Data visualization

2025-01-17

M1 MIDS/MFA/LOGOS Université Paris Cité Année 2024 Course Homepage Moodle



Objectives

This workbook introduces visualization according to the *Grammar of Graphics* framework.

Using ggplot2, we reproduce Rosling's gapminder talk.

This is an opportunity to develop the layered construction of graphical objects.

Grammar of Graphics

We will use the *Grammar of Graphics* approach to visualization

The expression *Grammar of Graphics* was coined by Leiland Wilkinson to describe a principled approach to visualization in Data Analysis (EDA)

A plot is organized around tabular data (a table with rows (observations) and columns (variables))

A plot is a graphical object that can be built layer by layer

Building a graphical object consists in *chaining* elementary operations

The acclaimed TED presentation by Hans Rosling illustrates the Grammar of Graphics approach

Visit https://www.youtube.com/embed/jbkSRLYSojo

We will reproduce the animated demonstration using

- ggplot2: an implementation of grammar of graphics in 'R
- plotly: a bridge between R and the javascript library D3.js
- Using plotly, opting for html ouput, brings the possibility of interactivity and animation

Setup

We will use the following packages. If needed, we install them.

```
stopifnot(
  require(tidyverse),
```

```
require(patchwork),
require(glue),
require(ggforce),
require(plotly),
require(ggthemes),
require(gapminder),
require(gapmepel)
)
```

The data we will use can be obtained by loading package gapminder

• Tip

If the packages have not yet been installed on your hard drive, install them. You can do that using base R install.packages() function:

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

It is often faster to use functions from package pak

```
install.packages("pak")
pak::pkg_install("tidyverse")
```

You need to understand the difference between installing and loading a package

Question

- How do we get the list of *installed* packages?
- How do we get the list of *loaded* packages?
- Which objects are made available by a package?

solution

The (usually very long) list of installed packages can be obtained by a simple function call.

```
df <- installed.packages()</pre>
head(df)
##
                 Package \ LibPath
## abind
                 "abind"
                                    "/home/boucheron/R/x86_64-pc-linux-gnu-library/4.4"
## arkhe "arkhe" "/home/boucheron/R/x86_64-pc-linux-gnu-library/4.4" ## arrow "arrow" "/home/boucheron/R/x86_64-pc-linux-gnu-library/4.4" ## ash "ash" "/home/boucheron/R/x86_64-pc-linux-gnu-library/4.4"
## AsioHeaders "AsioHeaders" "/home/boucheron/R/x86_64-pc-linux-gnu-library/4.4"
## askpass "askpass" "/home/boucheron/R/x86_64-pc-linux-gnu-library/4.4" ## Version Priority Depends
              "1.4-5" NA "R (>= 1.5.0)"
"1.6.0" NA "R (>= 3.5)"
"16.1.0" NA "R (>= 4.0)"
"1.0-15" NA NA
## abind
## arkhe
## arrow
## ash
                                       4_{NA}
## AsioHeaders "1.22.1-2" NA
## askpass "1.2.0" NA
                                              NA
```

Have a look at gapminder dataset

The gapminder table can be found at gapminder::gapminder

- A table has a schema: a list of named columns, each with a given type
- A table has a *content*: rows. Each row is a collection of items, corresponding to the columns

Question

Explore gapminder::gapminder, using glimpse() and head()

- glimpse() allows to see the schema and the first rows
- head() allows to see the first rows
- Use the pipe |> to chain operations

```
solution
Dataframes
gapminder <- gapminder::gapminder</pre>
glimpse(gapminder)
## Rows: 1,704
## Columns: 6
               <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", "
## $ country
## $ continent <fct> Asia, ~
## $ year
               <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 19$2, 1997, ~
## $ lifeExp
               <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.8~
               <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14$80372, 12~
## $ pop
## $ qdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, ~
gapminder |>
  glimpse()
## Rows: 1,704
## Columns: 6
               <fct> "Afghanistan", "Afghanistan", "Afghanistan", "
## $ country
## $ continent <fct> Asia, ~
               <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, ~
## $ year
## $ lifeExp
               <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.8~
               <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14$80372, 12~
## $ pop
## $ qdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, ~
gapminder |>
  head()
## # A tibble: 6 x 6
##
   country
               continent year lifeExp
                                              pop gdpPercap
                 <fct> <int> <dbl>
                                            \langle int \rangle
                                                     <dbl>
     <fct>
## 1 Afghanistan Asia
                           1952
                                   28.8 8425333
                                                       779.
## 2 Afghanistan Asia
                           1957
                                  30.3 9240934
                                                       821.
                                   32.0 10267083
                                                       853.
## 3 Afghanistan Asia
                          1962
                           1967 34.0 11537966
## 4 Afghanistan Asia
                                                       836.
## 5 Afghanistan Asia
                            1972
                                   36.1 13079460
                                                       740.
## 6 Afghanistan Asia
                       1977 38.4 14880372
                                                       786.
Even an empty dataframe has a scheme:
gapminder |>
  head(0) >
  glimpse()
Rows: 0
Columns: 6
$ country
            <fct>
$ continent <fct>
$ year
            <int>
$ lifeExp
            <dbl>
$ pop
            <int>
$ gdpPercap <dbl>
# glimpse(head(gapminder, 0))
```

