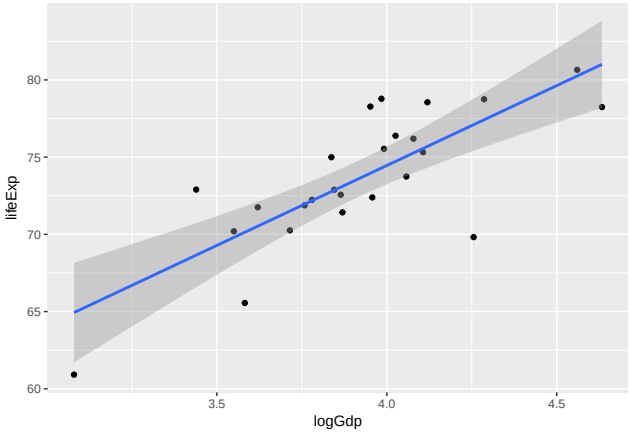
```
p <- ggplot(data = gap.2007.amr, aes(x = logGdp, y = lifeExp))
p + geom_point()
```




```
# Add geom_smooth() for a running line
p + geom_point() + geom_smooth()
```

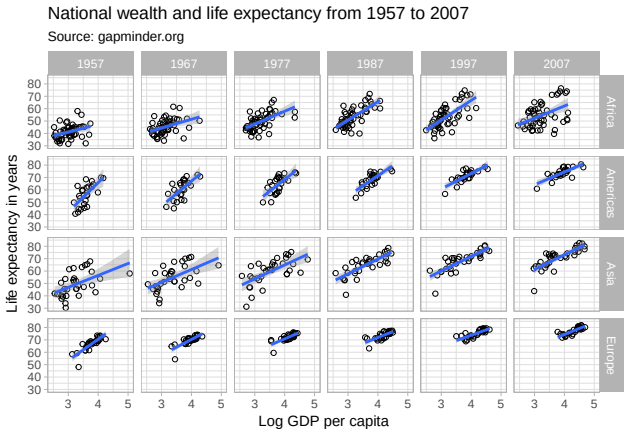



```
# For straight lines, write `method = "lm"` inside `geom_smooth()`
p + geom_point() + geom_smooth(method = "lm")
```



Are we any closer since beginning?

It seems to me like we have already created the Americas in 2007. And we're closer than you might think!



But first a colourful detour :)

Let's use all of 2007 data for now - `gap.2007`

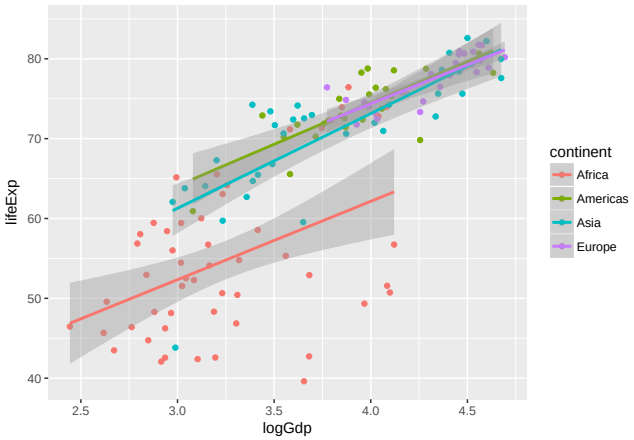
We can link a colour to a variable using aesthetics. Let's colour the scatterplot by `continent`. The code we'd use is `aes(color = continent)`. We can place this alone inside `geom_point()` or together with `x =` and `y =` inside the `aes()` inside `ggplot()`:

- `ggplot(data = gap.2007, aes(x = logGdp, y = lifeExp, color = continent))`
- `geom_point(aes(color = continent))`

Both options give us a nice legend

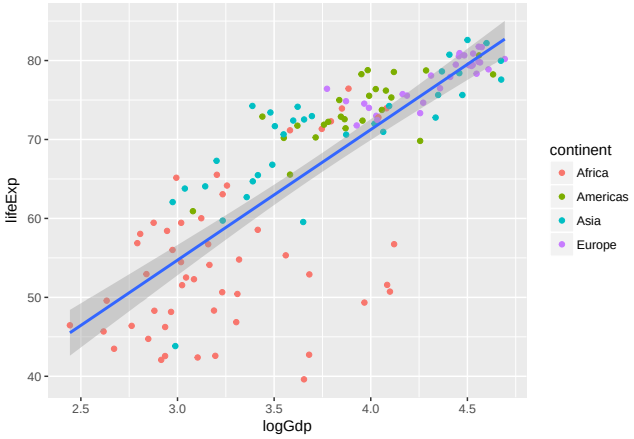
Let's try option one

Ugh, not out best work!



