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).
- `[ ]`

  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. Do this now before starting R.

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

- `[ ]`

  Check where your computer gets its CRAN packages from. If you put the code 1 into `.Rprofile`

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

  ```
                           CRAN
  "https://cran.microsoft.com"
  ```

- `[ ]`

  We need a couple of packages and a dataset. Put the code in [[lockAndLoad] 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

```
##      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.
  - `[ ]` The original Gapminder data set is much larger than the subset at CRAN or the dslabs dataset.
- `[ ]` 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. Check here what to do to your Emacs + ESS setup to get rid of the extra characters. Tibbles are limited to 10 rows by default.
- `[ ]` 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.
- `[ ]`

  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.

```
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

3 Checking and getting to know the data

- `[ ]`

  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
```

- [ ] Check the local help for the dslabs `gapminder` dataset for the meaning of the variables
- [ ]

  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
```

- [ ]

  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)
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
   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
```

- [ ]

  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
```

- [ ]

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

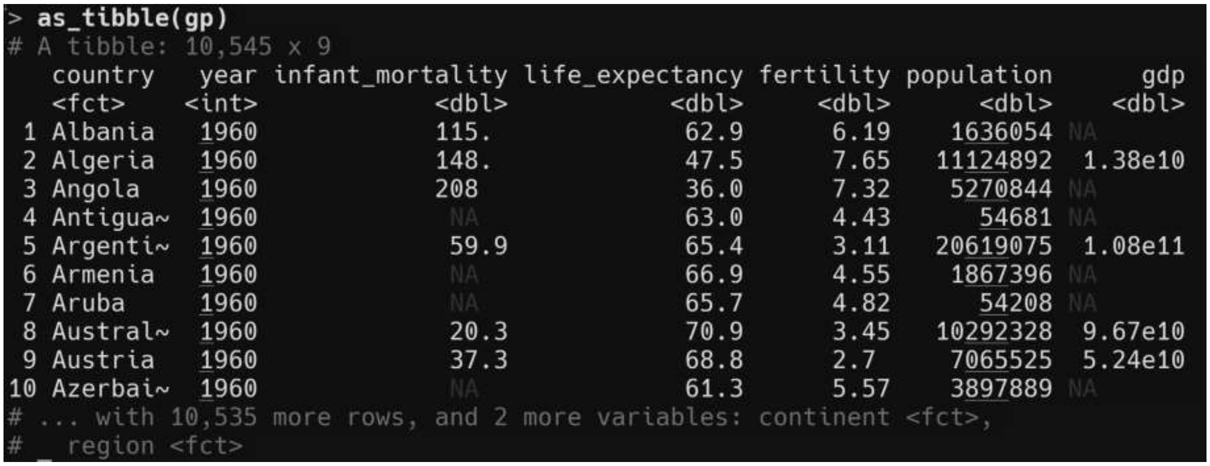
  In Emacs, you will see the control characters obscuring the display. To view it as it was meant to look like, switch to the R console in the **R** buffer and run the command there.

```
as_tibble(gm)
```

```
 [90m# A tibble: 10,545 x 9 [39m
   country              year infant_mortality life_expectancy fertility population
    [3m [90m<fct> [39m [23m               [3m [90m<int> [39m [23m            [3m [90m<db
 [90m 1 [39m Albania               [4m1 [24m960            115.             62.9      6.19
 [90m 2 [39m Algeria               [4m1 [24m960            148.             47.5      7.65
 [90m 3 [39m Angola                [4m1 [24m960            208              36.0      7.32
 [90m 4 [39m Antigua and Barbuda   [4m1 [24m960              [31mNA [39m              63.0
 [90m 5 [39m Argentina             [4m1 [24m960             59.9             65.4      3.11
 [90m 6 [39m Armenia               [4m1 [24m960              [31mNA [39m              66.9
 [90m 7 [39m Aruba                 [4m1 [24m960              [31mNA [39m              65.7
 [90m 8 [39m Australia             [4m1 [24m960             20.3             70.9      3.45
 [90m 9 [39m Austria               [4m1 [24m960             37.3             68.8      2.7
 [90m10 [39m Azerbaijan            [4m1 [24m960              [31mNA [39m              61.3
 [90m# ... with 10,535 more rows [39m
```

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)[3]. Since last year, base R also has its own pipeline operator, which is a little less obscure looking.

  You don't see the potential power of pipes if you only use one. It becomes a handy tool (to some, not to me[4]) 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.

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

```
[90m# A tibble: 10,545 x 9 [39m
  country              year infant_mortality life_expectancy fertility population
   [3m [90m<fct> [39m [23m                 [3m [90m<int> [39m [23m              [3m [90m<db

[90m 1 [39m Albania               [4m1 [24m960             115.             62.9      6.19
[90m 2 [39m Algeria               [4m1 [24m960             148.             47.5      7.65
[90m 3 [39m Angola                [4m1 [24m960             208              36.0      7.32
[90m 4 [39m Antigua and Barbuda   [4m1 [24m960               [31mNA [39m              63.0
[90m 5 [39m Argentina             [4m1 [24m960              59.9            65.4      3.11
[90m 6 [39m Armenia               [4m1 [24m960               [31mNA [39m              66.9
[90m 7 [39m Aruba                 [4m1 [24m960               [31mNA [39m              65.7
[90m 8 [39m Australia             [4m1 [24m960              20.3            70.9      3.45
[90m 9 [39m Austria               [4m1 [24m960              37.3            68.8      2.7
[90m10 [39m Azerbaijan            [4m1 [24m960               [31mNA [39m              61.3
[90m# ... with 10,535 more rows [39m
[90m# A tibble: 10,545 x 9 [39m
  country              year infant_mortality life_expectancy fertility population
   [3m [90m<fct> [39m [23m                 [3m [90m<int> [39m [23m              [3m [90m<db
[90m 1 [39m Albania               [4m1 [24m960             115.             62.9      6.19
[90m 2 [39m Algeria               [4m1 [24m960             148.             47.5      7.65
[90m 3 [39m Angola                [4m1 [24m960             208              36.0      7.32
[90m 4 [39m Antigua and Barbuda   [4m1 [24m960               [31mNA [39m              63.0
[90m 5 [39m Argentina             [4m1 [24m960              59.9            65.4      3.11
```

```
[90m 6 [39m Armenia              [4m1 [24m960            [31mNA [39m              66.9
[90m 7 [39m Aruba                [4m1 [24m960            [31mNA [39m              65.7
[90m 8 [39m Australia            [4m1 [24m960           20.3            70.9      3.45
[90m 9 [39m Austria              [4m1 [24m960           37.3            68.8      2.7
[90m10 [39m Azerbaijan           [4m1 [24m960            [31mNA [39m              61.3
[90m# ... with 10,535 more rows [39m
```

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

- `[ ]`

  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	33.6	South Africa	10.5

- `[ ]`

  Let's run the numbers, then put the results in the table 1

  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
```

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

- [ ]

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

```
gm %>%
  filter(year == 2015 & country %in% c("Poland", "South Korea")) %>%
  select(country, infant_mortality)
```

```
      country infant_mortality
1 South Korea              2.9
2      Poland              4.5
```

```
gm %>%
  filter(year == 2015 & country %in% c("Malaysia", "Russia")) %>%
  select(country, infant_mortality)
```

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

```
gm %>%
  filter(year == 2015 & country %in% c("Pakistan", "Vietnam")) %>%
  select(country, infant_mortality)
```

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

```
gm %>%
  filter(year == 2015 & country %in% c("Thailand", "South Africa")) %>%
  select(country, infant_mortality)
```

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

5 TODO Scatterplots

6 TODO Faceting

7 TODO Time series plots

8 TODO Data transformations

9 TODO Boxplots and ridge plots

10 TODO Data presentation

11 TODO Summary of Concepts

12 TODO Summary of Code

13 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] 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.

[4] 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.

Author: Marcus Birkenkrahe

Created: 2022-03-09 Wed 14:40

Validate