solution

The schema of a dataframe/tibble is the list of column names and classes. The content of a dataframe is made of the rows. A dataframe may have null content

```
gapminder |>
   filter(FALSE) |>
   glimpse()

## Rows: 0

## Columns: 6

## $ country <fct>
## $ continent <fct>
## $ year <int>
## $ lifeExp <dbl>
## $ pop <int>
## $ gdpPercap <dbl>
```

Get a feeling of the dataset

i Question

Pick two random rows for each continent using slice_sample()

solution

To pick a slice at random, we can use function slice_sample. We can even perform sampling within groups defined by the value of a column.

```
gapminder |>
  slice_sample(n=2, by=continent)
# A tibble: 10 x 6
   country
                        continent year lifeExp
                                                      pop gdpPercap
   <fct>
                                           <dbl>
                                                              <dbl>
                        <fct>
                                  <int>
                                                    <int>
                                                                849.
 1 Indonesia
                        Asia
                                   1962
                                            42.5 99028000
 2 Israel
                                            69.4
                                                  2310904
                                                              7106.
                        Asia
                                   1962
                                   1997
 3 Ireland
                        Europe
                                            76.1
                                                  3667233
                                                             24522.
 4 Italy
                                   1982
                                           75.0 56535636
                                                             16537.
                        Europe
 5 Eritrea
                        Africa
                                   1952
                                            35.9
                                                  1438760
                                                                329.
                        Africa
                                   1992
                                            39.7
                                                  6099799
                                                                927.
 6 Somalia
 7 Trinidad and Tobago Americas
                                   2002
                                            69.0
                                                  1101832
                                                              11461.
                                                  2500689
 8 Honduras
                        Americas
                                   1967
                                           50.9
                                                              2538.
 9 Australia
                        Oceania
                                   1962
                                            70.9 10794968
                                                              12217.
10 New Zealand
                        Oceania
                                   1987
                                            74.3
                                                  3317166
                                                              19007.
#< or equivalently</pre>
# gapminder |>
    group_by(continent) |>
    slice_sample(n=2)
```

Question

What makes a table *tidy*?

🥊 Tip

Have a look at Data tidying in R for Data Science (2nd ed.)

Question

Is the gapminder table redundant?

solution

gapminder is redundant: column country completely determines the content of column continent. In database parlance, we have a functional dependancy: country \rightarrow continent whereas the key of the table is made of columns country, year. Table gapminder is not in Boyce-Codd Normal Form (BCNF), not even in Third Normal Form (3NF).

Gapminder tibble (extract)

i Question

Extract/filter a subset of rows using dplyr::filter(...)

- All rows concerning a given country
- All rows concerning a year
- All rows concerning a given continuent and a year

g solution

```
# q: in gapminder table extract all raws concerning France
gapminder |>
  filter(country=='France') |>
 head()
# A tibble: 6 x 6
  country continent year lifeExp
                                       pop gdpPercap
  <fct>
          <fct>
                    <int>
                            dbl>
                                     <int>
                                                <dbl>
1 France Europe
                     1952
                             67.4 42459667
                                                7030.
2 France Europe
                             68.9 44310863
                                                8663.
                     1957
3 France Europe
                     1962
                             70.5 47124000
                                               10560.
4 France Europe
                             71.6 49569000
                                               13000.
                     1967
                     1972
                             72.4 51732000
                                               16107.
5 France Europe
6 France Europe
                     1977
                             73.8 53165019
                                               18293.
```

♦ Equality testing is performed using ==, not = (which is used to implement assignment)

Filtering (selection σ from database theory): Picking one year of data

There is simple way to filter rows satisfying some condition. It consists in mimicking indexation in a matrix, leaving the colum index empty, replacing the row index by a condition statement (a logical expression) also called a mask.

```
# q: in gapminder table extract all raws concerning year 2002
gapminder_2002 <- gapminder[gapminder$year==2002, ]</pre>
```

Have a look at gapminder\$year==2002. What is the type/class of this expression?