Let's try option two

Some redemption



Ending our not so colourful detour

Where colour goes matters

```r
# Syntax 1
(p <- ggplot(data = gap.2007, aes(x = logGdp, y = lifeExp, colour = continent)) +
   geom_point() +
   geom_smooth(method = "lm"))
# Syntax 2
(p <- ggplot(data = gap.2007, aes(x = logGdp, y = lifeExp)) +
   geom_point(aes(colour = continent)) +
   geom_smooth(method = "lm"))
```

When we have several categories, maybe colours are not our best bet. However, when they occupy somewhat different regions, colours may be of use - It all depends on your purpose. Let's see faceting.

Faceting

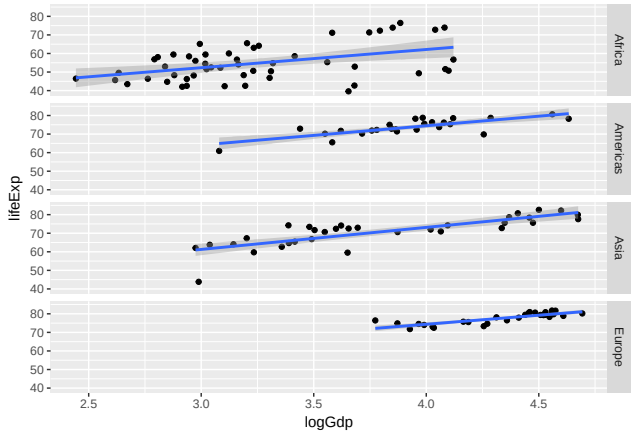
It's actually a very simple step to move to facets.

```r
# Define p without any reference to colours
p <- ggplot(data = gap.2007, aes(x = logGdp, y = lifeExp)) +
  geom_point() + geom_smooth(method = "lm")
```

Try adding `facet_grid(continent ~ .)`. Then switch it to `facet_grid(. ~ continent)`

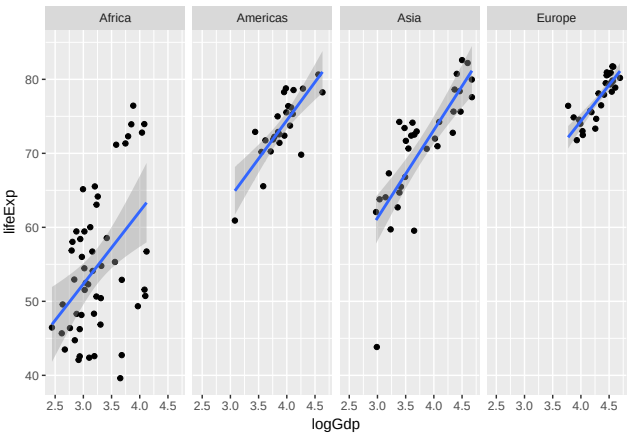
We seem close!

```r
p + facet_grid(continent ~ .)
```



We seem close, yasss?

```r
p + facet_grid(. ~ continent)
```

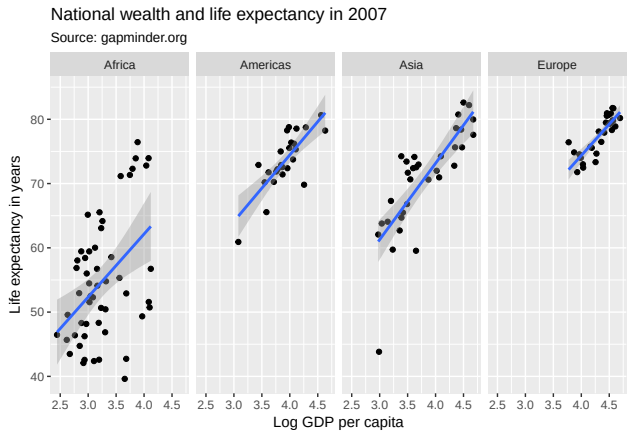


Labelling exercise

Use what you know about labels to label the chart. I have the following recommendations:

- title: National wealth and life expectancy in 2007
- subtitle: Source: gapminder.org
- x-axis: Log GDP per capita
- y-axis: Life expectancy in years

Did we get this?



