Exploring gapminder

Practice notebook for DSC 205 Spring 2022

1 README

In this extensive notebook, you're learning how relatively simple ggplot2 and dplyr code can create interesting plots - always provided the underlying dataset is interesting.

For an overview of both of these "Tidyverse" packages, see the online reference pages for ggplot2 and dplyr[1].

The dataset is gapminder adapted from the original dataset created by Hans Rosling[2].

This workbook is adapted from chapter 10 "Data visualization in practice" of the textbook by Irizarry (2021), which is also the basis of an introductory data science course at Harvard U.

2 Getting the data

- We need to install some things first. Don't just click through these commands but make absolutely sure that you really understand every step and every command.
- If you really understand a command in R, you know what the alternative paths are. There is always at least one alternative way of getting what you want in R (this is true for all good programming languages).
- `[X]`

  To be able to run the installation inside Org-mode (and see the output here), I put one line in my `.Rprofile`. This file will be fetched by Emacs + ESS, so it needs to be placed in the Emacs HOME directory. You can also run the code right now.

```
options(repos=c("https://cloud.r-project.org/"))
```

- `[X]`

  Check where your computer gets its CRAN packages from. If you put the code 1 into `.Rprofile`, you should get the nearest mirror.

```
getOption("repos")
```

```
[1] "https://cloud.r-project.org/"
```

- `[X]`

  We need a couple of packages and a dataset. Put the code in 1 below and run it:

  1. Install the `tidyverse` and the `dslabs` packages
  2. Load the `tidyverse` and the `dslabs` packages
  3. Load the `gapminder` dataset from the `dslabs` package

```
## The install commands are commented out to save time
##      install.packages("tidyverse")
##      install.packages("dslabs")
library(tidyverse)
library(dslabs)
data(gapminder)
```

  - I set the output of 1 to `silent`. If you want to see what R did, check the R console session in the `*R*` buffer.
  - `[X]` The original Gapminder data set is much larger than the subset at CRAN or the dslabs dataset.
  - `[X]` Check `??gapminder` in the R console to see how many gapminder data sets you have available for loading[3].
- `[X]`

  If you looked at the output of the `data` loading command, you may have noticed that the data display is messed up with control characters. If this happens with a familiar command (e.g. `str`) it's to with the "tibble" format that "Tidyverse" data frames come in. To get rid of the extra characters, run this command (or better put it into your .Rprofile:

```
options(crayon.enabled = FALSE)
```

  Tibbles are limited to 10 rows by default.

- `[X]` If 1 ran successfully, I recommend you go back to the code block and comment the `install.packages` commands. Otherwise, your workbook may attempt to re-install them.
- `[X]`

  You don't need to re-install packages unless a) you upgraded to a different version of R, or b) the package was upgraded.

  You can update your installed packages with one command (Wasser, 2021). You should do this on the console - answer "Yes" always.

  Check the (local) `update.packages()` reference for additional options and commands. You can run the commmand below in the R console.

```
update.packages()
```

2.1 Section summary

- Knowing alternative paths in R is not a waste
- Installing and loading R packages
- Updating R packages
- Tibble format for data frames
- Fixing an `Internet.dll` issue on Windows

3 Checking and getting to know the data

- `[X]`

  Check the structure of `gapminder`. You see `numeric` and `factor` vectors.

```
str(gapminder)
```

```
'data.frame':   10545 obs. of  9 variables:
 $ country         : Factor w/ 185 levels "Albania","Algeria",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ year            : int  1960 1960 1960 1960 1960 1960 1960 1960 1960 1960 ...
 $ infant_mortality: num  115.4 148.2 208 NA 59.9 ...
 $ life_expectancy : num  62.9 47.5 36 63 65.4 ...
 $ fertility       : num  6.19 7.65 7.32 4.43 3.11 4.55 4.82 3.45 2.7 5.57 ...
 $ population      : num  1636054 11124892 5270844 54681 20619075 ...
 $ gdp             : num  NA 1.38e+10 NA NA 1.08e+11 ...
 $ continent       : Factor w/ 5 levels "Africa","Americas",..: 4 1 1 2 2 3 2 5 4 3 ...
 $ region          : Factor w/ 22 levels "Australia and New Zealand",..: 19 11 10 2 15 21
```

- [X] Check the local help for the dslabs `gapminder` dataset for the meaning of the variables (do this from the R console).
- [X]

  Make a copy of `gapminder` as `gm` so you won't have to do so much typing (and also to protect the original data). Check that they're identical!

```
gm <- gapminder
identical(gm, gapminder)
```

```
[1] TRUE
```

- [X]

  Print the first 10 lines of the first four columns, and then the first 10 lines of the next four columns of the data frame.

```
head(gm[1:4], 10)
paste("")
head(gm[5:9], 10)
```

```
              country year infant_mortality life_expectancy
1             Albania 1960           115.40           62.87
2             Algeria 1960           148.20           47.50
3              Angola 1960           208.00           35.98
4  Antigua and Barbuda 1960               NA           62.97
5           Argentina 1960            59.87           65.39
6             Armenia 1960               NA           66.86
7               Aruba 1960               NA           65.66
8           Australia 1960            20.30           70.87
9             Austria 1960            37.30           68.75
10         Azerbaijan 1960               NA           61.33
[1] ""
   fertility population          gdp continent                  region
1       6.19    1636054           NA    Europe          Southern Europe
2       7.65   11124892  13828152297    Africa          Northern Africa
3       7.32    5270844           NA    Africa             Middle Africa
```

```
4       4.43      54681           NA  Americas                Caribbean
5       3.11   20619075 108322326649  Americas             South America
6       4.55    1867396           NA      Asia              Western Asia
7       4.82      54208           NA  Americas                Caribbean
8       3.45   10292328  96677859364   Oceania Australia and New Zealand
9       2.70    7065525  52392699681    Europe            Western Europe
10      5.57    3897889           NA      Asia              Western Asia
```

- [X]

  This isn't a Nintendo Gameboy. You've got screen space! Reset the number of columns printed on a line by resetting the attribute `width` of `options` to the value 140 (the default is 80, the maximum value is 10,000).

