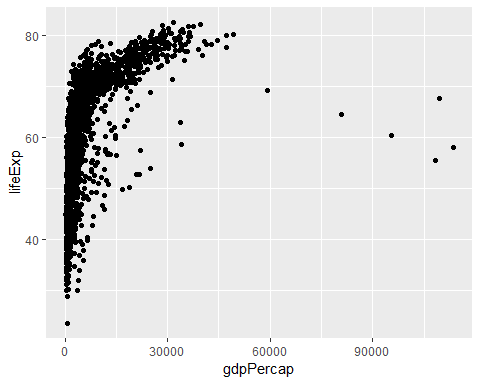
Datavis - advanced tutorial

This is part 2 - if you are comfortable with everything in part 1, you can move onto this tutorial.

# Building an animated plot

We will start building a plot with geom\_point:

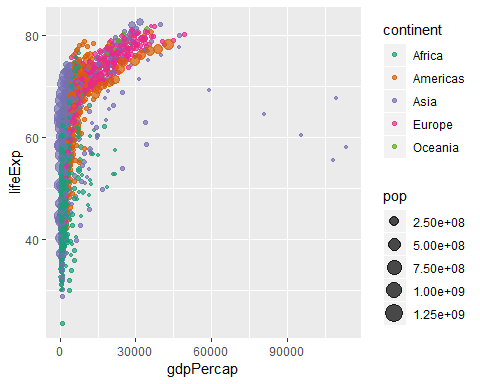
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +  
 geom\_point()



**Exercise**: Change the size of the points according to the variable pop, and the colour according to the continent.

Now we will make the points a bit more transparent, by setting alpha. We will also change the colour of the points:

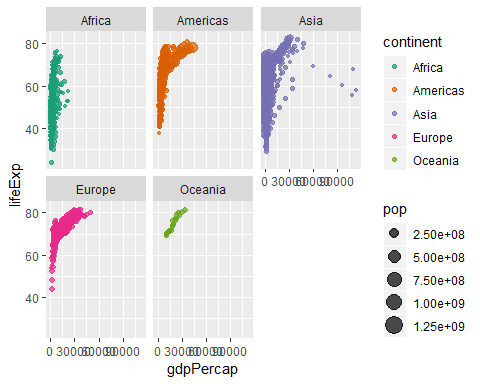
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp, size = pop, colour = continent)) +  
 geom\_point(alpha = .7) +  
 scale\_colour\_brewer(palette = "Dark2")



**Exercise**: Change the palette to a different one. Check <http://colorbrewer2.org/> for more palettes, and pick one that is colour-blind friendly.

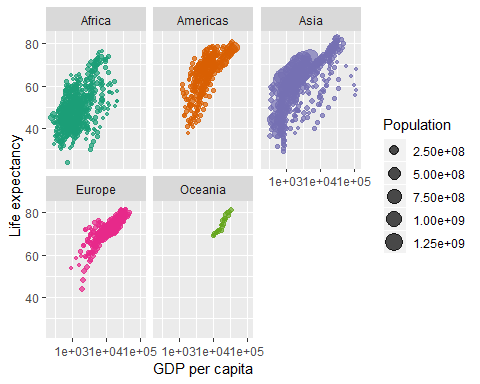
Now we will make subplots, by splitting the data according to the continent:

ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp, size = pop, colour = continent)) +  
 geom\_point(alpha = .7) +  
 facet\_wrap(~ continent) + # we need ~ in front of the variable, as we could facet by another one  
 # then the format would be facet\_wrap(variable1 ~ variable2)  
 scale\_colour\_brewer(palette = "Dark2")



We will change a few more things, to make sure this will look really nice in the end:

ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp, size = pop, colour = continent)) +  
 geom\_point(alpha = .7) +  
 facet\_wrap(~ continent) +  
 scale\_colour\_brewer(palette = "Dark2") +  
 scale\_size(name = "Population") +  
 guides(colour = "none") +  
 scale\_x\_log10() +  
 labs(x = 'GDP per capita', y = 'Life expectancy')



**Exercise**: What happens in each of the last 4 lines of code?

Finally, we will add the code to make the plot show each year individually. We use the function transition\_time from the gganimate package:

ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp, size = pop, colour = continent)) +  
 geom\_point(alpha = .7) +  
 facet\_wrap(~ continent) +  
 scale\_colour\_brewer(palette = "Dark2") +  
 scale\_size(name = "Population") +  
 guides(colour = "none") +  
 scale\_x\_log10() +  
 labs(title = 'Year: {frame\_time}', x = 'GDP per capita', y = 'Life expectancy') +  
 transition\_time(year)

We can also save the output as a gif:

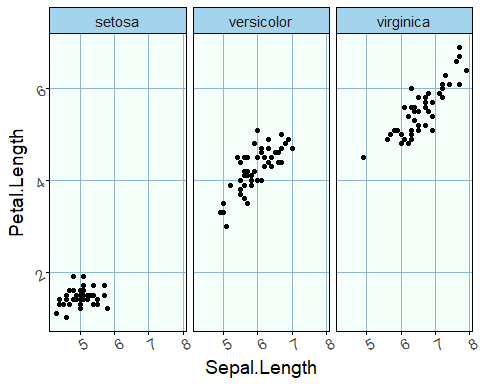
p <- ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp, size = pop, colour = continent)) +  
 geom\_point(alpha = .7) +  
 facet\_wrap(~ continent) +  
 scale\_colour\_brewer(palette = "Dark2") +  
 scale\_size(name = "Population") +  
 guides(colour = "none") +  
 scale\_x\_log10() +  
 labs(title = 'Year: {frame\_time}', x = 'GDP per capita', y = 'Life expectancy') +  
 transition\_time(year)  
p <- animate(p)  
anim\_save("YayImadeaGIF.gif")

# A completely custom ggplot2 theme

You can make customised plots by changing the theme. This will not change how the data are displayed, but how the plot itself looks. It is useful if you want all plots in your work to have the same look.

Here is an example of a customised plot, with annotation for the code:

ggplot(data = iris, aes(x = Sepal.Length, y = Petal.Length)) +  
 geom\_point() +  
 facet\_wrap( ~ Species) +  
 scale\_colour\_gradient(low = "green", high = "blue", name = "Petal width") +  
 # this is similar to a plot we made earlier  
   
 theme(  
 text = element\_text(size = 14),  
 # this changes the font size to 14  
   
 axis.text = element\_text(angle = 30, size = rel(0.8)),  
 # this changes the angle of the axis labels (often useful for bar plots),   
 # and makes the font slightly smaller than the rest of the plot  
   
 axis.ticks = element\_line(colour = "black"),  
   
 plot.background = element\_rect(fill = "white"),  
 #this is the colour behind the plot  
   
 panel.background = element\_rect(fill = "mintcream"),  
 # this is the colour inside the plots  
   
 panel.border = element\_rect(fill = NA, colour = "black"),  
   
 panel.grid.major = element\_line(colour = "lightskyblue3", size = .1),  
   
 panel.grid.minor = element\_line(colour = "mintcream"),  
 # the minor grid lines - same colour as each panel  
   
 strip.background = element\_rect(  
 fill = "lightskyblue2",  
 colour = "black",  
 size = 0.5))



#this refers to the box with the heading above each facet

**Exercise: make a nice looking theme which you can use for your own work**

# Making world maps in R (Chloropleth maps)

You will need the packages rworldmap and ggplot2. We will plot some gapminder data, so will need that package too:

library(rworldmap)  
library(tidyverse)  
library(gapminder)

## Load map data

This function will give you the data for plotting each country:

map.all <- map\_data(map = "world")  
head(map.all) ## region denotes the country

## long lat group order region subregion  
## 1 -69.89912 12.45200 1 1 Aruba <NA>  
## 2 -69.89571 12.42300 1 2 Aruba <NA>  
## 3 -69.94219 12.43853 1 3 Aruba <NA>  
## 4 -70.00415 12.50049 1 4 Aruba <NA>  
## 5 -70.06612 12.54697 1 5 Aruba <NA>  
## 6 -70.05088 12.59707 1 6 Aruba <NA>

## Get your own data in order

Your data need to match the country names in map.all. This is probably what will take you longest to sort out, but we will use some good enough data today:

head(gapminder) ## This is the gapminder dataset

## # A tibble: 6 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.

gapminder <- rename(gapminder, region = country) ## we need to rename country to region, so we can merge gapminder and map.all

## Merge datasets

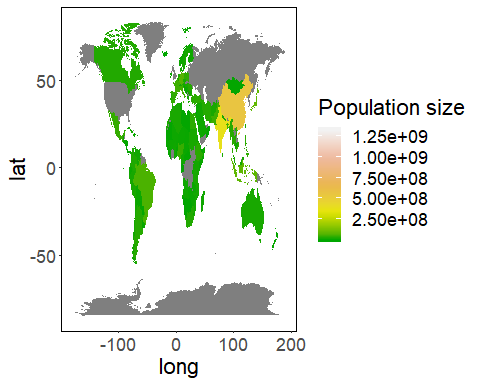
Then merge the two dataframes map.all and all:

map.all <- left\_join(map.all, gapminder, by = "region")   
head(map.all)

## long lat group order region subregion continent year lifeExp  
## 1 -69.89912 12.45200 1 1 Aruba <NA> <NA> NA NA  
## 2 -69.89571 12.42300 1 2 Aruba <NA> <NA> NA NA  
## 3 -69.94219 12.43853 1 3 Aruba <NA> <NA> NA NA  
## 4 -70.00415 12.50049 1 4 Aruba <NA> <NA> NA NA  
## 5 -70.06612 12.54697 1 5 Aruba <NA> <NA> NA NA  
## 6 -70.05088 12.59707 1 6 Aruba <NA> <NA> NA NA  
## pop gdpPercap  
## 1 NA NA  
## 2 NA NA  
## 3 NA NA  
## 4 NA NA  
## 5 NA NA  
## 6 NA NA

## Now you can make the plot:

ggplot() + geom\_map(data = map.all, map = map.all,   
 aes(map\_id = region, x = long, y = lat, fill = pop)) +   
 ## map\_id is the variable name of the countries,  
 ## fill will colour in each country.   
 ## Use colour = varname to change colour of the outline  
 scale\_fill\_gradientn(colours=terrain.colors(5)) +   
 ## this will change the colours in the gradient  
 labs(fill = "Population size") + ## changes the name of the legend  
 theme\_classic() + ## change the overall look of the plot  
 theme(panel.background = element\_rect(colour = "black", fill = "NA"),   
 axis.ticks.length = unit(.05, "cm"),   
 text = element\_text(size = 16.5)) ## change some more of the defaults



**Exercise: using the ggplot2 cheatsheet, can you change the colour gradient using some of the inbuilt colour schemes?**