This is possible in base R and very often convenient.

Nevertheless, this way of performing row filtering does not emphasize the connection between the dataframe and the condition. Any logical vector with the right length could be used as a mask. Moreover, this way of performing filtering is not very functional.

In the parlance of Relational Algebra, filter performs a selection of rows. Relational expression

 $\sigma_{\rm condition}({\rm Table})$

translates to

filter(Table, condition)

where condition is a boolean expression that can be evaluated on each row of Table. In SQL, the relational expression would translate into

SELECT

*
FROM
Table
WHERE
condition

Check Package dplyr docs

The posit cheatsheet on dplyr is an unvaluable resource for table manipulation.

Use dplyr::filter() to perform row filtering

solution # filter(gapminder, year==2002) gapminder |> filter(year==2002) # A tibble: 142 x 6 continent year lifeExp pop gdpPercap country <fct> <dbl> <fct> <int> <int> <dbl> 42.1 25268405 727. 1 Afghanistan Asia 2002 2 Albania Europe 2002 75.7 3508512 4604. 3 Algeria 2002 71.0 31287142 5288. Africa 2773. 4 Angola Africa 2002 41.0 10866106 5 Argentina Americas 2002 74.3 38331121 8798. Oceania 6 Australia 2002 80.4 19546792 30688. 7 Austria Europe 2002 79.0 8148312 32418. 8 Bahrain Asia 2002 74.8 656397 23404. 9 Bangladesh 2002 62.0 135656790 1136. Asia 10 Belgium Europe 2002 78.3 10311970 30486. # i 132 more rows

Data masking

Note that in stating the condition, we simply write year==2002 even though year is not the name of an object in our current session. This is possible because filter() uses data masking, year is meant to denote a column in gapminder. SQL interpreters use the same mechanism.

The ability to use data masking is one of the great strengths of the R programming language.

Static plotting: First attempt

i Question

Define a plot with respect to <code>gapminder_2002</code> along the lines suggested by Rosling's presentation.

solution

```
p <- gapminder_2002 |>
   ggplot()
```

You should define a ggplot object with data layer gapminder_2022 and call this object p for further reuse.

i Question

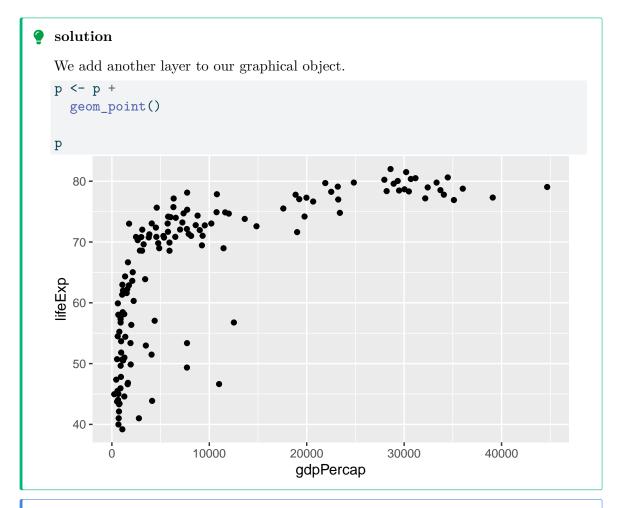
Map variables gdpPercap and lifeExp to axes x and y. Define the axes. In ggplot2 parlance, this is called *aesthetic mapping*. Use aes().

q: Map variables gdpPercap and lifeExp to axes x and y. Define the axes. p <- p + aes(x=gdpPercap, y=lifeExp) p 8070 CX 400 10000 20000 30000 40000 gdpPercap

Use ggplot object p and add a global aesthetic mapping gdpPercap and lifeExp to axes x and y (using + from ggplot2).

i Question

For each row, draw a point at coordinates defined by the mapping. You need to add a geom_layer to your ggplot object, in this case geom_point() will do.



What's up?