  To test the new setting, print the top 10 lines of the whole dataframe.

```
options(width=140)
head(gm, 10)
```

```
               country year infant_mortality life_expectancy fertility population
1              Albania 1960           115.40           62.87      6.19    1636054
2              Algeria 1960           148.20           47.50      7.65   11124892  1382815
3               Angola 1960           208.00           35.98      7.32    5270844
4  Antigua and Barbuda 1960               NA           62.97      4.43      54681
5            Argentina 1960            59.87           65.39      3.11   20619075 10832232
6              Armenia 1960               NA           66.86      4.55    1867396
7                Aruba 1960               NA           65.66      4.82      54208
8            Australia 1960            20.30           70.87      3.45   10292328  9667785
9              Austria 1960            37.30           68.75      2.70    7065525  5239269
10          Azerbaijan 1960               NA           61.33      5.57    3897889
```

- [X]

  Print the dataframe as a "tibble". To do this, run the function `as_tibble` with `gapminder` as the argument.

  In Emacs, you will see the control characters obscuring the display. To view it as it was meant to look like, switch to the R console in the `*R*` buffer and run the command there.

```
options(crayon.enabled = FALSE)
as_tibble(gm)
```

```
# A tibble: 10,545 x 9
   country              year infant_mortality life_expectancy fertility population
   <
              <
           <
         <
        <
    <
     <
       <
<
    <
```

```
 1 Albania              1960          115.          62.9      6.19    1636054
 2 Algeria              1960          148.          47.5      7.65   11124892  138281
 3 Angola               1960          208           36.0      7.32    5270844
 4 Antigua and Barbuda  1960           NA           63.0      4.43      54681
 5 Argentina            1960           59.9        65.4      3.11   20619075 1083223
 6 Armenia              1960           NA           66.9      4.55    1867396
 7 Aruba                1960           NA           65.7      4.82      54208
 8 Australia            1960           20.3         70.9      3.45   10292328  966778
 9 Austria              1960           37.3         68.8      2.7     7065525  523926
10 Azerbaijan           1960           NA           61.3      5.57    3897889
# ... with 10,535 more rows
```

The figure 1 shows what you should see. As you can see, the format is condensed to fit the 80-char default display setting. `NA` values are highlighted in color, data types are shown in a separate row, and 10 lines are shown by default only.

None of these are either essential or even add much to our understanding of the data (beyond the basic `str` command). At the same time, an extra dependency (character layout) is introduced.

```
> as_tibble(gp)
# A tibble: 10,545 x 9
   country   year infant_mortality life_expectancy fertility population      gdp
   <fct>    <int>           <dbl>           <dbl>     <dbl>      <dbl>    <dbl>
 1 Albania   1960           115.             62.9      6.19    1636054 NA
 2 Algeria   1960           148.             47.5      7.65   11124892  1.38e10
 3 Angola    1960           208              36.0      7.32    5270844 NA
 4 Antigua~  1960            NA              63.0      4.43      54681 NA
 5 Argenti~  1960            59.9            65.4      3.11   20619075  1.08e11
 6 Armenia   1960            NA              66.9      4.55    1867396 NA
 7 Aruba     1960            NA              65.7      4.82      54208 NA
 8 Austral~  1960            20.3            70.9      3.45   10292328  9.67e10
 9 Austria   1960            37.3            68.8      2.7     7065525  5.24e10
10 Azerbai~  1960            NA              61.3      5.57    3897889 NA
# ... with 10,535 more rows, and 2 more variables: continent <fct>,
#   region <fct>
```

Figure 1: Gapminder as tibble

- The `dplyr` package is a package for data frame manipulation. We're going to really use it in a moment. `dplyr` makes ample use of the "piping" operator from another package, `magrittr`, `%>%` (Bache, 2014)[4]. Since last year, Base R also has its own pipeline operator, which is a little less obscure looking: `|>`. [5]
- Emacs tip: to follow the footnotes, press `C-c C-o` on the footnote, to get back here, press `C-c C-o` again, this time in the footnote.
- `[X]`

You don't see the potential power of pipes if you only use one. It becomes a handy tool (to some, not to me[6]) when you build a "pipeline" of several commands as we will soon see.

In 1, "pipe" the data frame into the `as_tibble` function by putting it on the left, and the function on the right of the operator. Do this first for the `magrittr`, then for the Base R operator. (There are some spurious `>` characters in the output - you can just delete the respective rows).

```
gm %>% as_tibble()
gm |> as_tibble()
```

```
# A tibble: 10,545 x 9
   country              year infant_mortality life_expectancy fertility population
   <
            <
          <
         <
       <
        <
          <
<
    <

 1 Albania              1960            115.             62.9      6.19    1636054
 2 Algeria              1960            148.             47.5      7.65   11124892  138281
 3 Angola               1960            208              36.0      7.32    5270844
 4 Antigua and Barbuda  1960             NA              63.0      4.43      54681
 5 Argentina            1960             59.9            65.4      3.11   20619075 1083223
 6 Armenia              1960             NA              66.9      4.55    1867396
 7 Aruba                1960             NA              65.7      4.82      54208
 8 Australia            1960             20.3            70.9      3.45   10292328  966778
 9 Austria              1960             37.3            68.8      2.7     7065525  523926
10 Azerbaijan           1960             NA              61.3      5.57    3897889
# ... with 10,535 more rows
# A tibble: 10,545 x 9
   country              year infant_mortality life_expectancy fertility population
   <
            <
          <
         <
       <
        <
          <
<
    <

 1 Albania              1960            115.             62.9      6.19    1636054
 2 Algeria              1960            148.             47.5      7.65   11124892  138281
 3 Angola               1960            208              36.0      7.32    5270844
 4 Antigua and Barbuda  1960             NA              63.0      4.43      54681
 5 Argentina            1960             59.9            65.4      3.11   20619075 1083223
 6 Armenia              1960             NA              66.9      4.55    1867396
 7 Aruba                1960             NA              65.7      4.82      54208
 8 Australia            1960             20.3            70.9      3.45   10292328  966778
 9 Austria              1960             37.3            68.8      2.7     7065525  523926
10 Azerbaijan           1960             NA              61.3      5.57    3897889
# ... with 10,535 more rows
```

3.1 Section summary

- Reviewing structure checking commands
- Changing the display width option
- Printing a data frame as a tibble
- Pipes to pass data to functions

- Pipeline concept

4 Filtering the data

4.1 Rosling's survey

- [X]

  This is a famous survey question by Rosling at the start of his TED talks: for each of the six pairs of countries below,

  1. which country do you think had the highest child mortality rates in 2015? (Measured in infant deaths per 1000)
  2. Which pairs do you think are the most similar?

  Think about this, then fill in the table 1 according to your opinion (IM = Infant Mortality per 1000). Put a cross next to the country that you think has the highter infant mortality.