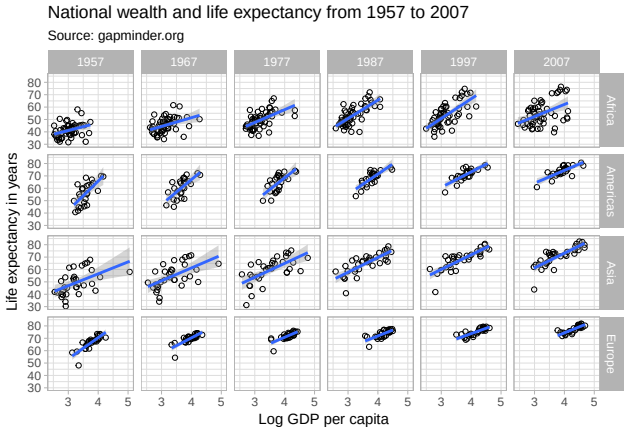
The full structure of the syntax

Looks scary but we all just built this together

```
p <- ggplot(data = gap, aes(x = logGdp, y = lifeExp)) +
  geom_point() +
  geom_smooth(method = "lm") +
  facet_grid(. ~ continent) +
  labs(x = "Log GDP per capita", y = "Life expectancy in years",
       title = "National wealth and life expectancy in 2007",
       subtitle = "Source: gapminder.org")
```

See the elements, they make some sense, right? right? ...

Let's run towards the goal



1957, 1967, 1977, 1987, 1997, 2007

We want to go back to the `gap` data frame. We will create a subset of it using data from only the years above.

Save the new data frame as `gap.7s`

Don't forget to delete Oceania!

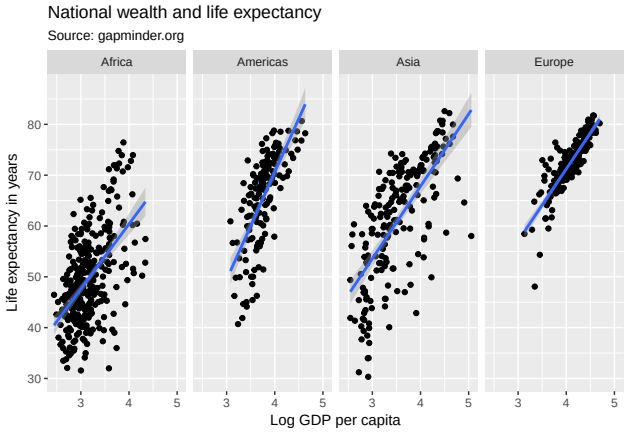
You are going to do the rest, I'm entering `no-code-mode` :).

What you might need:

- `filter()`
- `%in%`
- `c(1957, 1967, 1977, 1987, 1997, 2007)`

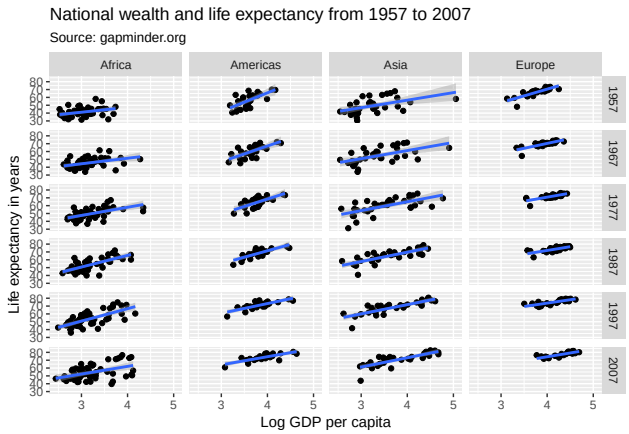
Re-create the old full structure

This time, use `gap.7s` as your data frame. Got this?