We are building a graphical object (a ggplot object) around a data frame (gapminder)

We supply aesthetic mappings (aes()) that can be either global or specifically bound to some geometries (geom_point()) or statistics

The global aesthetic mapping defines which columns (variables) are

- mapped to position (which columns are mapped to axes),
 - possibly mapped to colours, linetypes, shapes, ...

Geometries and Statistics describe the building blocks of graphics

What's missing here?

when comparing to the Gapminder demonstration, we can spot that

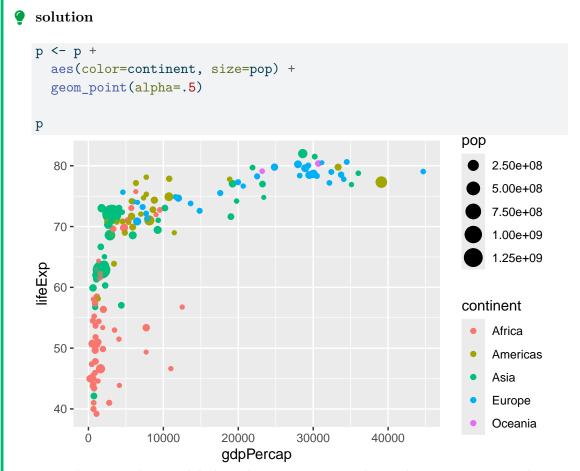
- colors are missing
- bubble sizes are all the same. They should reflect the population size of the country
- titles and legends are missing. This means the graphic object is useless.

We will add other layers to the graphical object to complete the plot

Second attempt: display more information

Question

- Map continent to color (use aes())
- Map pop to bubble size (use aes())
- Make point transparent by tuning alpha (inside geom_point() avoid overplotting)



Note that we only use global aesthetic mappings. This makes sense since we do not need to taylor aesthetics to specific geometries. Indeed we only have one geometry in our graphical object.

solution

In this enrichment of the graphical object, *guides* have been automatically added for two aesthetics: color and size. Those two guides are deemed necessary since the reader has no way to guess the mapping from the five levels of continent to color (the color scale), and the reader needs help to connect population size and bubble size.

ggplot2 provides us with helpers to fine tune guides.

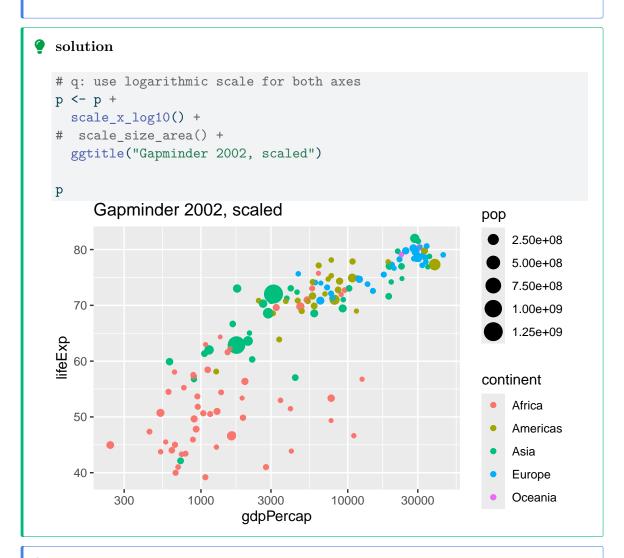
The scalings on the x and y axis do not deserve guides: the ticks along the coordinate axes provide enough information.

Scaling

To pay tribute to Hans Rosling, we need to take care of two scaling issues:

- the gdp per capita axis should be logarithmic scale_x_log10()
- the area of the point should be proportional to the population scale_size_area()

Complete the graphical object accordingly



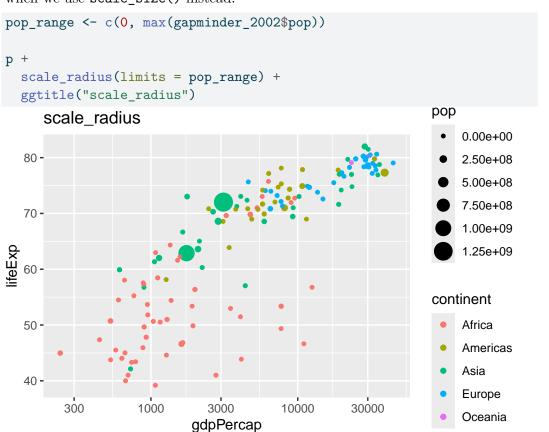
Question

Motivate the proposed scalings.