COUNTRY	IM	COUNTRY	IM
Sri Lanka	8.4	Turkey	11.6
Poland	4.5	South Korea	2.9
Malaysia	6.0	Russia	8.2
Pakistan	65.8	Vietnam	17.3
Thailand	10.5	South Africa	33.6

- [X]

  Let's run the numbers in the code blocks below, then put the results in the table 1 and interpret them.

  The code in 1 shows

  - two pipes `%>%`
  - the function `dplyr::filter` to filter rows for year and countries
  - the operator `%in%` to identify if an element is in a vector
  - the function `dplyr::select` to select two column vectors

```
gm %>%
  filter(year == 2015 & country %in% c("Sri Lanka", "Turkey")) %>%
  select(country, infant_mortality)
```

```
    country infant_mortality
1 Sri Lanka              8.4
2    Turkey             11.6
```

- [X]

  Put in the code for the other four pairs. Now, don't you wish you'd have written a function first? (See below)

```
gf2(2015,c("Poland", "SouthKorea"))
```

```
  country infant_mortality
1  Poland              4.5
```

```
gf2(2015,c("Malaysia", "Russia"))
```

```
   country infant_mortality
1 Malaysia              6.0
2   Russia              8.2
```

```
gf2(2015,c("Pakistan", "Vietnam"))
```

```
   country infant_mortality
1 Pakistan             65.8
2  Vietnam             17.3
```

```
gf2(2015,c("Thailand", "South Africa"))
```

```
       country infant_mortality
1 South Africa             33.6
2     Thailand             10.5
```

- [X] Extra credit: Write a function that achieves this and test it with the five pairs in table 1 - test it and send me the solution via email.
- [X] Solution

1. start with one argument in the `filter` function

```
library(dplyr)
library(dslabs)
data(gapminder)

gf1 <- function(year_) {
  gapminder %>%
    filter(year == year_ &
           country %in% c("Sri Lanka", "Turkey")) %>%
    select(country, infant_mortality)
}
gf1(2015)
```

```
    country infant_mortality
1 Sri Lanka              8.4
2    Turkey             11.6
```

2. add the other argument in the `filter` function

```
## function definition
gf2 <- function(year_, country_v) {
  gapminder %>%
    filter(year == year_ & country %in% country_v) %>%
    select(country, infant_mortality)
}
## function call
gf2(year_ = 2015,
    country_v = c("Sri Lanka", "Turkey"))
```

```
    country infant_mortality
1 Sri Lanka              8.4
2    Turkey             11.6
```

4.2 Interpretation

- [ ]

  Interpreting table 1: apparently, less than 50% of surveyed would hit on these numbers, even if they were generally educated. Most would automatically assume that non-European countries have higher infant mortality rates.

  Rosling took this as a sign that people were misinformed rather than ignorant. Can you think of other more concrete reasons why the survey answers and the data answers differed so wildly?[7]

- [ ]

  List potential reasons (other than misinformation) for the survey results in opposition to the data findings in 1:

  1. Quality: the raw data may be wrong. Data quality depends on reporting and monitoring quality processes, which can be assumed to differ wildly between different countries.
  2. Sample: the data were all taken from one year, 2015. What if this was a year of infant mortality outliers for some countries? (For example as a result of a pandemic or war.)
  3. Context: while the data are what they are, the selection of infant mortality is highly emotional (compared to, say, the average loss of hair in these countries). Human beings respond sensitively to context, and are prone to exaggeration when over-stimulated, or to the opposite, when under-stimulated.

- Rosling's central point in his TED talks and in his books was: our 1960s view of the world is outdated. The old dichotomy of rich and poor countries no longer holds. The response to table 1 in contrast to the findings illustrates this one point quite well. But as the analysis shows, the whole truth may well be more multi-layered.

4.3 Code analysis and examples

4.3.1 Code

```
gm %>%
  filter(year == 2015 & country %in% c("Sri Lanka", "Turkey")) %>%
  select(country, infant_mortality)
```

There is a lot going on in this code already. Let's analyze it element by element:

- pipes `%>%`
- `filter`
- `select`
- `%in%`

4.3.2 gm %>%

- The first line calls the data frame and pipes it into the function on the right hand side of the pipe operator.
- `[ ]`

  Pipe `gm` into the function `is.data.frame`. The answer should be `TRUE`.

```
gm %>% is.data.frame()
```

```
[1] TRUE
```

- `[ ]`

  How would you find out how many countries there are in `gapminder` by stringing functions together using the pipe?

```
gm$country %>% levels() %>% length()
```

```
[1] 185
```

  This is equivalent to the expression without pipe:

```
length(levels(gm$country))
```

```
[1] 185
```

- `[ ]`

  Count the number of missing `infant_mortality` values.

```
gm %>% select(infant_mortality) %>% is.na() %>% sum()
```

```
[1] 1453
```

4.3.3 dplyr::filter

- `filter` returns the rows for which its argument is `TRUE`. This is like the `WHERE` operator in SQL.

- In the code extract 1, the function argument is true for all rows of the data frame for whom both conditions are true:
  - the `year` is `2015`, AND
  - the `country` is either Sri Lanka or Turkey

```
filter(year == 2015 & country %in% c("Sri Lanka", "Turkey"))
```

4.3.4 dplyr::select

- `select` selects column vectors. This is like `SELECT` in SQL.

  In the code block below, write a command with **two** pipes starting with the data set `gm`: first, `select` the variable `region`, and then print the first `5` lines of the column.

```
gm %>% select(region) %>% head(5)
```

```
           region
1 Southern Europe
2 Northern Africa
3   Middle Africa
4       Caribbean
5   South America
```

4.4 The %in% operator

- `%in%` is a value matching operator. It is itself a function that runs the `match` function: it returns a logical vector indicating if there is a match or not for its left operand.

  Check out `example("%in%")` for a demo based on the examples from the help. A nice touch: `%in%` never returns `NA`.

```
"%in%" <- function(x, table) match(x, table, nomatch=0) > 0
```

4.5 Section summary

- Data findings need to be critically viewed just like people's opinions. Important aspects include: data quality, samples, and context. Though the data may not change, their interpretation may.
- `dplyr`'s commands - like `select` for column vector selection, or `%in%` for pattern matching- were designed with SQL in mind.
- The pipe operator can be used to string commands together as a pipeline where the output of the last command becomes the input of the next. In R, both `|>` (base R) and `%>%` (`dplyr`) are valid pipe operators.

