Using ggplot for graphics

Grammar of Graphics

Most of us are familar with producing plots with spreadsheet software such as Microsoft Office or Libre Office. We usually try to plot one or two dimensional data (e.g. plotting a series of share prices, or the relationship between income and education), and then select one of the graphs provided by the software package. While this approach is very straight forward, it (i) diminishes the creativity of the users, (ii) struggles with plotting data that has more than 2 dimensions and (iii) can be very time consuming to replicate or modify the plot. Luckily, Leland Wilkinson (1999) introduced the idea of the Grammar of Graphis to the world. It took several years until Hadley Wickham's operationalised a variant of the GoG in his ggplot2 R package.

What is the grammar of graphics? Let us have Wilkinson explain it to us:

"The grammar of graphics (GoG) denotes a system with seven classes embedded in a data flow. This data flow specifies a strict order in which data are transformed from a raw dataset to a statistical graphic. Each class contains multiple methods, each of which is a function executed at the step in the data flow corresponding to that class. The classes are orthogonal, in the sense that the product set of all classes (every possible sequence of class methods) defines a space of graphics which is meaningful at every point. The meaning of a statistical graphic is thus determined by the mapping produced by the function chain linking data and graphic." (Wilkinson, 2010)

What does it mean? In a nutshell:

- GoG is a structured approach of turning data into visualizations by following a specific set of steps.
- Each step plays a unique role, like preparing the data, choosing how to display it, and adjusting its appearance.
- Using these steps, we can create a wide variety of meaningful charts and graphs.

Wickham (2008) operationalised this idea by writing the ggplot2 package that is based on this layered approach, and we are going to use the package today! Using it is similar to building a house:

- **Foundation**: We select a dataset that we want to use for plotting. We decide what variable we put on the axes, and set some global *aesthetics* options that will apply for all the layers we use.
- **Building the house**: To actually produce the chart, we have to add geometrics opjects (geoms). E.g. this could be bars, lines, points, etc. We could add another data layer to the graph with its own aesthetics, which is similar to building a conservatory to your house.
- **Decorating**: To make sure that the graph looks perfect, we can modify the scales, lables and legend of the graph. We can also apply themes and facetting.

Loading necessary packages and data

The R base package is like a smart phone without any extra apps installed. For example, you can make calls, send messages and even have an exciting ringing tone. While this will be enough for using your smart phone as a "phone", to get most out of it, you will install additional apps, like a music player, VPN, and so on. Similarly, to boost the power of R as a statistical software package, we will install additional packages. One of the most popular packages for example is the `ggplot2` package that allows you to produce professional graphics. We will discuss them in more detail in today's session.

Let's install the following package: `tidyvers`. This package also include the `ggplot2` package and others that will simplify data manipulation.

```r
install.packages("tidyverse")
```

Once it is install, don't forget to activate it!

```r
library("tidyverse")
```

The `ggplot2` package also contains some example datasets. We will use the mpg dataset to introduce ggplot and follow the example by Wickham et al. (2025) https://ggplot2-book.org

To get an idea what the data looks like, just type:

```r
head(mpg)
```

```
# A tibble: 6 x 11
  manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
  <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa~
```

```
2 audi         a4      1.8  1999     4 manual(m5) f         21    29 p     compa~
3 audi         a4      2    2008     4 manual(m6) f         20    31 p     compa~
4 audi         a4      2    2008     4 auto(av)   f         21    30 p     compa~
5 audi         a4      2.8  1999     6 auto(l5)   f         16    26 p     compa~
6 audi         a4      2.8  1999     6 manual(m5) f         18    26 p     compa~
```

The variables are:

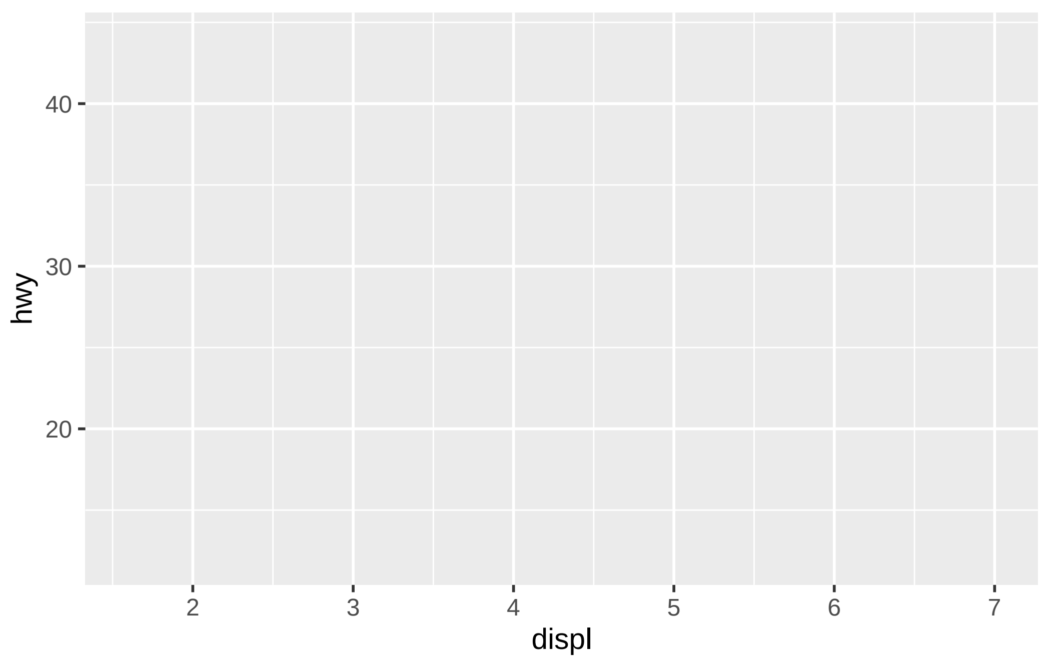
- `cty` and `hwy` record miles per gallon (mpg) for city and highway driving.
- `displ` is the engine displacement in litres.
- `drv` is the drivetrain: front wheel (f), rear wheel (r) or four wheel (4).
- `model` is the model of car. There are 38 models, selected because they had a new edition every year between 1999 and 2008.
- `class` is a categorical variable describing the "type" of car: two seater, SUV, compact, etc.

Let's get started with ggplot

Using the grammar of graphics concept, we will analyse the relationship between `displ` and `hwy`. Remember the three steps of producing graphs: 1. select the **data** 2. set **aesthetic** properties 3. add **layers** to display the data

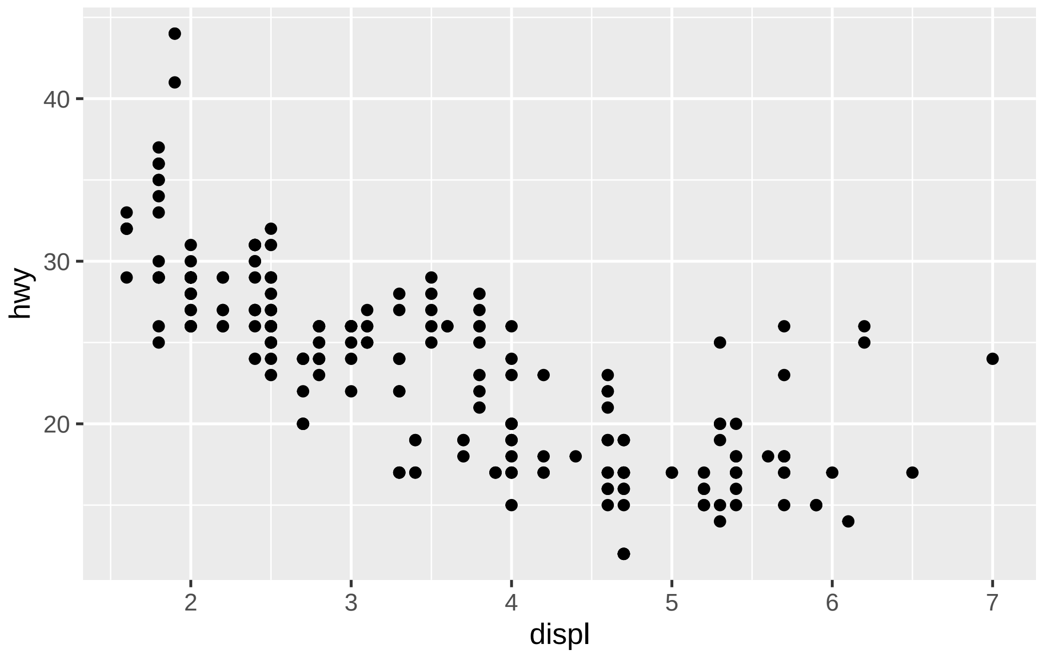
We begin with mapping data to aesthetic properties:

```
ggplot(mpg, aes(x = displ, y = hwy))
```



The graph is empty! We can see that we have the correct labels on the axes, but we did not add the necessary layer to display the data. We can use geometric object, or *geoms* to do make the command do what we would expect it to do: to show data!