- Why is it important to use logarithmic scaling for gdp per capita?
- When is it important to use logarithmic scaling on some axis (in other contexts)?
- Why is it important to specify scale_size_area() ?

g solution

To see why using scale_size_area() is important, we can check what happens when we use scale_size() instead.



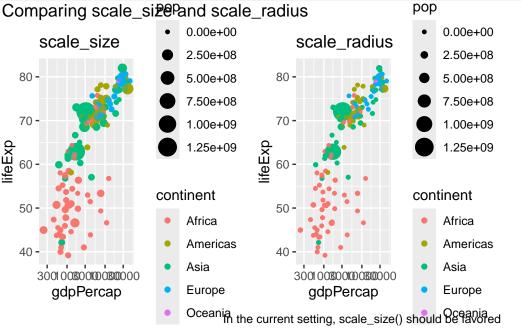
With scale_size_area(), the area of the point is proportional to the value of the variable mapped to size. With scale_size(), the radius of the point is proportional to the value of the variable mapped to size, so the area is proportional to the square of the value of the variable. This tends to exaggerate the differences between the sizes of the points. This is a way of lying with statistics.

solution

We use package patchwork to collect and present several graphical objects.

```
ptchwrk <- (
  p +
  scale_size(limits = pop_range) +
  ggtitle("scale_size")) +
  (p +
  scale_radius(limits = pop_range) +
  ggtitle("scale_radius"))

ptchwrk + plot_annotation(
  title='Comparing scale_size and scale_radius',
  caption='In the current setting, scale_size() should be favored'
)</pre>
```



According to the documentation, scale_size_area() ensures that a value of 0 is mapped to a size of 0. This is not the case with scale_size().

```
ptchwrk <- (
  p +
  scale_size(limits = pop_range) +
  ggtitle("scale_size")) +
  (p +
  scale_size_area() +</pre>
```

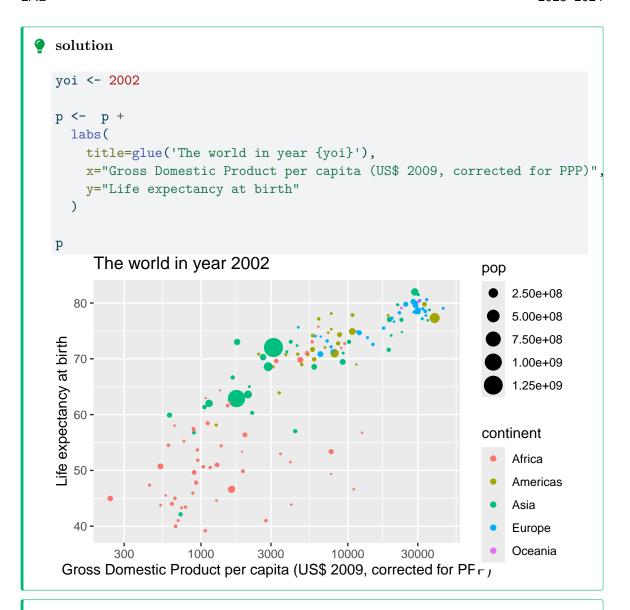
In perspective

i Question

Using copilots completions, we can summarize the construcion of the graphical object in a series of questions.

- # q: Define a plot with respect to table gapminder_2002 along the lines suggested by H
- # q: Map variables gdpPercap and lifeExp to axes x and y. Define the axes.
- # q: For each row, draw a point at coordinates defined by the mapping.
- # q: Map continent to color
- # q: Map pop to bubble size
- # q: Make point transparent by tuning alpha (inside geom_point() avoid overplotting)
- # q: Add a plot title
- # q: Make axes titles explicit and readable
- # q: Use labs(...)
- # q: Use scale_x_log10() and scale_size_area()
- # q: Fine tune the guides: replace pop by Population and titlecase continent
- # q: Use theme_minimal()
- # q: Use scale_color_manual(...) to fine tune the color aesthetic mapping.
- # q: Use facet_zoom() from package ggforce
- # q: Add labels to points. This can be done by aesthetic mapping. Use aes(label=..)

`continent`.



solution

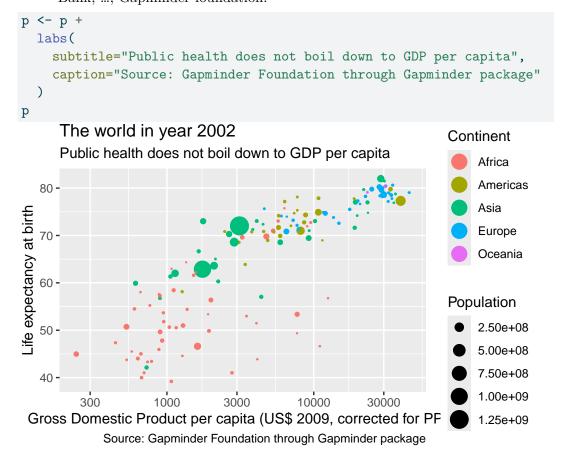
We should also fine tune the guides: replace pop by Population and titlecase continent.

Question

What should be the respective purposes of Title, Subtitle, Caption, ...?

solution

- The title should be explicit and concise. It should summarize the content of the graphic object. Our title here "The world in year 2002" is concise but not explicit enough. The world may signify widely different things. Here, we mean world countries
- The subtitle should provide additional information: "Public health does not boil down to GDP per capita"
- The caption should provide additional information. Here we could explain the meaning of the axes, the color scale, the size scale, ... provided guides are not enough. Here we could spot the source(s) of the data: UNO, WHO, World Bank, ..., Gapminder foundation.



Theming using ggthemes (or not)

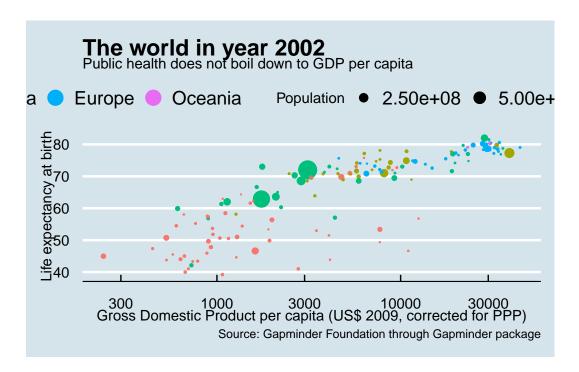
```
stopifnot(
  require("ggthemes")
)
```

A theme defines the look and feel of plots

Within a single document, we should use only one theme

See Getting the theme for a gallery of available themes

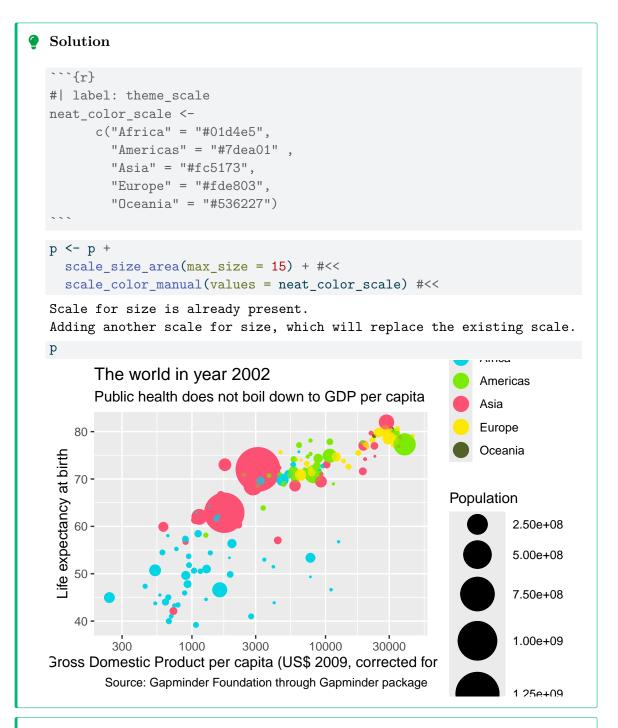
```
p +
theme_economist()
```



Tuning scales

i Question

Use scale_color_manual(...) to fine tune the color aesthetic mapping.



🕊 Tip

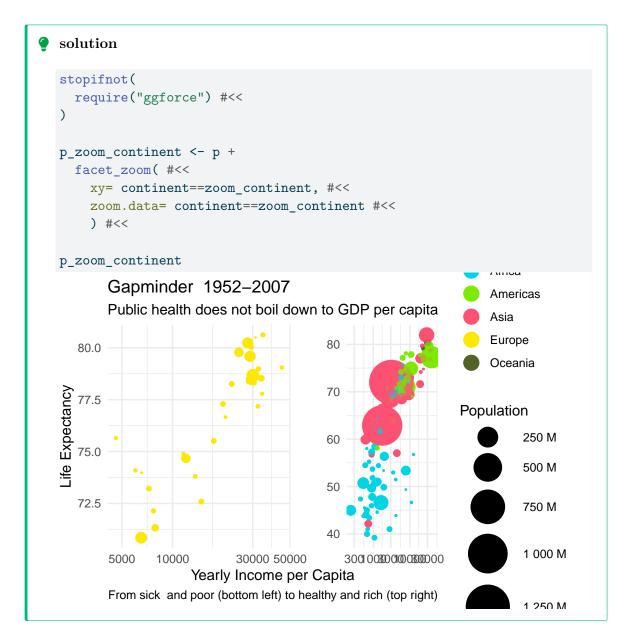
Choosing a color scale is a difficult task viridis is often a good pick.

solution Mimnimalist themes are often a good pick. old_theme <- theme_set(theme_minimal())</pre> p <- p + scale_size_area(max_size = 15, labels= scales::label_number(scale=1/1e6, suffix=" M")) + scale_color_manual(values = neat_color_scale) + labs(title= glue("Gapminder {min(gapminder\$year)}-{max(gapminder\$year)}"), x = "Yearly Income per Capita", y = "Life Expectancy", caption="From sick and poor (bottom left) to healthy and rich (top right)") Scale for size is already present. Adding another scale for size, which will replace the existing scale. Scale for colour is already present. Adding another scale for colour, which will replace the existing scale. p + theme(legend.position = "none") Gapminder 1952-2007 Public health does not boil down to GDP per capita 80 Life Expectancy 40 300 1000 3000 10000 30000 Yearly Income per Capita From sick and poor (bottom left) to healthy and rich (top right)

Zooming on a continent

zoom_continent <- 'Europe' # choose another continent at your convenience</pre>

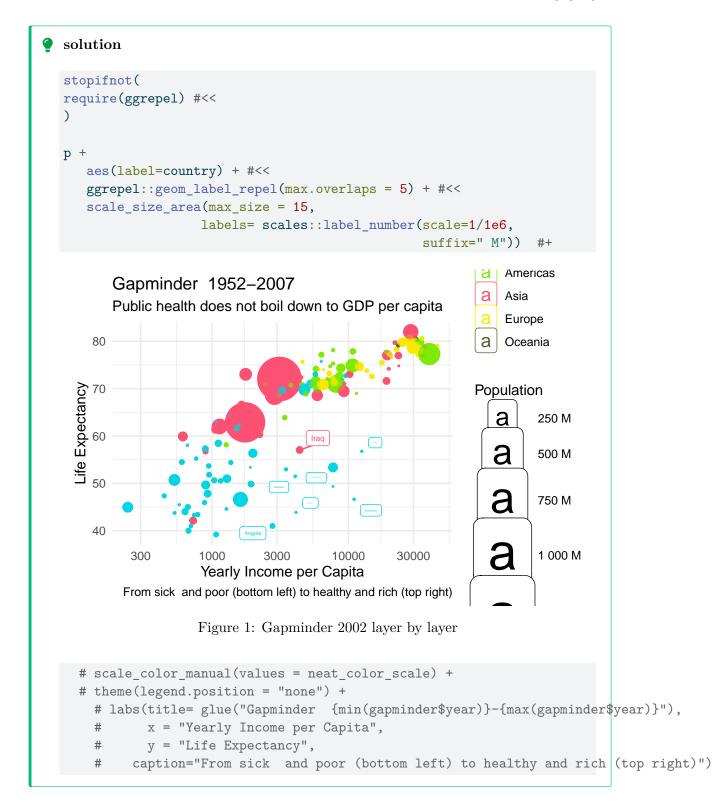
• Use facet_zoom() from package ggforce



Adding labels

i Question

Add labels to points. This can be done by aesthetic mapping. Use aes(label=..) To avoid text cluttering, package ggrepel offers interesting tools.



Facetting

So far we have only presented one year of data (2002)

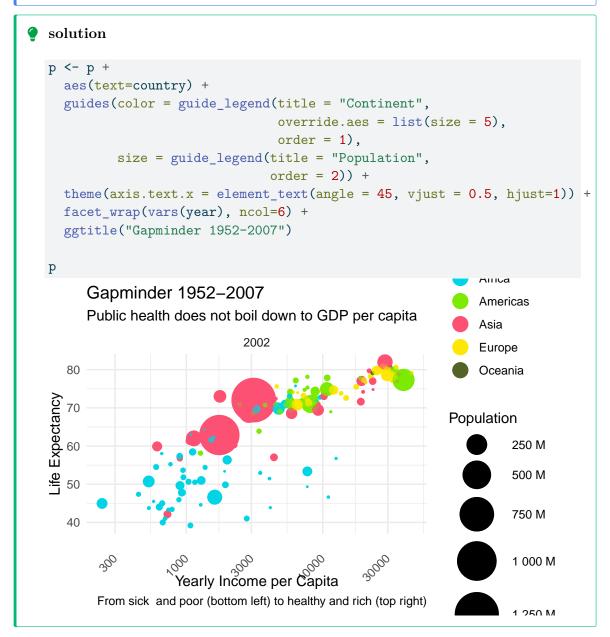
Rosling used an animation to display the flow of time

If we have to deliver a printable report, we cannot rely on animation, but we can rely on facetting

Facets are collections of small plots constructed in the same way on subsets of the data

Question

Add a layer to the graphical object using facet_wrap()



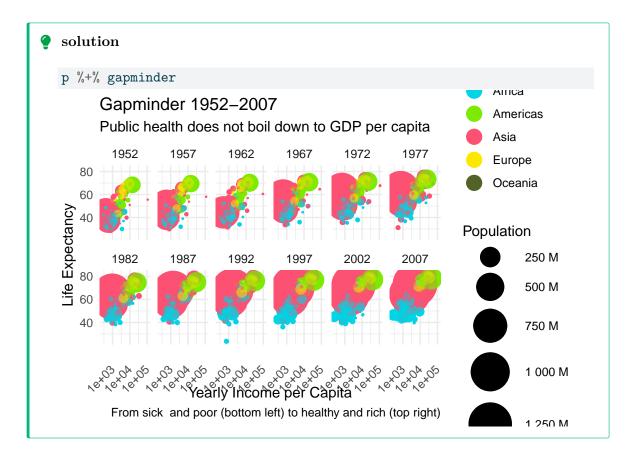
As all rows in <code>gapminder_2002</code> are all related to <code>year 2002</code>, we need to rebuild the graphical object along the same lines (using the same <code>graphical pipeline</code>) but starting from the whole <code>gapminder</code> dataset.

Should we do this using *cut and paste*?

♀ No!!!

Don't Repeat Yoursel (DRY)

Abide to the DRY principle using operator %+%: the ggplot2 object p can be fed with another dataframe and all you need is proper facetting.



Animate for free with plotly

i Question

Use plotly::ggplotly() to create a Rosling like animation. Use frame aesthetics.

```
solution
```{r}
#| label: animate
#| eval: !expr knitr::is_html_output()
#| code-annotations: hover
q <- filter(gapminder, FALSE) |>
 ggplot() +
 aes(x = gdpPercap) +
 aes(y = lifeExp) +
 aes(size = pop) +
 aes(text = country) +
 #
 aes(fill = continent) +
 # aes(frame = year) +
 geom_point(alpha=.5, colour='black') +
 scale_x_log10() +
 scale_size_area(max_size = 15,
 labels= scales::label_number(scale=1/1e6,
 suffix=" M")) +
 scale_fill_manual(values = neat_color_scale) +
 theme(legend.position = "none") +
 labs(title= glue("Gapminder {min(gapminder$year)}-{max(gapminder$year)}"),
 x = "Yearly Income per Capita",
 y = "Life Expectancy",
 caption="From sick and poor (bottom left) to healthy and rich (top right)")
(q %+% gapminder) |>
 plotly::ggplotly(height = 500, width=750)
 1. text will be used while hovering
 2. frame is used by plotly to drive the animation. One frame per year
```

## solution ```{r} #| eval: !expr knitr::is\_html\_output() (p %+% gapminder + facet\_null() + aes(frame=year)) |> plotly::ggplotly(height = 500, width=750) ```

### Suggestions

Think about ways to visualize specific aspects of the gapminder data.

- How could you overlay the world in 1952 and 2007?
- How could you visualize the evolution of life expectancy and population across the different countries?

- Visualize the evolution of former colonies and their colonizers.
- Visualize the evolution of countries from the former Soviet Union, Warsaw Pact, and Yugoslavia.
- Visualize the evolution of countries from the former British Empire.

### More material

Visit Data visualization using ggplot2 and its extensions, UseR 2021 Tutorial

Read Visualization in R for Data Science