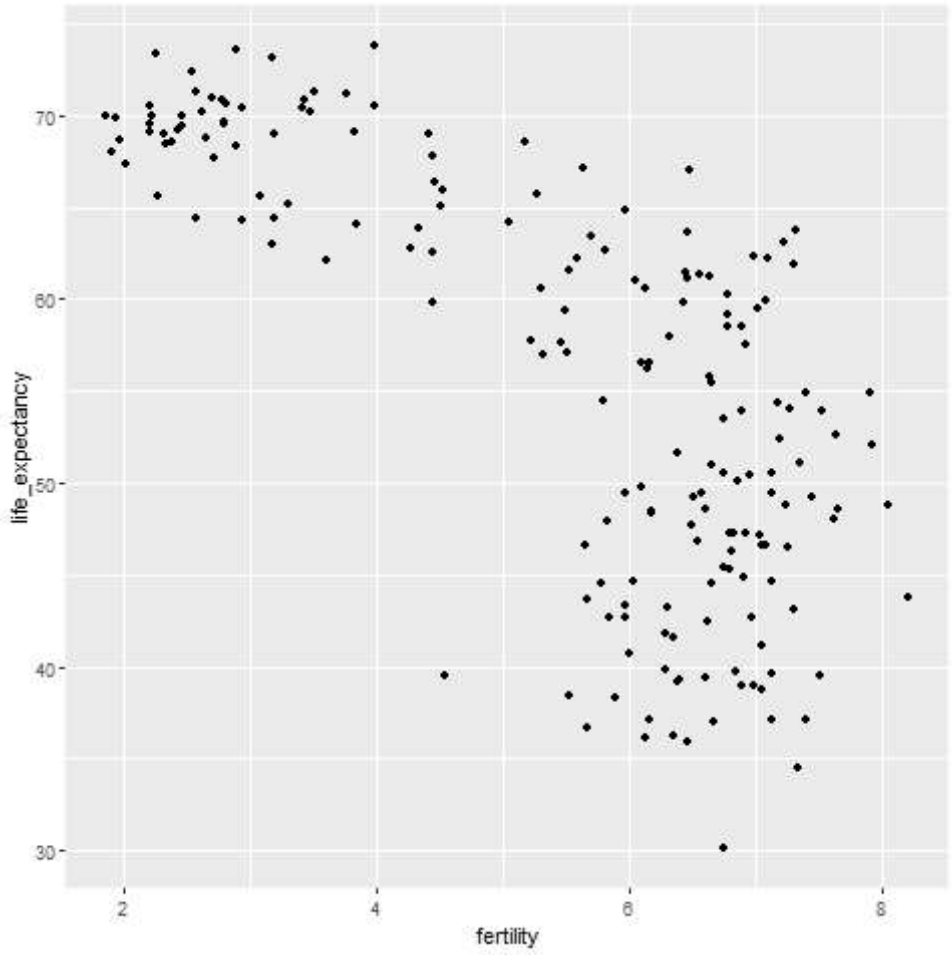
5 Scatterplots

- `[X]`

  Use ggplot to plot `life_expectancy` vs. `fertility` rate (average number of children per woman), for data from 50 years ago - `filter` the year `1962`).

Tip: you need to start with `filter` and plot the result.

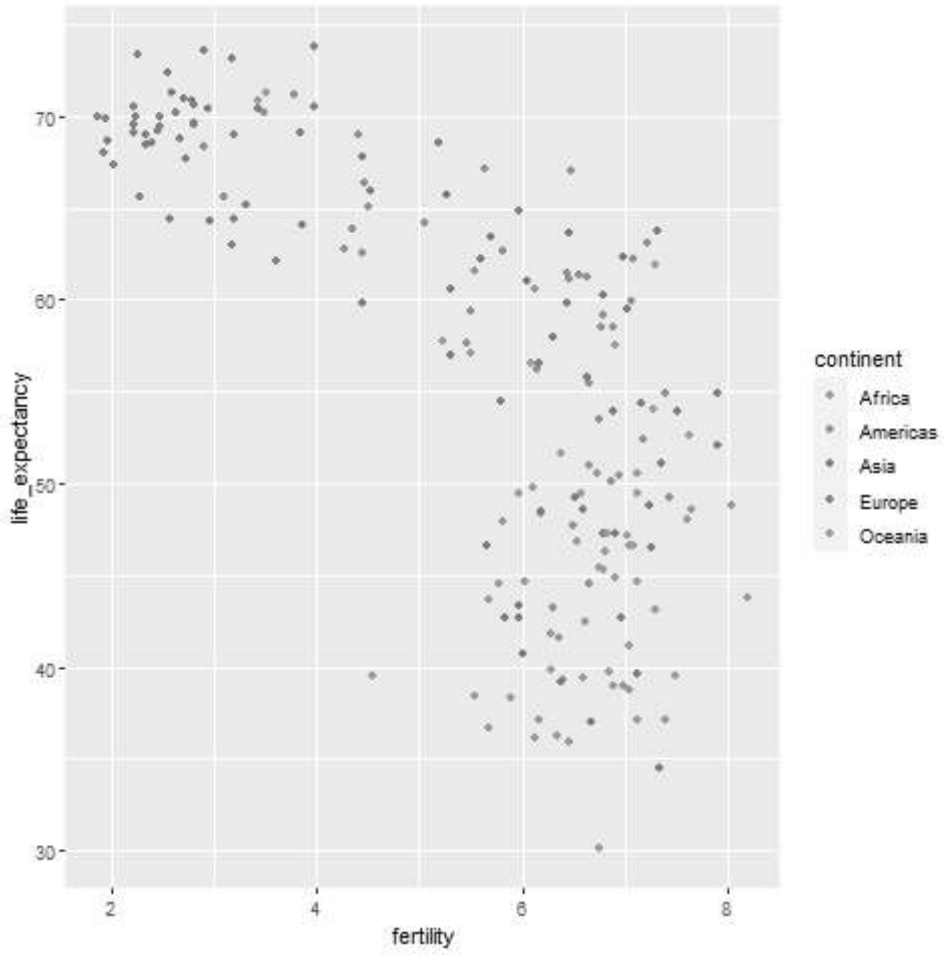
```
gm %>% filter(year == 1962) %>%
  ggplot(aes(fertility, life_expectancy)) +
  geom_point()
```



- There are two clusters:
    1. life expectancy around 70 years and 3 or fewer children per family
    2. life expectancy lower than 65 years and more than 5 children per family
- [X]

To confirm that these countries are from the regions we expect, add `color=continent` to the aesthetic mapping.

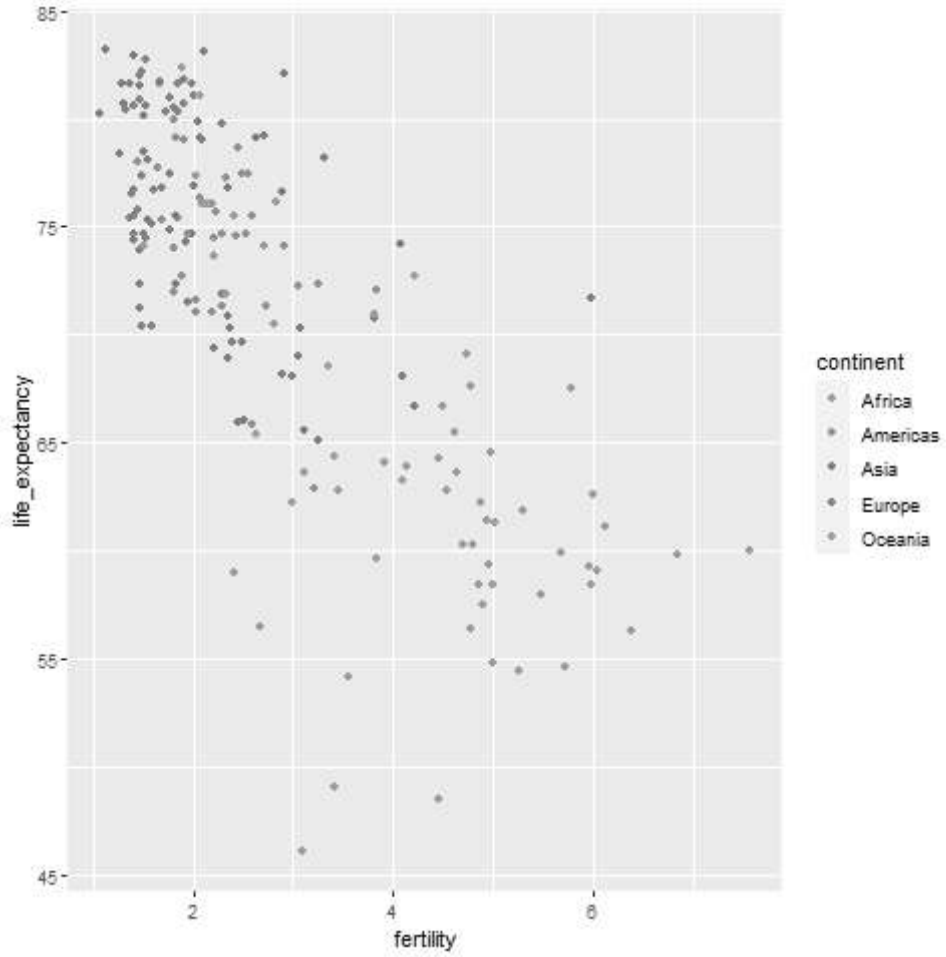
```
gm %>% filter(year == 1962) %>%
  ggplot(aes(fertility,
             life_expectancy,
             color = continent )) +
  geom_point()
```

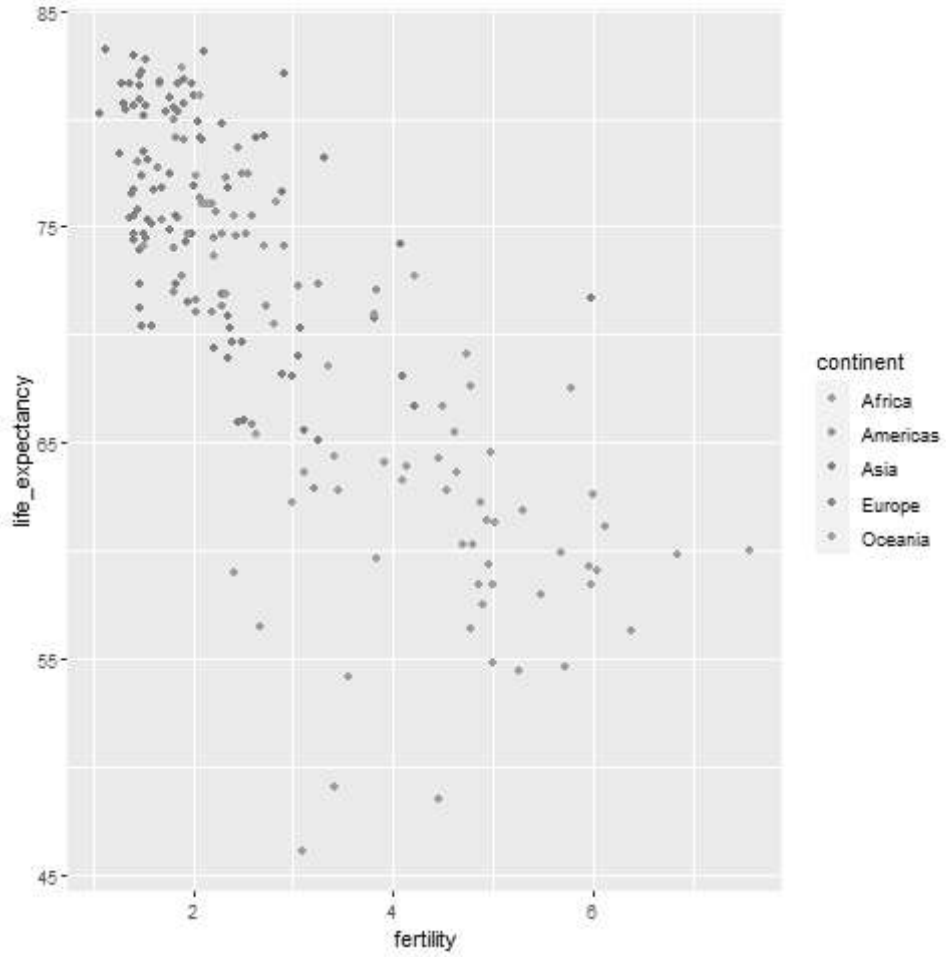


- In 1962, the view of the West vs. developing world is clearly visible. Let's check on the situation 50 years later.
- `[ ]`

  Change the `year` to `2012`.

```
gm %>% filter(year == 2012) %>%
  ggplot(aes(fertility,
             life_expectancy,
             color = continent )) +
  geom_point()
```


85
75
65
55
45
life_expectancy
2
4
6
fertility
continent
Africa
Americas
Asia
Europe
Oceania


85
75
65
55
45
life_expectancy
2
4
6
fertility
continent
Africa
Americas
Asia
Europe
Oceania

]]

6 Faceting

- It would be nice to see plots of 1962 and 2012 side by side.
- This is achieved by adding a layer `facet_grid`, which automatically separates the plots.

6.1 `facet_grid`

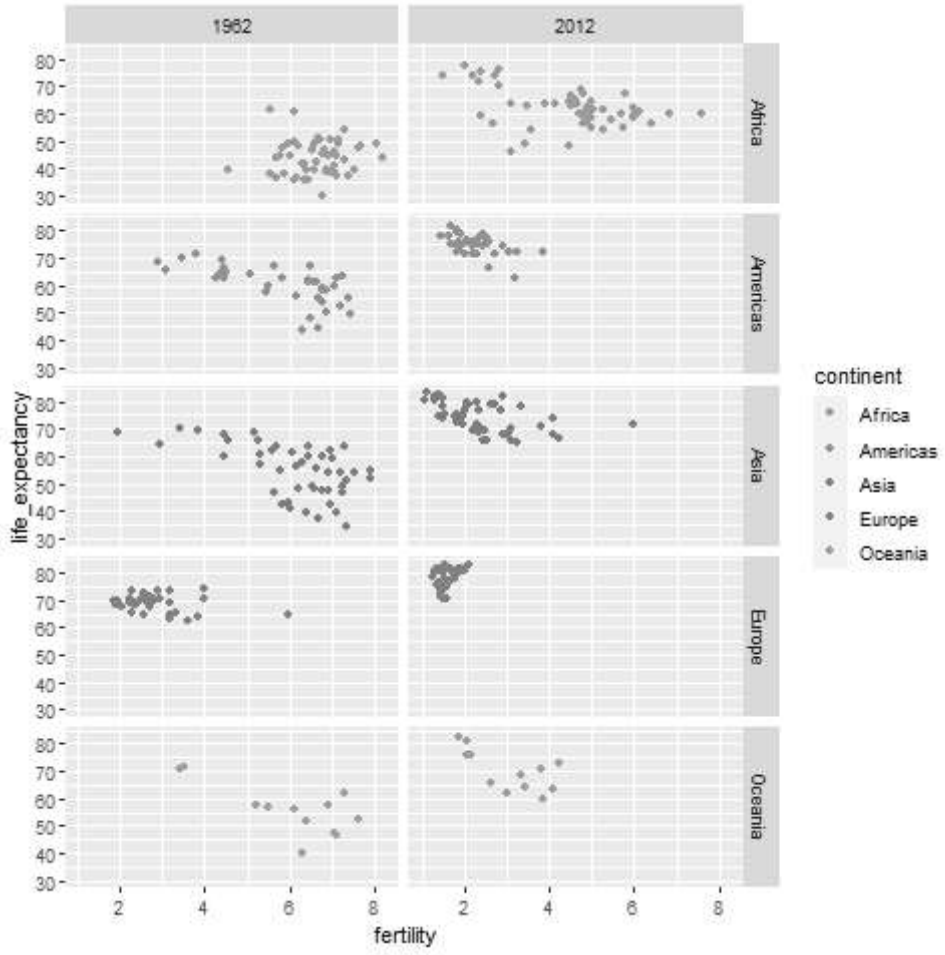
- `[X]`

  Use the `%in%` function to `filter` the years `1962` and `2012`, and add the layer `facet_grid(continent ~ year)` to the pipe.

  Tip: `%in%` matches elements on its right side.

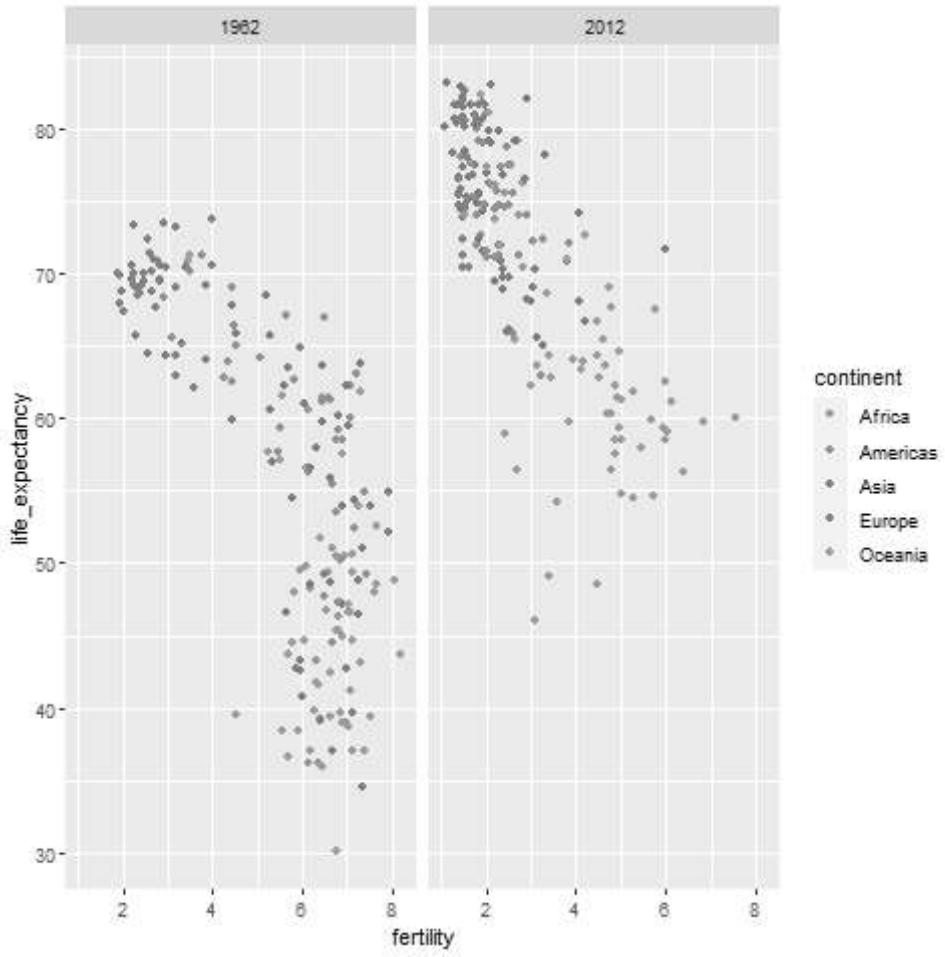
  #+name facet1

```
gm %>% filter(year %in% c(1962,2012)) %>%
  ggplot(aes(fertility,
             life_expectancy,
             color = continent )) +
  geom_point() +
  facet_grid(continent ~ year) ## y = continent, x = year
```



- You should see a plot for each continent/year pair. This is more than we want - we only need one faceting variable. Replace `continent` in the argument of `facet_grid` by a period `.` , and run the code again.

```
gm %>% filter(year %in% c(1962,2012)) %>%
  ggplot(aes(fertility,
             life_expectancy,
             color = continent )) +
  geom_point() +
  facet_grid( . ~ year)
```



- This plot shows that many countries have moved from the developing world cluster (low life expectancy and high fertility) to the western world one (high life expectancy and low fertility).
- The effect is more easily visible if you change the y axis of the plot. Perhaps you can find out yourself how to do that so that it looks more like this[8]:

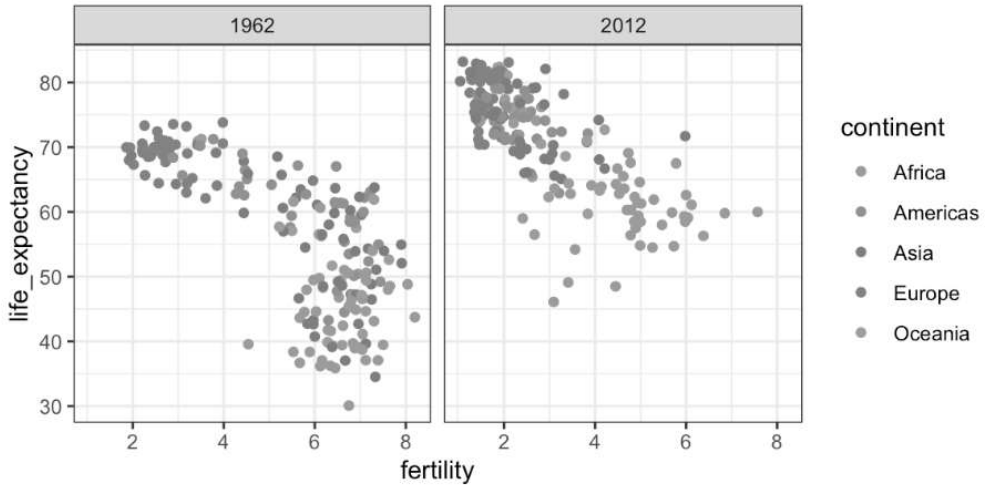


Figure 6: life expectancy vs fertility (Source: Irizarry 2012).

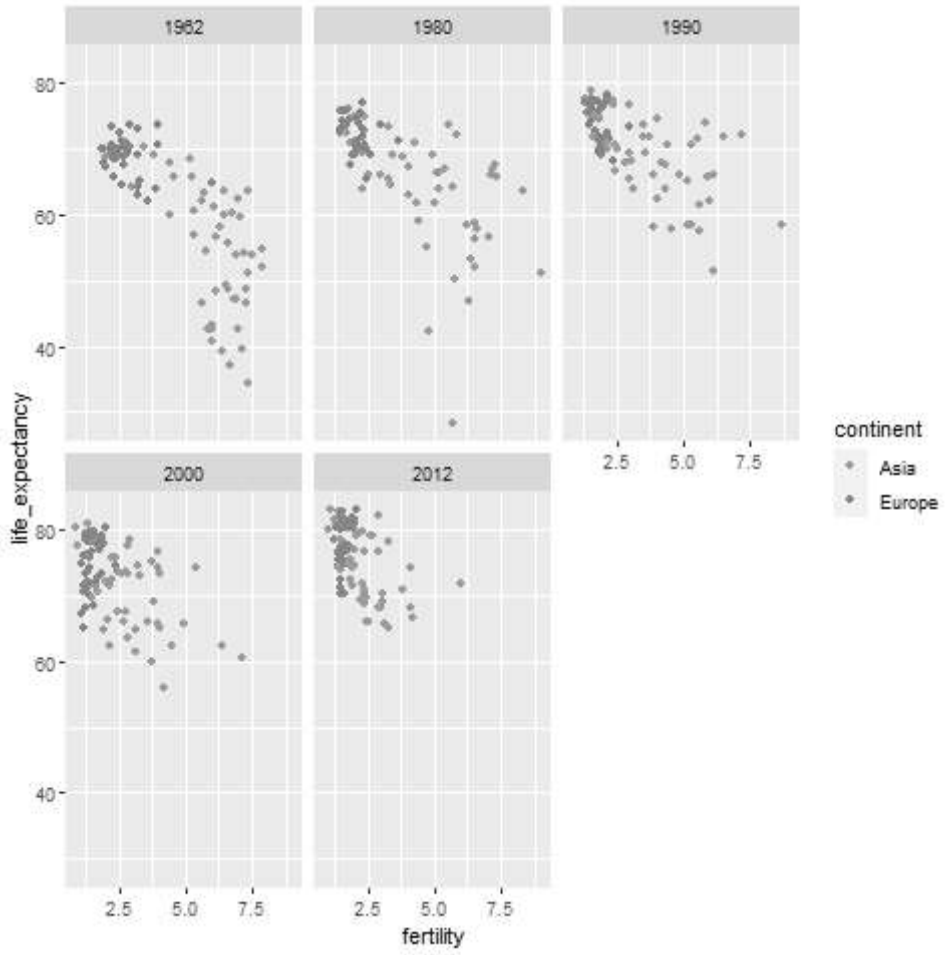
6.2 `facet_wrap`

- To explore how the transformation of a distribution evolved, we look at the plots for several years. This is also a suitable starting point for animation.
- `[ ]`

  In the code block below, construct a pipe that displays `fertility` vs. `life_expectancy` for several years on two continents.

  1. Define a vector for the variable `years` that contains the years `1962`, `1980`, `1990`, `2000`, and `2012`, and a vector `continents` for `Europe` and `Asia`. Now, begin building a pipe:
  2. Filter `years` and `continent` from the dataset `gapminder`
  3. Pipe the result into `ggplot`, plot points, and add the geometry `facet_wrap( ~ year)`.

```
years <- c(1962, 1980, 1990, 2000, 2012)
continents <- c("Europe", "Asia")
gapminder %>%
  filter(year %in% years & continent %in% continents) %>%
  ggplot( aes(fertility, life_expectancy, col = continent)) +
  geom_point() +
  facet_wrap(~year)
```

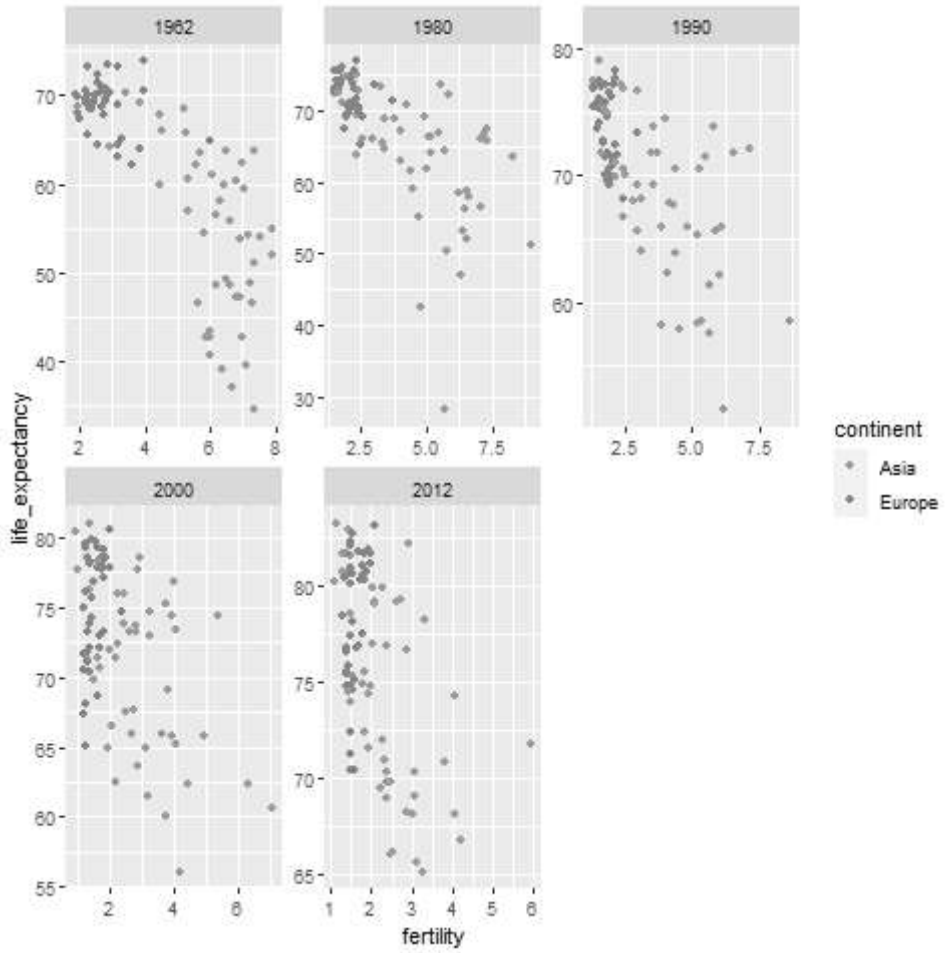


6.3 Fix scales

- Axis range is an important visualization parameter. Without using `facet`, the range is determined by the data shown in the plot.
- When using `facet`, the range is determined by the data shown in *all* plots and kept fixed across plots.
- If you adjust the scales individually by facet plot, you need to pay extra attention to see what has actually changed (or not).
- [ ]

  Add `scales = "free"` to the argument of `facet_wrap` in the previous code block.

```
years <- c(1962, 1980, 1990, 2000, 2012)
continents <- c("Europe", "Asia")
gapminder %>%
  filter(year %in% years & continent %in% continents) %>%
  ggplot( aes(fertility, life_expectancy, col = continent)) +
  geom_point() +
  facet_wrap(~year, scales = "free")
```



7 References

- Bache SM (Nov 2014). Introducing magrittr [vignette]. URL: cran.r-project.org.
- Berggren C (16 Nov 2018). The One-Sided Worldview of Hans Rosling [article]. URL: quillette.com.
- Irizarry R (2021). Introduction to Data Science - Data Analysis and Prediction Algorithms with R. CRC Press. URL: rafalab.github.io.
- <<wasser> Wasser L (Apr 8, 2021). Installing & Updating Packages in R [tutorial]. URL: neonscience.org.

Footnotes:

[1] A complete introduction to the "Tidyverse" is beyond my abilities. I don't work with the package much, and it consists of several packages each of which come with hundreds of functions. That's supposedly one of its strengths (not to me). Another popular, and useful, package is `readr`, which focuses on reading input into R. As I wrote before, ggplot2 actually predates the "Tidyverse" by a decade. If you're hungry for more, complete the DataCamp courses "Introduction to the Tidyverse" and "Introduction to Data Visualization with ggplot2", which are both quite enjoyable. I'm thinking about using the latter as an assignment for the "Data Visualization" course in fall 2022.

[2] The story of Hans Rosling and the Gapminder foundation has two sides. The bright side shines off Rosling's viral TED talks. The darker side is a little harder to detect, see e.g. "The One-Sided Worldview of Hans Rosling"

in Berggren (2018).

[3] On my machine, R version 4.1.3 cannot open documentation in a browser. This seems to be a Windows 10 issue (`internet.dll` is missing). I simply copied the file from *R-4.1.2/modules/x64* to *R-4.1.3/modules/x64* and that fixed it.

[4] This article, by the way, is a so-called "vignette", a long prose writeup documenting an R package. The best, and most used packages come with their own vignettes, which include use cases, examples etc., on top of the minimal package doc.

[5] This symbol has the added advantage that it uses the pipe operator `|` from UNIX, not just the redirection operator `>`. The pipe operator is used in UNIX-type systems like Linux to build complex commands from simple shell programs like `ls`, `cat`, `wc` or `grep`.

[6] You know me as a pipeline fanatic if you follow my Operating Systems course. However the UNIX command pipeline is completely different beast. It consists of single, super-focused, fast commands, each of them easy to understand, that unfold their great power when working side by side in a pipeline. The R pipeline only takes the general concept and idea from UNIX. In my view, it is unnecessary, slows process down and makes debugging much harder.

[7] Yet another issue I have whenever the word "misinformation" is used - and "disinformation" is even worse in this regard - it instantly summons the idea of an agent, and gives thereby rise to conspiracy theories before other avenues of explanation were explored.

[8] Look it up in the `facet_grid` documentation.

Author: Marcus Birkenkrahe
Created: 2022-04-04 Mon 16:19