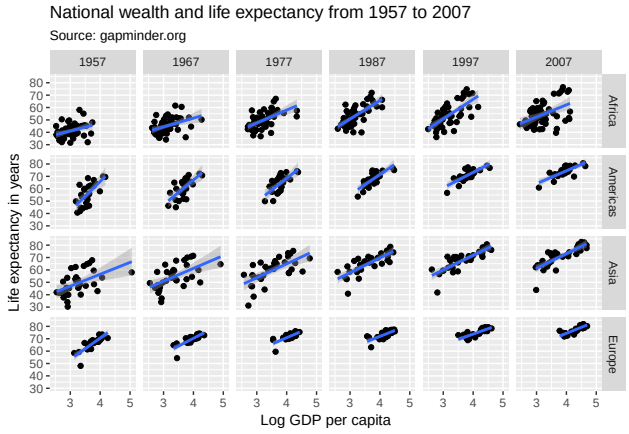
Final step?

Replace the `.` in `facet_grid()` with `year`



One more

Swap `year` with `continent` in `facet_grid()`



Then another

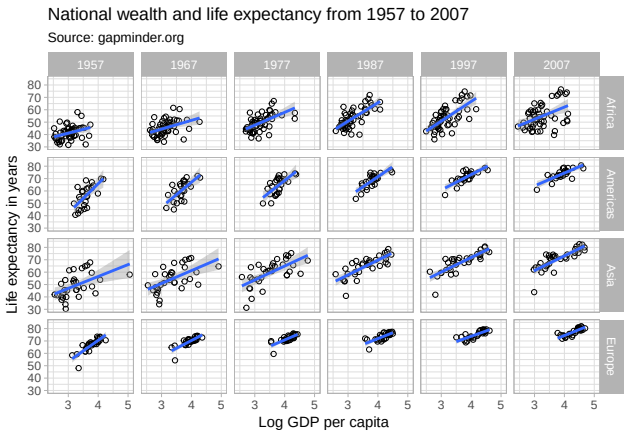
Our end goal uses empty circles. Add `shape = 1` inside `geom_point()`.

```
(p <- ggplot(data = gap.7s, aes(x = logGdp, y = lifeExp)) +
   geom_point(shape = 1) +
   geom_smooth(method = "lm") +
   facet_grid(continent ~ year) +
   labs(x = "Log GDP per capita", y = "Life expectancy in years",
        title = "National wealth and life expectancy from 1957 to 2007",
        subtitle = "Source: gapminder.org"))
```

`shape = 1` gives us empty circles which allow us to perceive overlap better than fully darkened circles. Try out other shape numbers.

This is our goal

A slight difference, right? We need `theme()`



theme()

The `theme_()` group of functions allows us modify plots in nice ways:

```
(p <- p + theme_classic())
```

The goal uses `theme_light()`. Try out the different types.

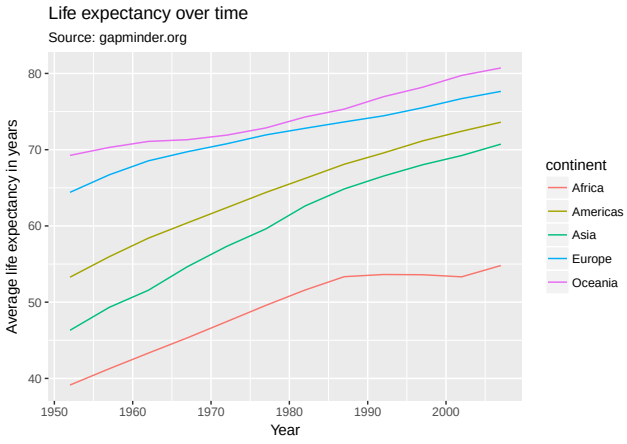
ggsave()

This is a precious memory, save it!

```
ggsave("plots/end_goal.png", p)
```

Back to earth :)

Try this out with the `gap` data frame, and `geom_line()`



You've probably failed, and that's fine

What do we know about this plot?

```
# We know this, right?
aes(x = year, y = lifeExp, colour = continent)
# And this too
labs(x = "Year", y = "Average life expectancy in years",
     title = "Life expectancy over time",
     subtitle = "Source: gapminder.org")
```

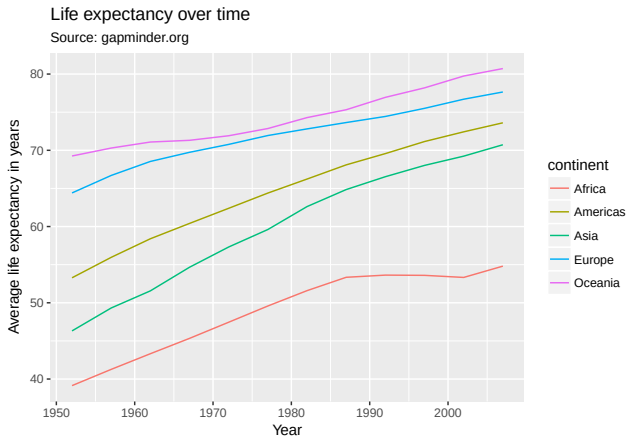
geom_line() uses summarised data

We need summary data of `lifeExp` by `continent` by `year`. Anyone recall how to do this?

