Exploring a data frame with `dplyr` and `ggplot2`.

First steps.

Silvie Cinková

2025-07-28

Table of contents

1 `readr`, `dplyr`, and `ggplot2` 2

2 Path management 3

3 Read the Gapminder labor cost data set 3

4 `dplyr::glimpse` 3

5 `summary` 4

6 `summary` with categorical columns as factors 4

7 Rename a column with base R 5

8 Rename a column with `dplyr` 5

9 Filter rows with `dplyr` 5

10 List distinct values with `dplyr::distinct` 6

11 Plot the data set 7

11.1 Comment on the plot 8

12 Different mapping in the same plot **8**

12.1 Comment on this plot as well 9

13 Save a plot **9**

14 First insights about `ggplot2` **10**

15 `ggplot2` ≈ implemented *Grammar of Graphics* **10**

16 Layers of `ggplot2` **11**

17 Data **11**

18 Aesthetic scales (aka *mappings*, *aesthetics*) **11**

19 Geometric objects aka *geoms* **12**

20 A neater example dataset: just Czechia and Germany **12**

21 Syntax **13**

1 `readr`, `dplyr`, and `ggplot2`

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(readr)
library(ggplot2)
library(glue) # just to make long strings wrap in PDF
```

These are the three libraries you need most when you explore a tabular dataset.

2 Path management

```
project_path <- "~/NPFL112_2025_ZS/"
datasaving_folder <- "DATA.NPFL112"
output_folder <- "OUTPUT_FILES"
```

3 Read the Gapminder labor cost data set

```
project_path <- "~/NPFL112_2025_ZS/"
datasaving_folder <- "DATA.NPFL112"
myfilepath <- file.path(project_path,
 ↪  "DATA.NPFL112/gapminder_hourly_labour_cost_constant_2017_usd--by--geo--time.csv"
 ↪   )
laborcost_df <- read_csv(file = myfilepath,
                         show_col_types = TRUE)
```

```
Rows: 548 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): geo
dbl (2): time, hourly_labour_cost_constant_2017_usd

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

It is the file demonstrated in the previous session. We saved it into the folder `datasets_ATRIUM`. You can also get it at https://raw.githubusercontent.com/open-numbers/ddf--gapminder--systema_globalis/refs/heads/master/countries-etc-datapoints/ddf--datapoints--hourly_labour_cost_constant_2017_usd--by--geo--time.csv.

Watch the message `read_csv` gives you about the file. You can suppress it by overriding the default to `show_col_types = FALSE`.

4 `dplyr::glimpse`

- peek at the dataset (tilted 90°)

```
glimpse(laborcost_df)
```

```
Rows: 548
Columns: 3
$ geo                                 <chr> "arg", "arg", "arm", "arm", "arm"~
$ time                                <dbl> 2011, 2012, 2011, 2012, 2013, 201~
$ hourly_labour_cost_constant_2017_usd <dbl> 0.92, 1.04, 4.23, 4.59, 6.12, 6.0~
```

Gives you the number of rows and columns, the column names with their data class, and it also shows as many elements (values) in each column as to fit your screen.

5 summary

```
summary(laborcost_df)
```

```
    geo                 time      hourly_labour_cost_constant_2017_usd
Length:548         Min.   :1994   Min.   : 0.000
Class :character   1st Qu.:2005   1st Qu.: 9.867
Mode  :character   Median :2011   Median :18.320
                   Mean   :2010   Mean   :19.686
                   3rd Qu.:2017   3rd Qu.:26.915
                   Max.   :2020   Max.   :48.720
```

Gives you the "five-number summary" of each numeric column (it's often called this way, although the numbers are obviously six…). With categorical columns, it depends, whether the column is a character vector or a factor.

6 summary with categorical columns as factors

```
     geo           time      hourly_labour_cost_constant_2017_usd
cze    : 22   Min.   :1994   Min.   : 0.000
svn    : 22   1st Qu.:2005   1st Qu.: 9.867
cyp    : 21   Median :2011   Median :18.320
deu    : 21   Mean   :2010   Mean   :19.686
pol    : 21   3rd Qu.:2017   3rd Qu.:26.915
svk    : 21   Max.   :2020   Max.   :48.720
(Other):420
```

If you have a data frame with categorical variables converted to factors, the summary will show you a glimpse of their **levels** (unique values) and their frequencies, as well as tell you how many levels there are.

So far, do not worry about factors. The `dplyr` as well as the `ggplot2` libraries do this factor conversion on the fly whenever they need it.

7 Rename a column with base R

`hourly_labour_cost_constant_2017_usd` too long, shorten to `labor_cost`.

```
colnames(laborcost_df)[colnames(laborcost_df) ==
                         "hourly_labour_cost_constant_2017_usd"] <-
 ↪  "labor_cost"
```

```
colnames(laborcost_df)
```

```
[1] "geo"        "time"       "labor_cost"
```

8 Rename a column with `dplyr`

```
laborcost_df <- rename(.data = laborcost_df,
                       labor_cost =
 ↪  hourly_labour_cost_constant_2017_usd
                       )
colnames(laborcost_df)
```

```
[1] "geo"        "time"       "labor_cost"
```

You already know you could have named all columns your way when reading in the file. Here are two ways to rename a column: one base-R-like and the other one provided by `dplyr`.

9 Filter rows with `dplyr`

```
cze_deu_df <- dplyr::filter(laborcost_df, geo %in% c("cze", "deu"))
write_csv(cze_deu_df, file.path(project_path, datasaving_folder,
 ↪  "gapminder_laborcost_cze_deu.csv"))
```

`dplyr::filter` keeps only rows that meet a condition (or a set of conditions) that you determine with logical operators.

We will return to this in more detail in a separate session.

10 List distinct values with `dplyr::distinct`

```
dplyr::distinct(.data = laborcost_df, geo, .keep_all = FALSE)
```

geo
arg
arm
aus
aut
aze
bel
bgr
can
che
chl
cri
cyp
cze
deu
dnk
esp
est
fin
fra
gbr
geo
grc
hrv
hun
irl
isl
isr
ita
kaz
ltu
lux
lva

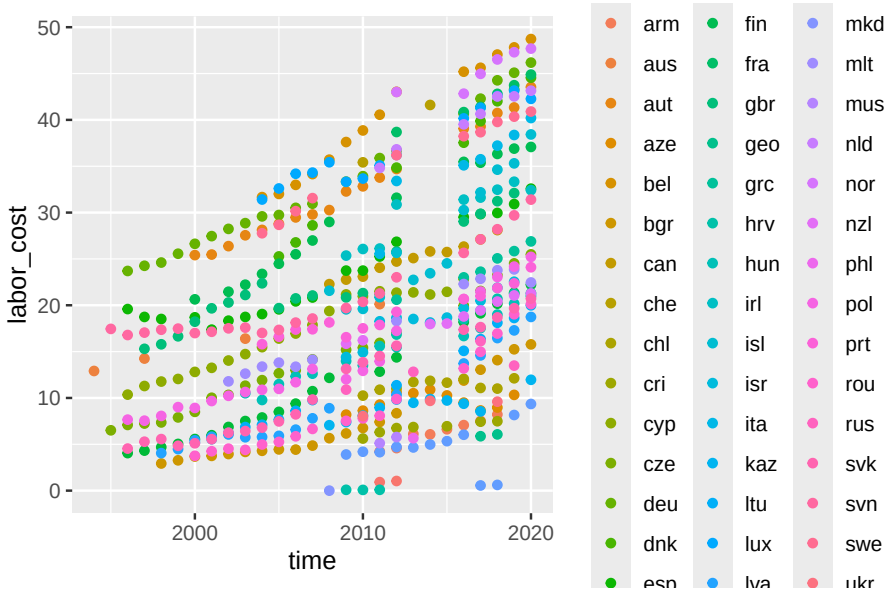
geo
mda
mkd
mlt
mus
nld
nor
nzl
phl
pol
prt
rou
rus
svk
svn
swe
ukr

The data set contains almost 50 countries. They will probably not be distinguishable in the plot.

11 Plot the data set

- obviously not a very helpful plot, but anyway…

```
laborcost_plot <- ggplot(data = laborcost_df,
       mapping = aes(x = time,
                     y = labor_cost,
                     color = geo)) +
  geom_point()
laborcost_plot
```



11.1 Comment on the plot

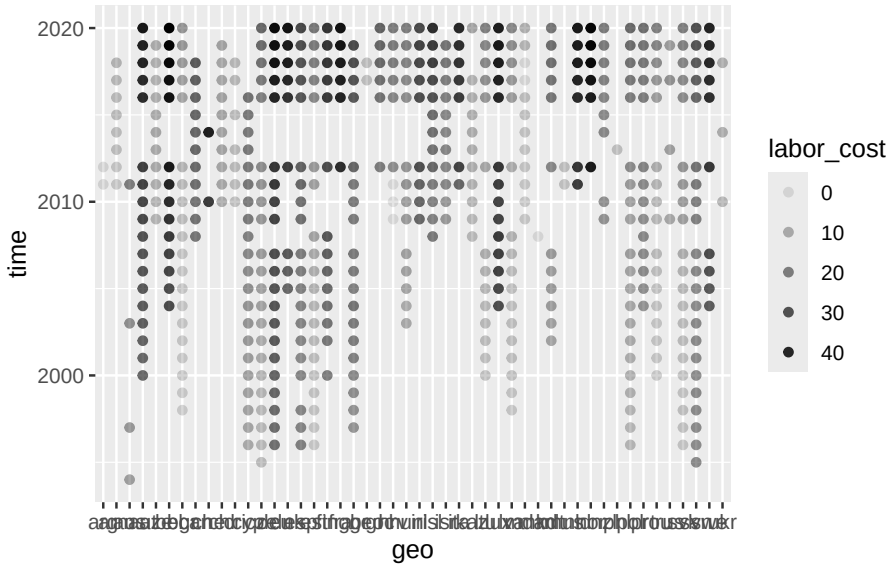
How do you call plots with points and two axes?

How many variables does the plot capture and how? Which are the types of variables?

Look at the script. Try to dissect it in parts and interpret them.

12 Different mapping in the same plot

```
laborcost_plot_size <- ggplot(data = laborcost_df,
                              mapping = aes(x = geo,
                                            y = time,
                                            alpha = labor_cost)) +
  geom_point()
laborcost_plot_size
```



12.1 Comment on this plot as well

How does it capture the variables now? Is it telling a different story?

13 Save a plot

```
ggsave(filename = file.path(project_path, output_folder,
 ↪  "laborcost_plot.svg"),
       plot = laborcost_plot)
```

```
Saving 5.5 x 3.5 in image
```

```
ggsave(filename = file.path(project_path, output_folder,
 ↪  "laborcost_plot.png"),
       plot = laborcost_plot,
       device = grDevices::png)
```

```
Saving 5.5 x 3.5 in image
```

```
# device = "png" or this when RStudio hiccups
ggsave(filename = file.path(project_path, output_folder,
 ↪  "laborcost_plot.pdf"),
       plot = laborcost_plot)
```

```
Saving 5.5 x 3.5 in image
```

```
list.files(path = ,file.path(project_path, output_folder), pattern =
↪  "laborcost_plot")
```

```
[1] "laborcost_plot.pdf" "laborcost_plot.png" "laborcost_plot.svg"
```

`ggsave` wants a file name (with path) including the format suffix.

Sometimes, especially when you run your scripts over and over in Quarto, something breaks behind the scenes and you get cryptical error messages about mismatch of graphical devices or similar. Sometimes the best solution is to restart R, close the file, clean the Environment and reopen the file. Or run the command directly in the Console.

14 First insights about `ggplot2`

- Specific syntax
- A plot is an object (goes in a variable)
- Can be saved to files - include desired format in the file name
- Maps variables on X, Y, color, transparency…

Feel free to add more insights or impressions.

15 `ggplot2` ≈ implemented *Grammar of Graphics*

https://ggplot2.tidyverse.org/

- All plots have the same logic and components in a few layers.
- Not just drawings, statistical transformations behind the scenes (e.g. histogram)
- When you see a `ggplot2` plot you have an idea how the source table is structured

Grammar of Graphics is a book by Leland Wilkinson.

Base R plots: many independent packages → less consistence, harder to learn properly

You cannot save base R plots as objects.

16 Layers of `ggplot2`

- Data
- Aesthetic mappings + Facets (subgraphs)
- Geometric objects (aka *geoms*)
- Statistical transformations (aka *stats*)
- Coordinate system
- Theme

Today just data, aesthetic mappings and geoms. This is enough to make an informative plot if you don't need it super pretty for publication.

17 Data

- data frame with tidy data structure
  - each observation on one row
  - each variable in one column
- categorical variables automatically read as factors

18 Aesthetic scales (aka *mappings*, *aesthetics*)

- axes X, Y
- shape / linetype
- color / fill / stroke
- size / linewidth
- alpha (transparency)
- label

19 Geometric objects aka *geoms*

plot types, such as:

- histogram
- scatterplot
- barplot
- boxplot
- heatmap
- and many others

Try and sketch these plots by hand (fake data). What are these plots good at telling? How many of which variables?

20 A neater example dataset: just Czechia and Germany

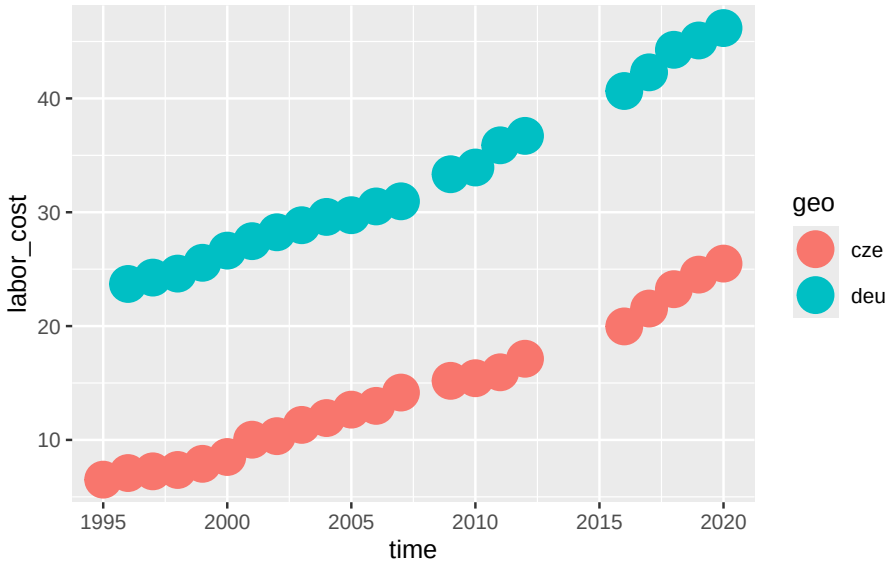
- only two values of a categorical variable, under 50 rows

```
cze_deu_df <- read_csv(file.path(project_path,
 ↪  datasaving_folder,"gapminder_laborcost_cze_deu.csv"))
```

```
Rows: 43 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): geo
dbl (2): time, labor_cost

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
ggplot(data = cze_deu_df,
       mapping = aes(x = time, y = labor_cost, color = geo)) +
  geom_point(size = 7)
```



21 Syntax

`ggplot(data, mappings) + geom_…( )`

or

`ggplot(data) + geom_…(mappings)`

The `mapping` argument always takes the `aes()` function.

- The plus sign never works at the beginning of a new line.
- The geometric object (actual plot) is generated by a `geom_`*something*`()` function.
- Different `geom_` functions require/accept different aesthetic scales. Look them up in help/cheat sheet.