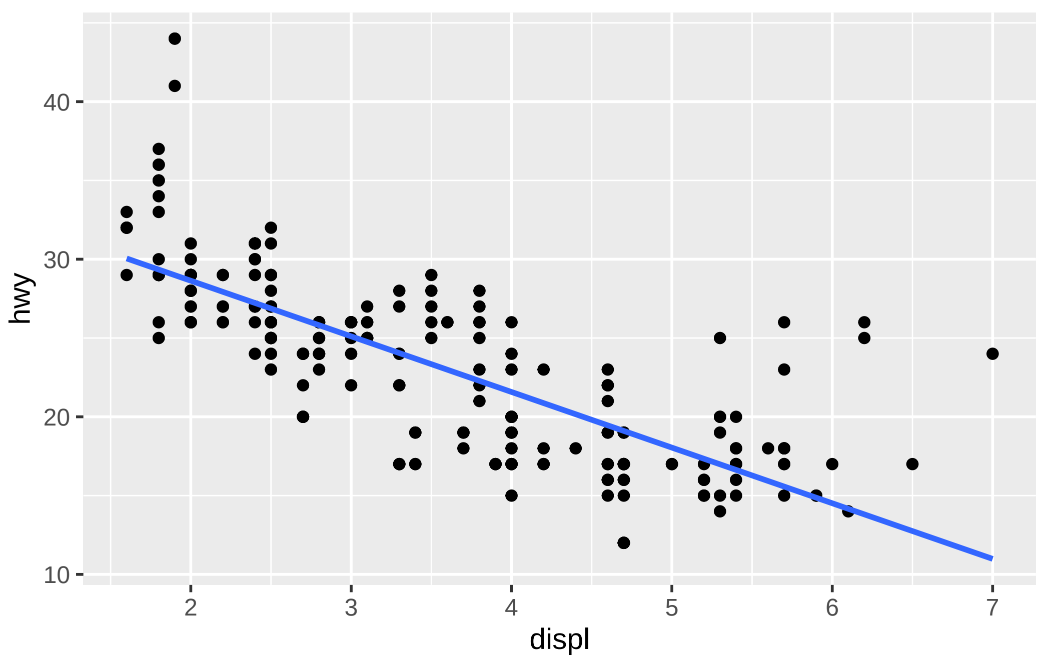
```
ggplot(mpg, aes(x = displ, y = hwy)) +
 geom_point()
```



You can add another layer like a regression line to the plot to picture the statistical relationship between the variables:

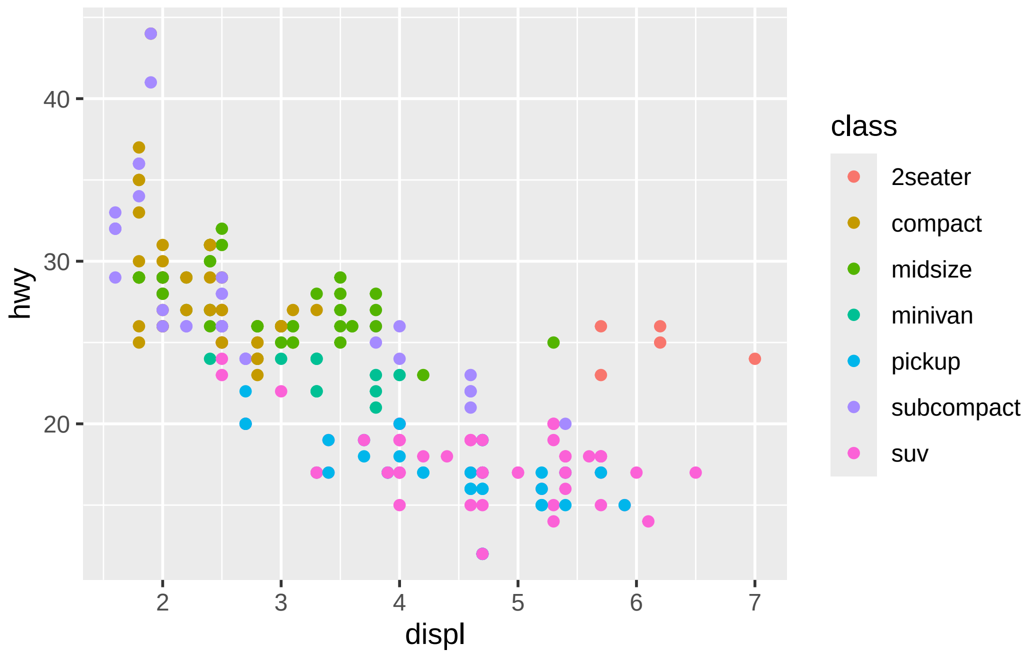
```
ggplot(mpg, aes(x = displ, y = hwy)) +
 geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```

```
`geom_smooth()` using formula = 'y ~ x'
```



By changing the mapping aesthetics, we can also introduce different colour schemes for different groups in our data. For example, we would like to add a “third” dimension to the graph, by colour-coding the data by car class.

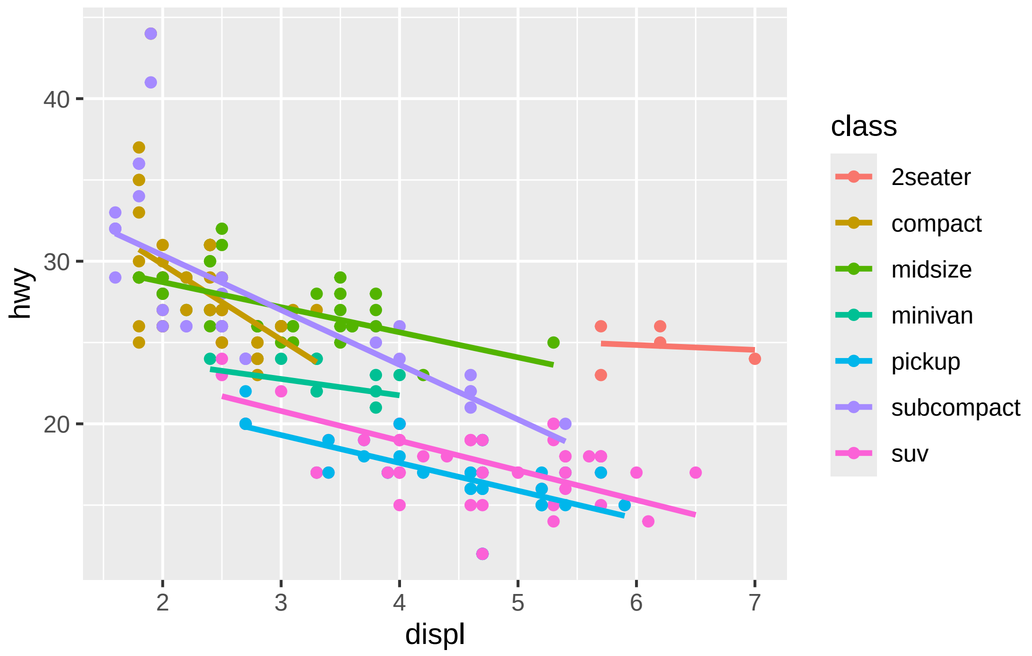
```
ggplot(mpg, aes(x = displ, y = hwy, colour = class)) +
 geom_point()
```



If you want to save the planet, don't drive an SUV or Pickup truck given that they have the lowest miles per gallon statistics. Using the colour option in the mapping aesthetics, all layers will apply the same grouping options. If we add our `geom_smooth` layer again, we do not have to specify the colouring!

```r
ggplot(mpg, aes(x = displ, y = hwy, colour = class)) +
 geom_point() +
  geom_smooth(method = "lm", se= FALSE)
```

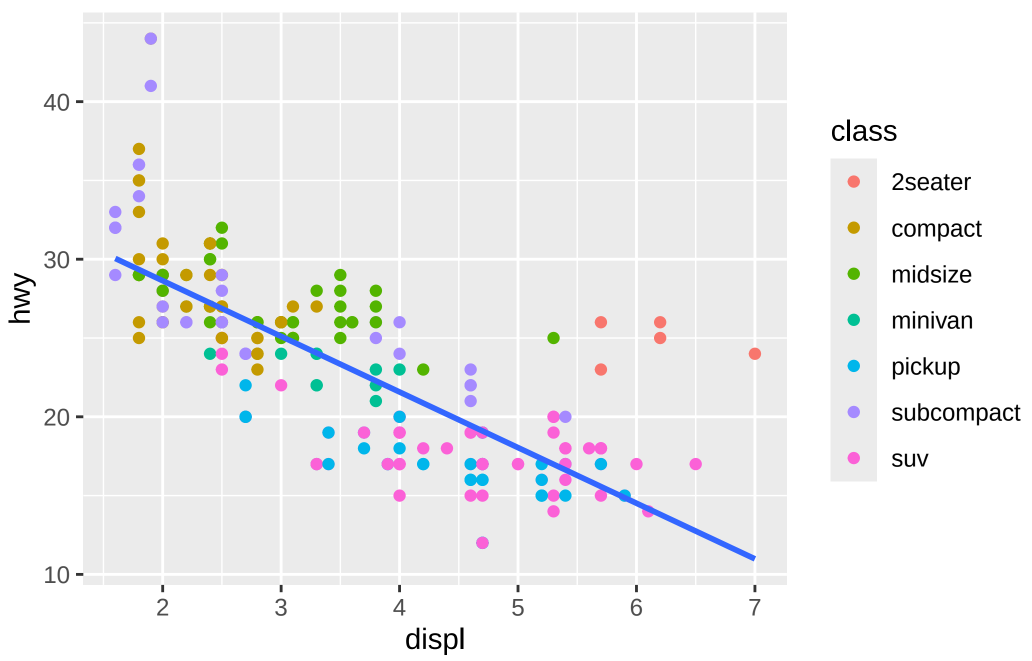
```
`geom_smooth()` using formula = 'y ~ x'
```



If you would like to add only one overall regression line using the whole data, but keeping the colour-coding, we have to make a small adjustment. We move the mapping aesthetics option `aes(..., colour = class)` into the `geom_point()` layer, so that colouring with respect to car class only applies to the points, and not to `geom_smooth()`.

```
ggplot(mpg, aes(x = displ, y = hwy)) +
 geom_point(aes(colour = class)) +
  geom_smooth(method = "lm",se = FALSE)
```

```
`geom_smooth()` using formula = 'y ~ x'
```

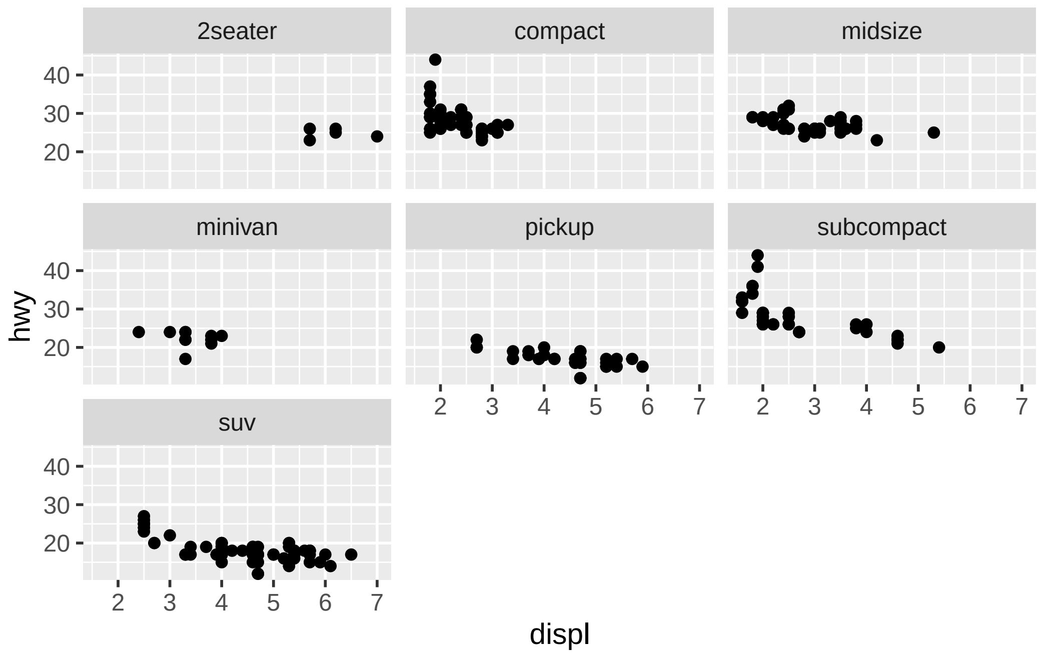


Faceting

Another popular method for plotting data for different groups is faceting. This means that we split the data into subsets and produce the same plot for each subset of the data. A great example for the usefulness of the method can be found her: https://www.scientificamerican.com/article/climate-change-drives-escalating-drought/

Before we can produce such advanced graphs, let's look at the basics first:

```r
ggplot(mpg, aes(displ,hwy)) +
  geom_point() +
  facet_wrap(~class)
```



Student Exercises

Import the *region_data.csv* dataset from https://github.com/rieglerr/gog. This dataset was sourced from https://www.ons.gov.uk/ and simplified to only contain aggregated regional data on gross disposable household income per capita for the period 1997 to 2022. Note that the data is measured in GBP at current prices.

Complete the following tasks:

1. Load the dataset
2. Look at the data
3. Create a plot using the ggplot command. Add the *year* variable to the x axis, and *GDHI_pc* on the y-axis and the `colour = Region_name` option to the mapping aesthetics to illustrate regional differences. Select a suitable geometric object to display the data. Experiment with different options from this list and select the one you think is most suitable for:

- `geom_area`
- `geom_line`
- `geom_point`

4. Use faceting to create a plot for each region. We do not need a separate legend, we can surpress it by adding the layer `theme(legend.position="none")`.

5. In what regions did we find the largest increase in gross disposable household income per capita?

Exercise solutions

1

```r
region_data <- read.csv("data/region_data.csv")
```

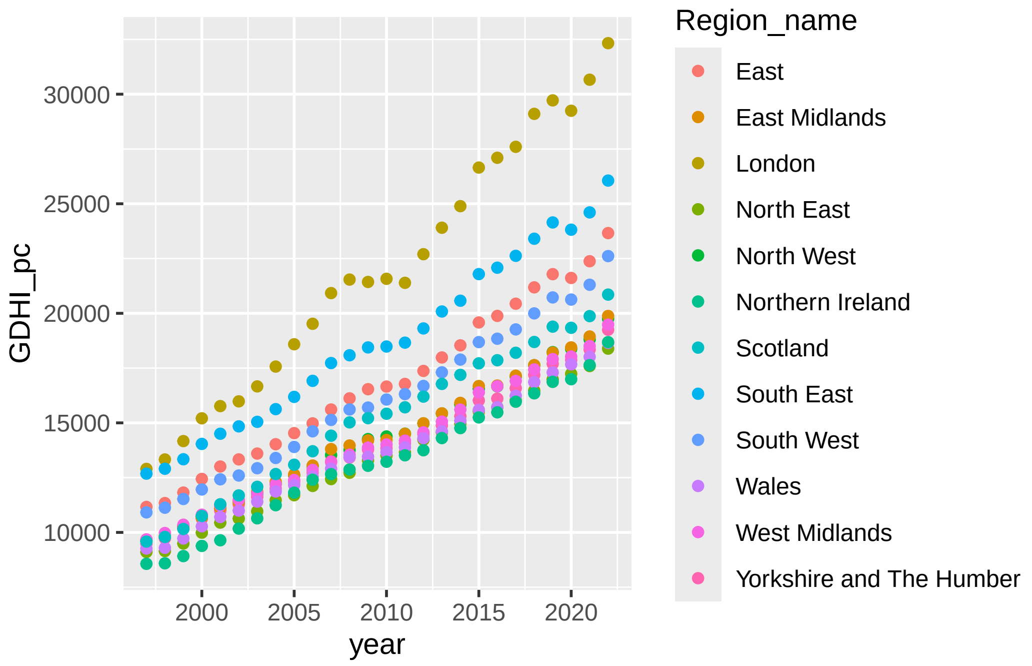
2

```r
head(region_data)
```

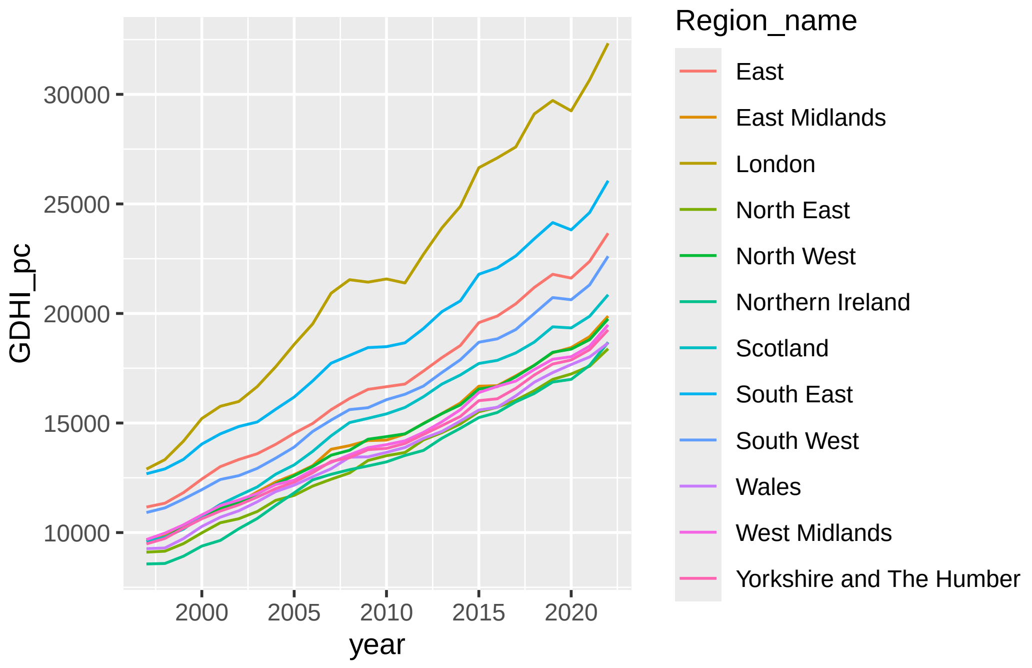
```
  Region_name year GDHI_pc
1  North East 1997    9107
2  North East 1998    9150
3  North East 1999    9494
4  North East 2000    9987
5  North East 2001   10448
6  North East 2002   10631
```

3

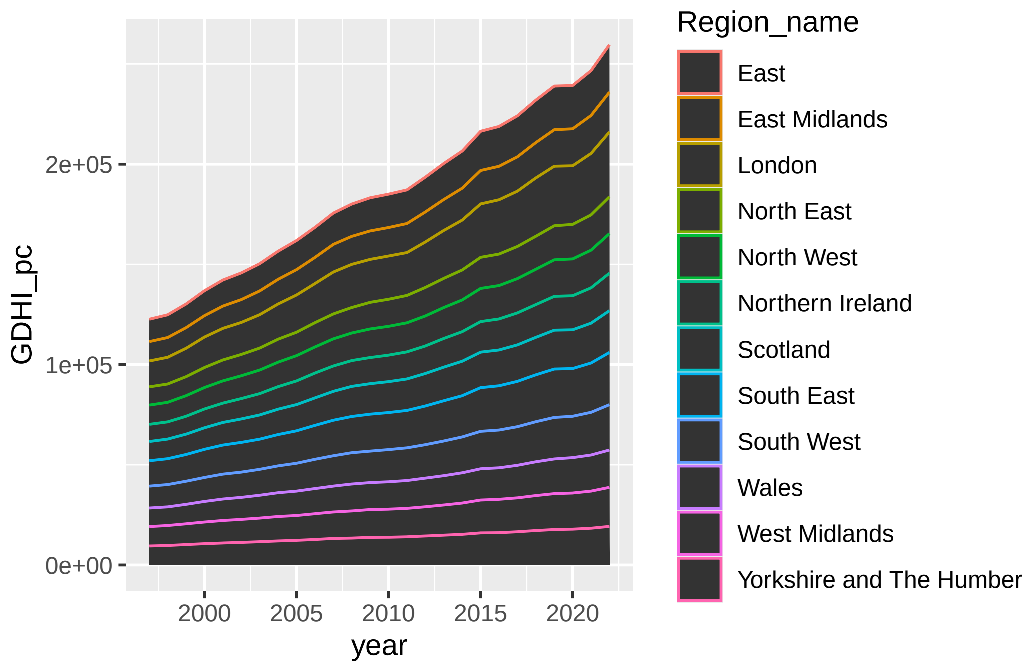
```r
ggplot(region_data, aes(x = year, y = GDHI_pc, colour = Region_name)) +
  geom_point()
```



```r
ggplot(region_data, aes(x = year, y = GDHI_pc, colour = Region_name)) +
  geom_line()
```



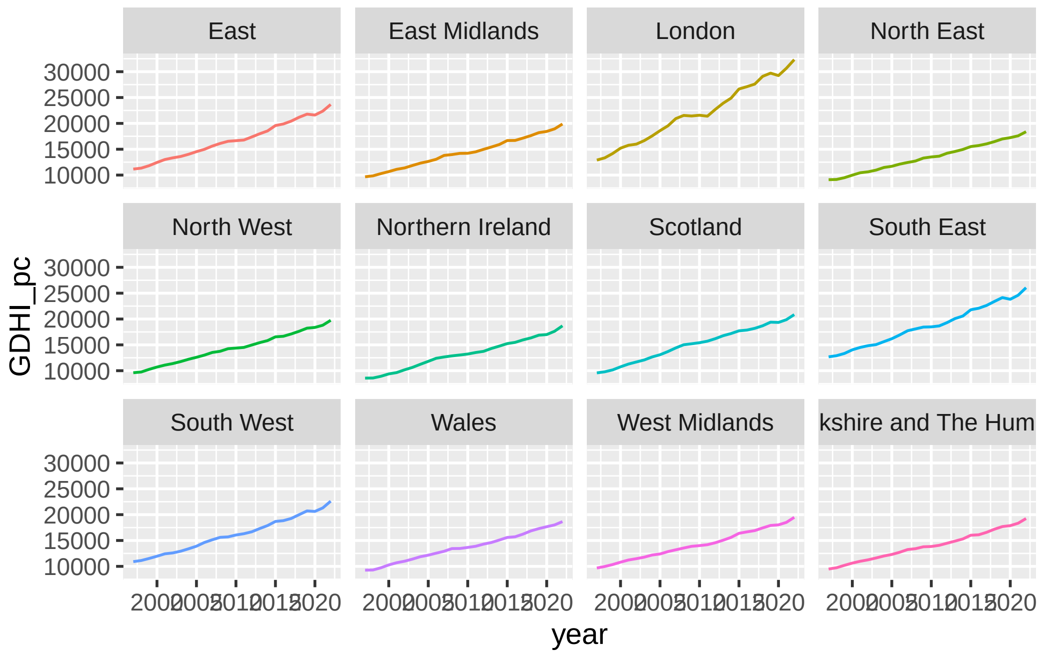
```r
ggplot(region_data, aes(x = year, y = GDHI_pc, colour = Region_name)) +
  geom_area()
```



`geom_line()` is most suitable for the data.

4

```r
ggplot(region_data, aes(x = year, y = GDHI_pc, colour = Region_name)) +
  geom_line() +
  facet_wrap(~Region_name) +
  theme(legend.position="none")
```



5

London has experienced the largest increase in GDHI per capita. With the exception of the South East, most regions have a significantly lower disposable income.

Themes

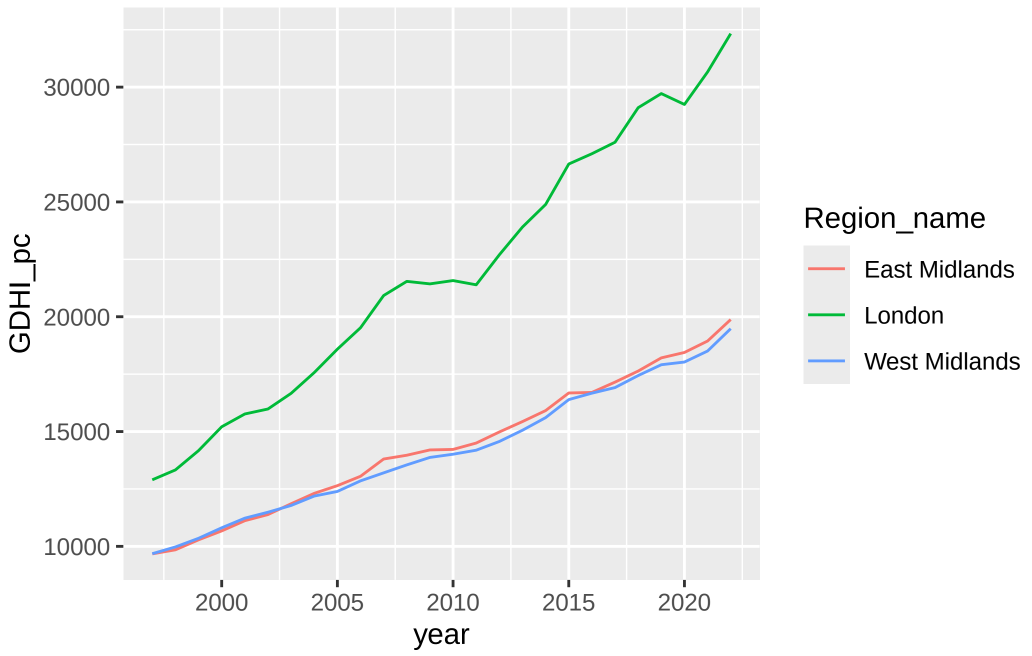
Using themes can facilitate the generation of professional graphics. Institutions like the BBC have their own themes. That means any BBC journalist can produce charts that follow the official BBC guidelines easily!

The BBC plot package is not part of the standard R repository, but can be easily downloaded from the BBC github account.

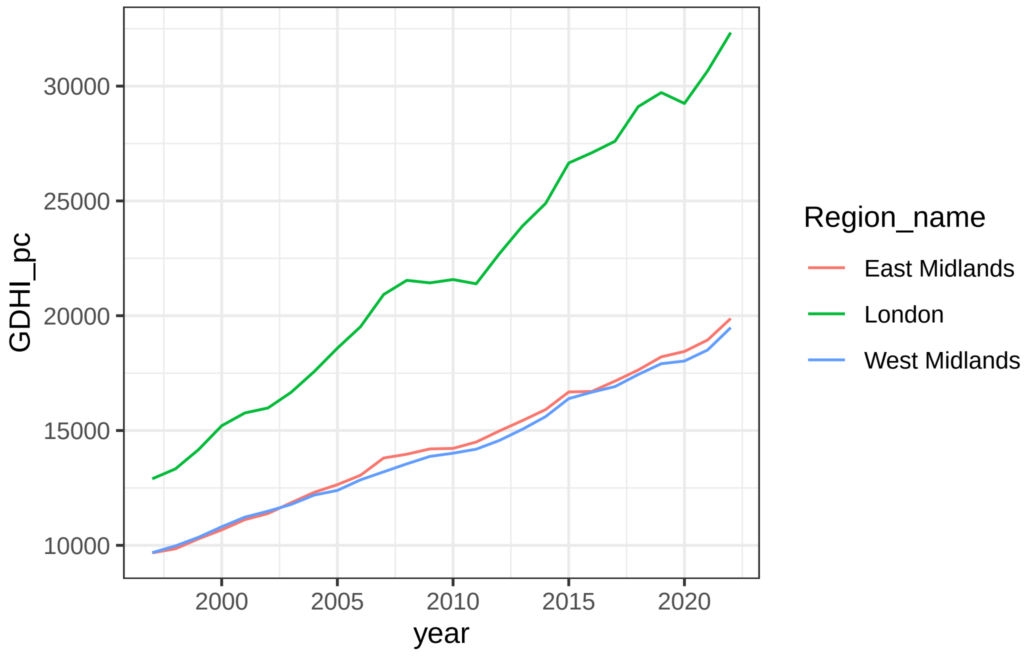
```
library(bbplot) # activate bbc style theme
```

```
region_data_sel <- read.csv("data/region_data_sel.csv")
```

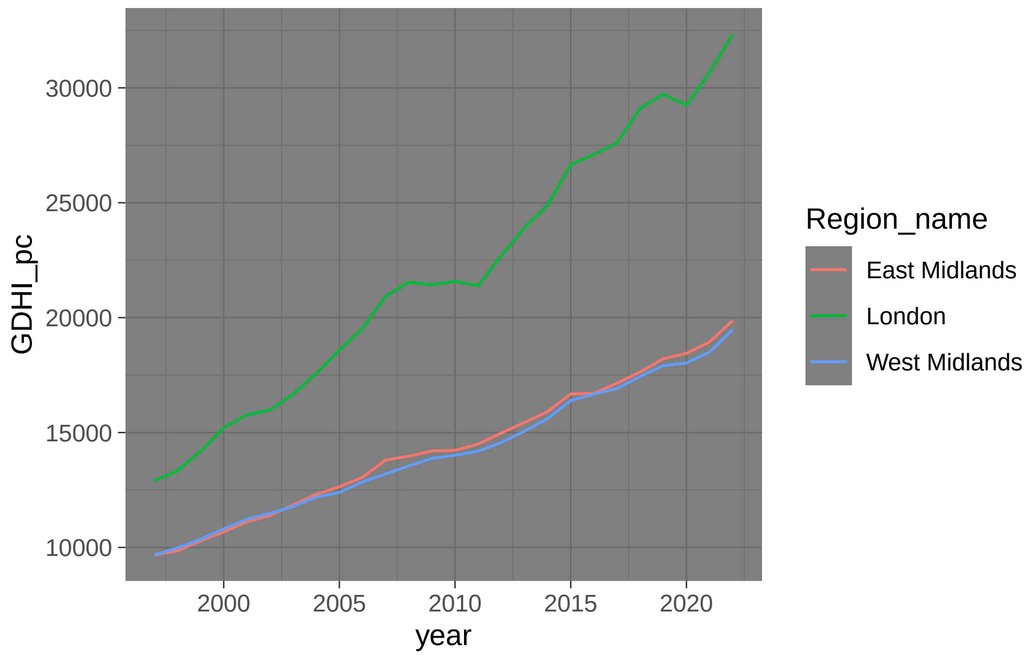
```
ggplot(region_data_sel, aes(x = year, y = GDHI_pc, colour = Region_name)) +
  geom_line()
```



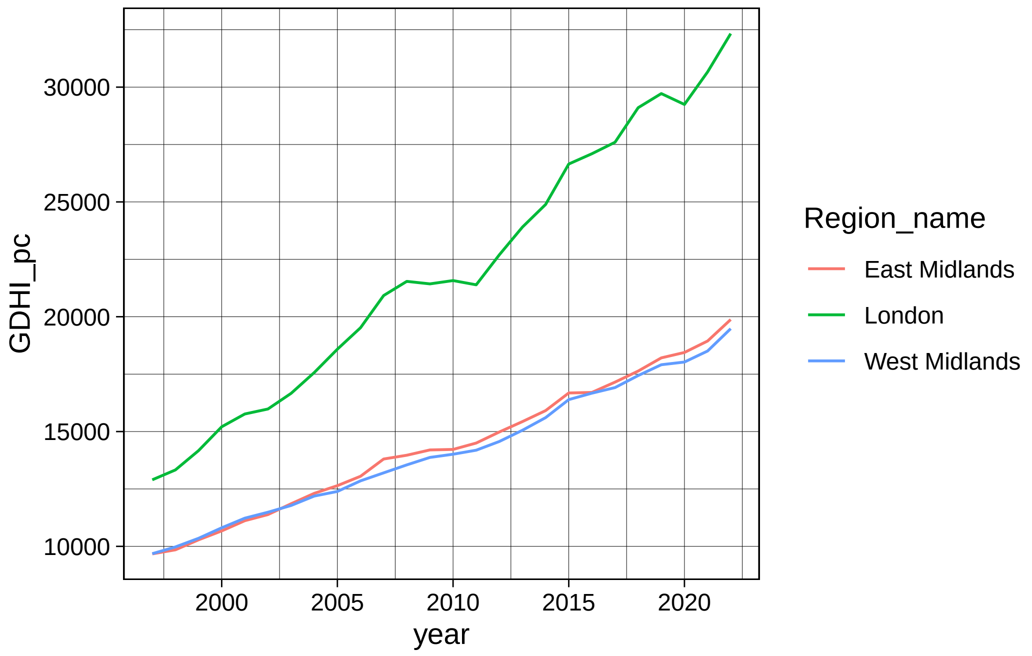
```r
ggplot(region_data_sel, aes(x = year, y = GDHI_pc, colour = Region_name)) +
  geom_line() +
  theme_bw()
```



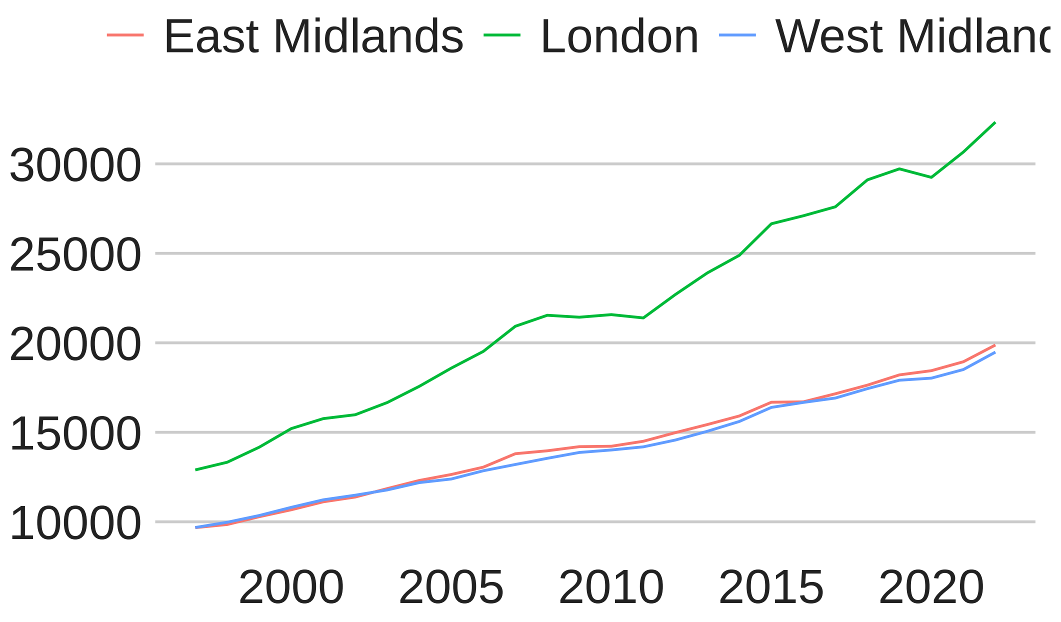
```r
ggplot(region_data_sel, aes(x = year, y = GDHI_pc, colour = Region_name)) +
  geom_line() +
  theme_dark()
```



```r
ggplot(region_data_sel, aes(x = year, y = GDHI_pc, colour = Region_name)) +
  geom_line() +
  theme_linedraw()
```



```r
ggplot(region_data_sel, aes(x = year, y = GDHI_pc, colour = Region_name)) +
  geom_line() +
  bbc_style()
```



The final touches

What about further annotations as the final touches for your plots? When you create graphs you have to think like an economist, i.e. what are the opportunity costs of your actions. Visually, you can do nearly anything you like with R's ggplot2 package, however, you will reach a point where some small annotations may be too time consuming. These could be swiftly done with other graphics packages like GIMP, Adobe Illustrator or even MS Paint.

As Keyes (2024) states: "Get yourself 90 percent of the way there with ggplot and then use Illustrator, Figma, or a similar tool to finish your work." Get the aesthetics, layers and a suitable theme so that you can easily replicate your work our use it as a template for updated data you are using. If you add annotations, that are very case specific, i.e. cannot be used again for other variants of your plot, then consider using an external programme.

Bubble charts

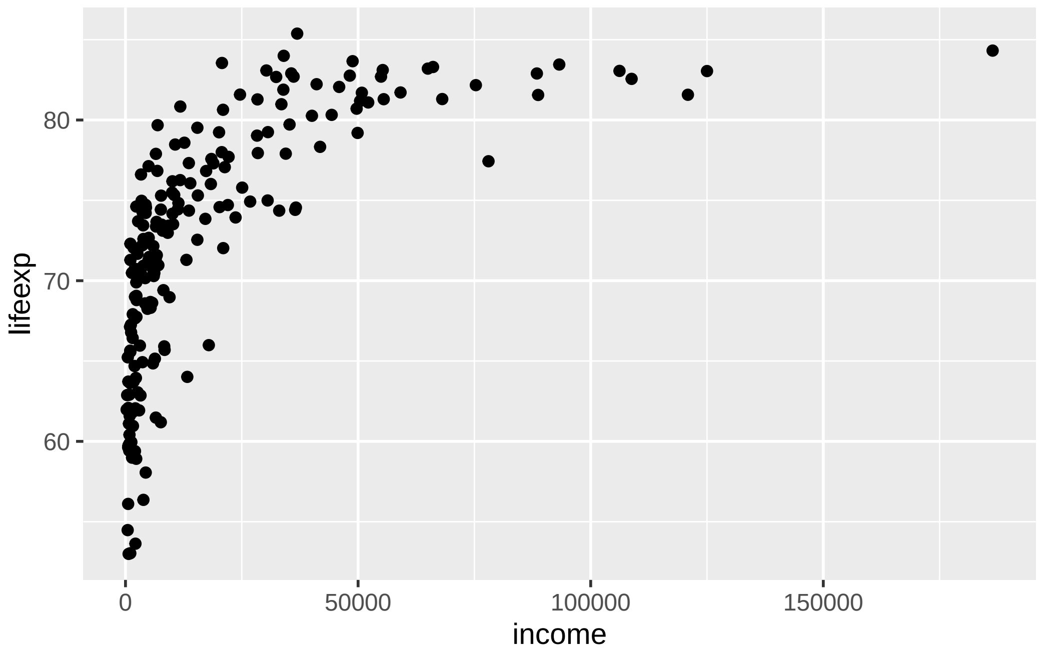
We are following now the example from https://data.europa.eu/apps/data-visualisation-guide/grammar-of-graphics-in-practice-ggplot2, with more recent data from the World Development Indicators of the World Bank.

Let's load the data:

```r
bubble.chart.data <- read.csv("data/bubble.chart.data.csv")
```

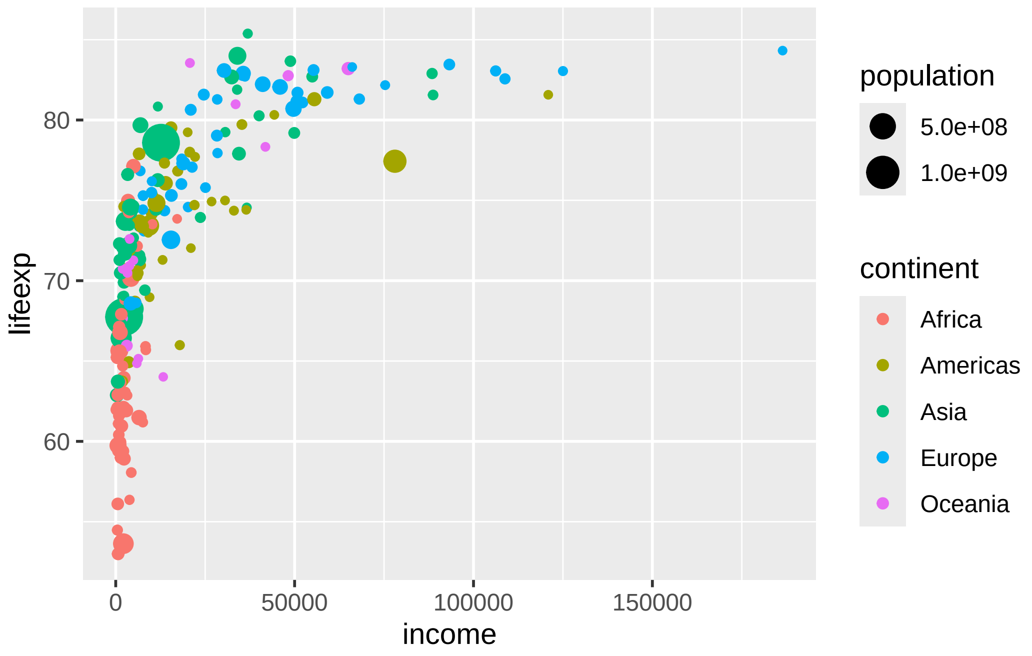
Now let's create a two dimensional graph to plot the relationship between income (measured as GDP per capita in current US Dollars) and life expectancy in years. The data is from 2022.

```r
ggplot(data = bubble.chart.data, aes(x = income, y = lifeexp)) +
  geom_point()
```



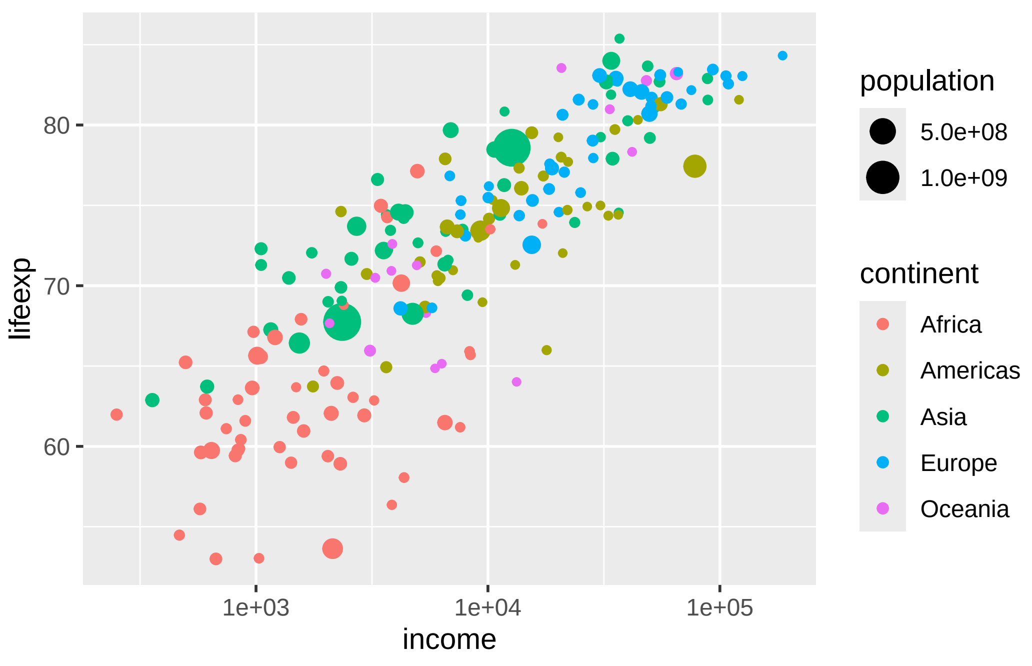
To find some interesting patterns, we want to add the dimensions of population size and continent to the graph. To do this, we use again the `colour = continent` aes option. We also want that the size of the "points" we have added to depend on the population size. This is achieved by adding also `size = population` aes option.

```
ggplot(data = bubble.chart.data, aes(x = income, y = lifeexp, size = population, colour = co
  geom_point()
```



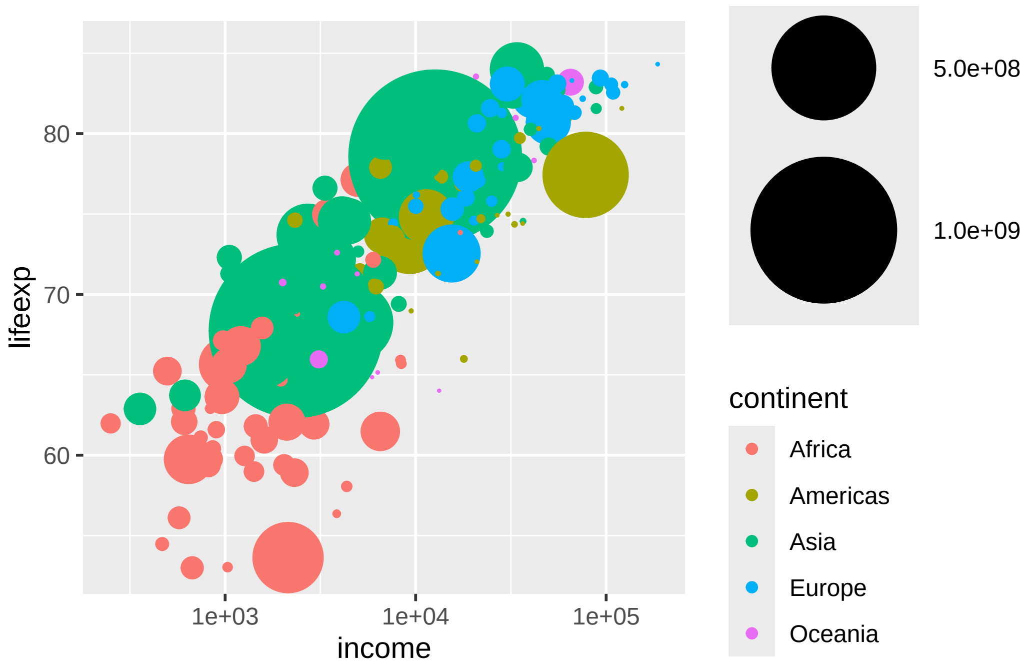
We are happy (I hope!) with the aesthetics, now we can modify the scaling. At the moment, the graph is very crowded on the left-hand side. We can adjust the x-axis scale to improve clarity. Let's use a logarithmic scale! Note the layered method we use, by adding our rescaling command after the `geom_point() +`.

```
ggplot(data = bubble.chart.data, aes(x = income, y = lifeexp, size = population, colour = co
  geom_point() +
  scale_x_log10()
```



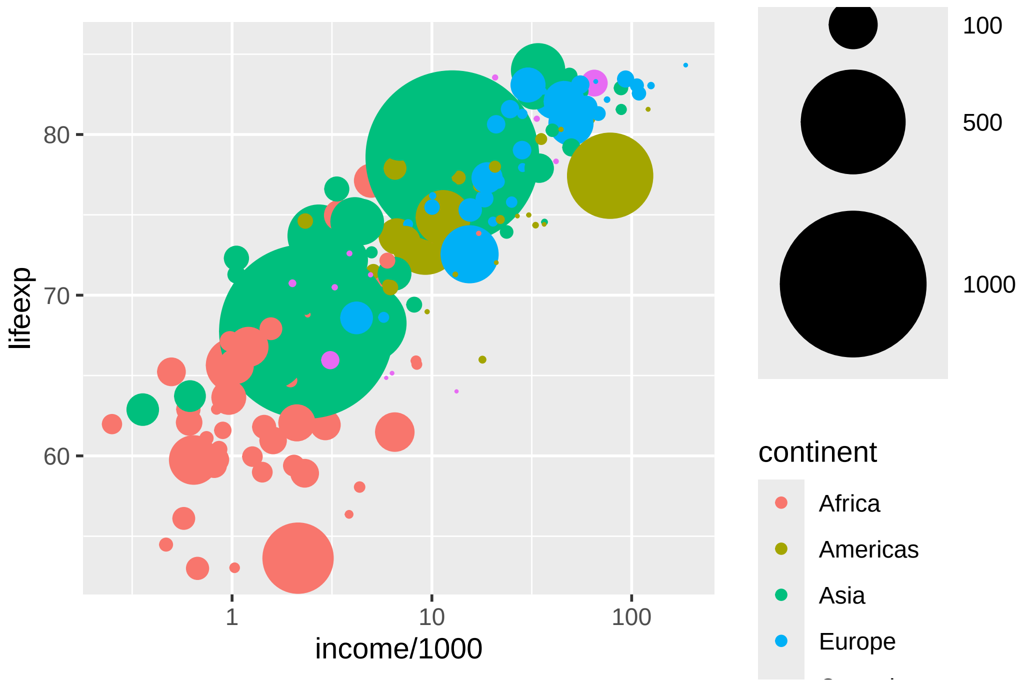
At the moment, the size of each point is just allocated to an ordinal scale, which only tells us which countries are large and which countries are small. We would like to change this, so that the point size actually represents the population size. We want the size to be a continuous scale. However, we set a maximum size so that we do not lose clarity.

```
ggplot(data = bubble.chart.data, aes(x = income, y = lifeexp, size = population, colour = co
  geom_point() +
  scale_x_log10() +
  scale_size_area(max_size = 30, name = "Population")
```



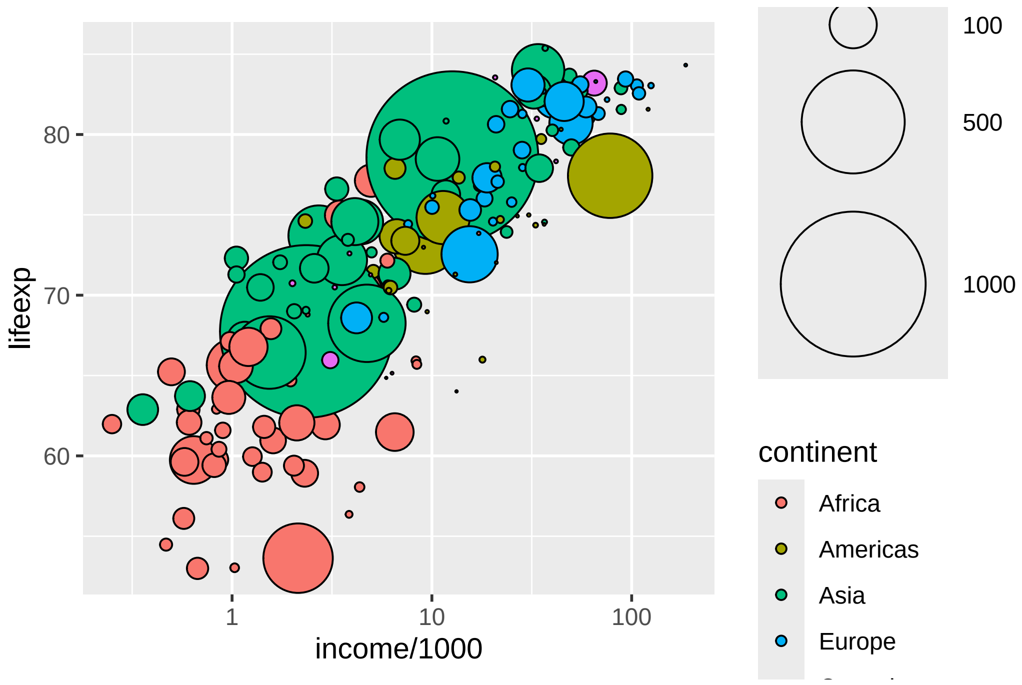
We try to avoid scientific numbers, thus we are going to improve the labelling by changing the units we are using. E.g., we are going to measure population in millions and income in thousands. Furthemore, we want to give some more intuitive legend items through the breaks command.

```
ggplot(data = bubble.chart.data, mapping = aes(
    x = income/1000,
    y = lifeexp,
    size = population/1000000,
    colour = continent)) +
  geom_point() +
  scale_x_log10() +
  scale_size_area(
    max_size = 30,
    name = "Population (millions)",
    breaks = c(10, 100, 500, 1000))
```



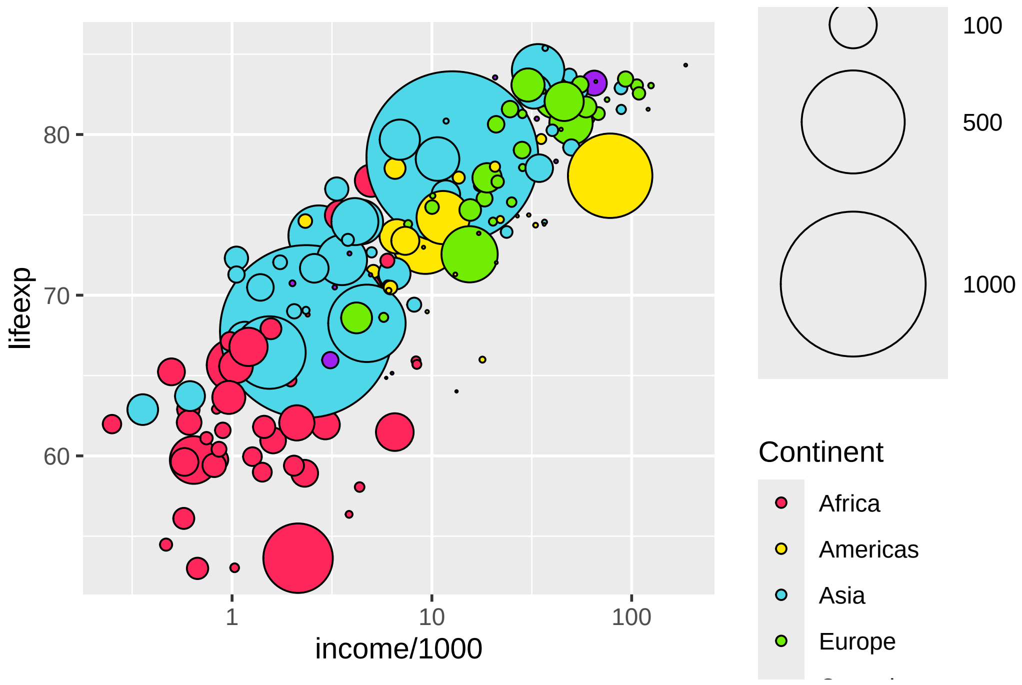
Ok, now we are happy with the scaling (I hope!!!). Let's continue with improving the styling of the chart. At the moment, it is difficult to differentiate all our observations. E.g. any small green point will be eaten up by the large green points! We will change therefore the style of our geometric objects. We will use shape 21 for adding a full circle around the points. See https://ggplot2.tidyverse.org/articles/ggplot2-specs.html#sec:shape-spec for different shape values. You will have realised that we changed one of the aes options: instead of colour, we used fill = continent. The reason is that we do not want the circle shape to be a colour specific to a continent, but the fill within the circle. If you stick to the colour option, you will see that we only have a colourful circle without a filling!

```
ggplot(data = bubble.chart.data, mapping = aes(
    x = income/1000,
    y = lifeexp,
    size = population/1000000,
    fill = continent)) +
  geom_point(shape = 21) +
  scale_x_log10() +
  scale_size_area(
    max_size = 30,
    name = "Population (millions)",
    breaks = c(10, 100, 500, 1000))
```



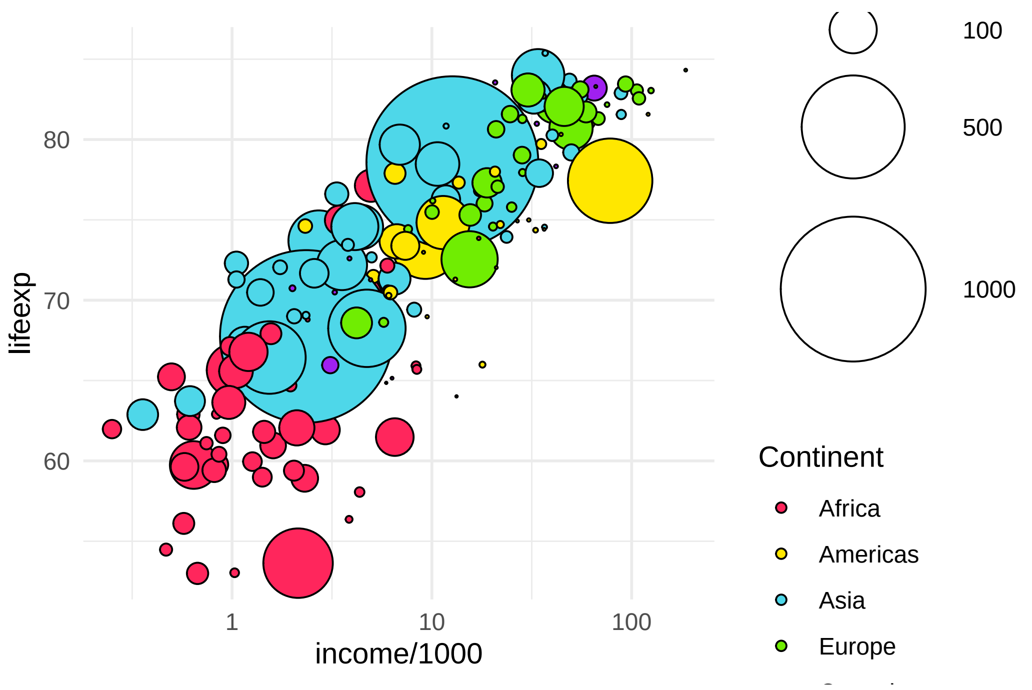
Not satisfied with the colours? We can set them manually (or use specific colour themes). We will set them manually first.

```r
ggplot(data = bubble.chart.data, mapping = aes(
    x = income/1000,
    y = lifeexp,
    size = population/1000000,
    fill = continent)) +
  geom_point(shape = 21) +
  scale_x_log10() +
  scale_size_area(
    max_size = 30,
    name = "Population (millions)",
    breaks = c(10, 100, 500, 1000)) +
  scale_fill_manual(
    values = c("#FF265C", "#FFE700", "#4ED7E9", "#70ED02", "purple"),
    name = "Continent")
```



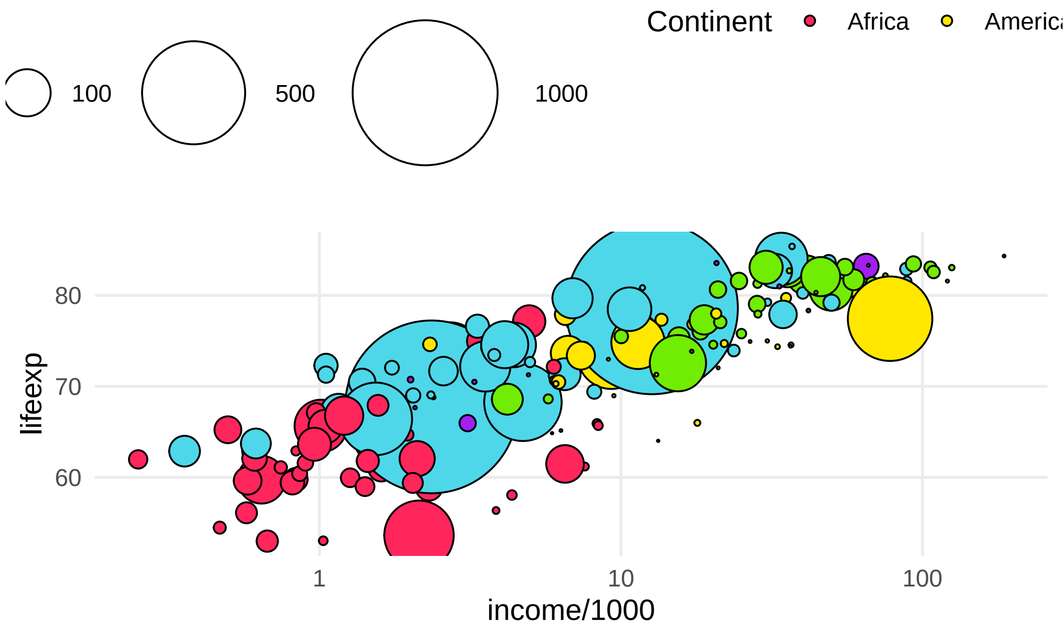
For the final touches, we will use now the minimal theme. Remember, keep a graph as simple as possible!

```
ggplot(data = bubble.chart.data, mapping = aes(
    x = income/1000,
    y = lifeexp,
    size = population/1000000,
    fill = continent)) +
  geom_point(shape = 21) +
  scale_x_log10() +
  scale_size_area(
    max_size = 30,
    name = "Population (millions)",
    breaks = c(10, 100, 500, 1000)) +
  scale_fill_manual(
    values = c("#FF265C", "#FFE700", "#4ED7E9", "#70ED02", "purple"),
    name = "Continent") +
  theme_minimal()
```



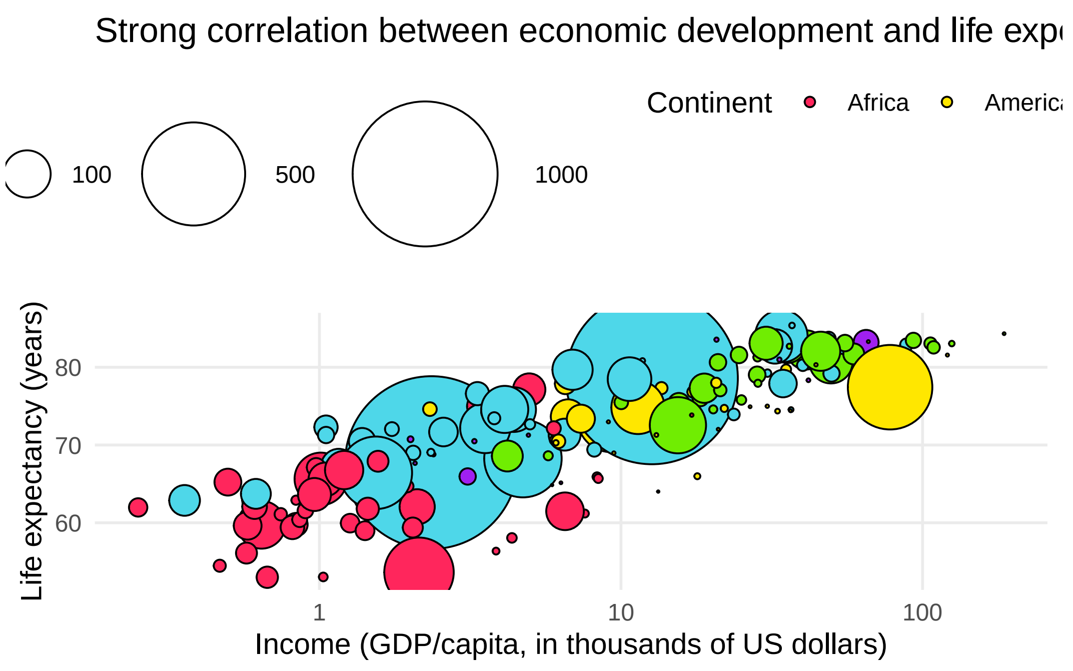
We can manually adjust some of the themes options, e.g. let's move the legend onto the top and remove the minor grid lines.

```
ggplot(data = bubble.chart.data, mapping = aes(
    x = income/1000,
    y = lifeexp,
    size = population/1000000,
    fill = continent)) +
  geom_point(shape = 21) +
  scale_x_log10() +
  scale_size_area(
    max_size = 30,
    name = "Population (millions)",
    breaks = c(10, 100, 500, 1000)) +
  scale_fill_manual(
    values = c("#FF265C", "#FFE700", "#4ED7E9", "#70ED02", "purple"),
    name = "Continent") +
  theme_minimal() +
  theme(panel.grid.minor = element_blank(),
        legend.position = "top")
```



Finally, let's improve the axes lables and add a title, and we are done!

```r
ggplot(data = bubble.chart.data, mapping = aes(
    x = income/1000,
    y = lifeexp,
    size = population/1000000,
    fill = continent)) +
  geom_point(shape = 21) +
  scale_x_log10() +
  scale_size_area(
    max_size = 30,
    name = "Population (millions)",
    breaks = c(10, 100, 500, 1000)) +
  scale_fill_manual(
    values = c("#FF265C", "#FFE700", "#4ED7E9", "#70ED02", "purple"),
    name = "Continent") +
  theme_minimal() +
  theme(panel.grid.minor = element_blank(),
        legend.position = "top") +
  labs(x ="Income (GDP/capita, in thousands of US dollars)",
       y ="Life expectancy (years)") +
  ggtitle("Strong correlation between economic development and life expectancy")
```



Animation

Let's finish off today's session with some animations! We need a new package before

```r
library(gganimate)
```

```
Warning: package 'gganimate' was built under R version 4.4.2
```

```r
library(gifski)
```

```
Warning: package 'gifski' was built under R version 4.4.2
```

```r
animate_data <- read.csv("data/animate.csv")
```

```r
p <- ggplot(data = animate_data, mapping = aes(
  x = income/1000,
  y = lifeexp,
  size = population/1000000,
  fill = continent)) +
  geom_point(shape = 21) +
  scale_x_log10() +
  scale_size_area(
    max_size = 30,
    name = "Population (millions)",
    breaks = c(10, 100, 500, 1000)) +
  scale_fill_manual(
    values = c("#FF265C", "#FFE700", "#4ED7E9", "#70ED02", "purple"),
    name = "Continent") +
  theme_minimal() +
  theme(panel.grid.minor = element_blank(),
        legend.position = "none") +
  labs(title = "Year: {frame_time}",
       x = "Income (GDP/capita, PPP (constant 2021 international $))",
       y = "Life expectancy (years)") +
  transition_time(year)

p_anim <- animate(p, renderer = gifski_renderer())
anim_save("animation_frames/my_animation.gif", animation = p_anim)
```