- use `group_by()` and `summarize()`
- save the `mean(lifeExp)` to `lifeExp`
- save the data frame into `gap.summary`

Now create a ggplot using `gap.summary`, with the right `aes()`, the `geom_line()`, and the `labs()`

Worked out?

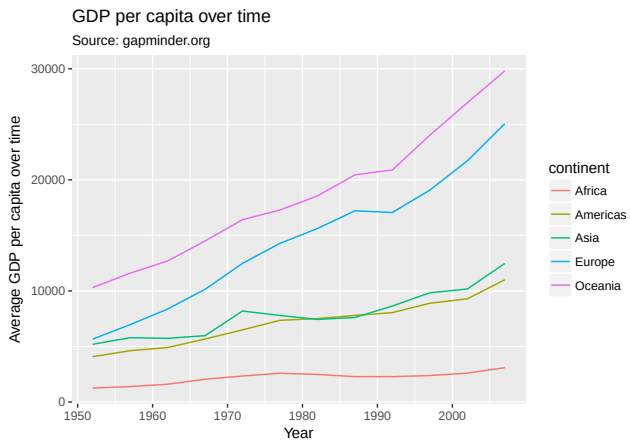


Did you do this?

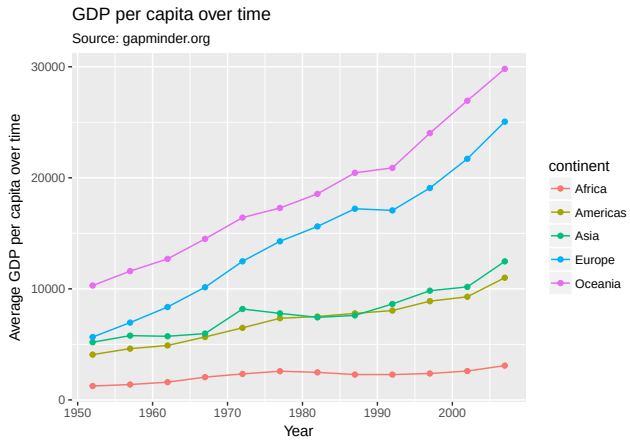
```
gap.summary <- summarize(group_by(gap, continent, year), lifeExp = mean(lifeExp))
(p <- ggplot(gap.summary, aes(x = year, y = lifeExp, colour = continent)) +
   geom_line() +
   labs(x = "Year", y = "Average life expectancy in years",
        title = "Life expectancy over time",
        subtitle = "Source: gapminder.org"))
```

New goal

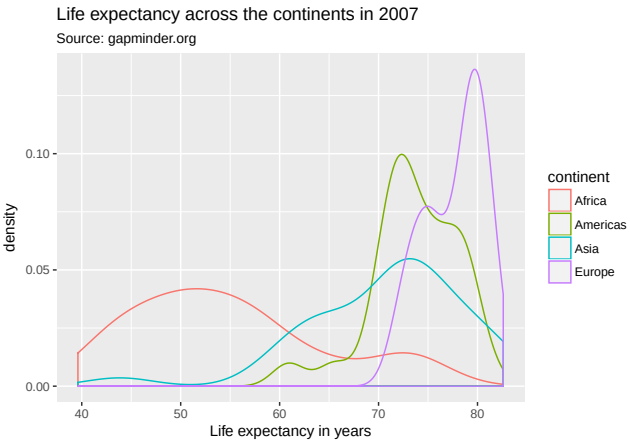
Use `gdpPercap` variable:



Next goal



Simple goal



Resources

- Mailing list: http://groups.google.com/group/ggplot2
- Wiki: https://github.com/hadley/ggplot2/wiki
- Website: http://had.co.nz/ggplot2/
- ggplot2 cheatsheet: https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf