Possible solutions for the recap exercises

Claudius Gräbner-Radkowitsch

Table of contents

1 Packages used

```
library(here)
library(dplyr)
library(tidyr)
library(ggplot2)
library(data.table)
```

2 CO2

We first import the raw data. Please make sure you use the `here`-package and adjust the relative paths of the following code.

Since many of the column headers were numbers (a.k.a. years), we need to make explicit that these are not values but header names. We do so by seeting the optional argument `header = TRUE`:

```
co2_data_raw <- fread("co2_raw.csv", header = TRUE)
```

After inspecting the data using functions such as `str()`, `unique()` or `head()`, we first remove colums we obviously do not need and that might be irritating:

```
co2_data_tidy_1 <- co2_data_raw %>%
  select(-c(
    "Indicator Name", "Indicator Code",
    # unique() tells you there is only one indicator
    "Country Code", # Not needed
    "V69" # Sometimes such erroneous columns are part of what you download
  ))
```

We then move the year columns into rows by using `tidyr::pivot_longer()`:

```
co2_data_tidy_2 <- co2_data_tidy_1 %>%
  tidyr::pivot_longer(
    cols = -"Country Name",
    names_to = "year",
    values_to = "co2_percap")
head(co2_data_tidy_2)
```

```
# A tibble: 6 x 3
  `Country Name` year  co2_percap
  <chr>          <chr>      <dbl>
1 Aruba          1960          NA
2 Aruba          1961          NA
3 Aruba          1962          NA
4 Aruba          1963          NA
5 Aruba          1964          NA
6 Aruba          1965          NA
```

We see that the year column is still a `character`. So me transform it into a `double` to then filter the years. We can also filter for the required countries within the same function call and then rename the column:

```r
co2_data_tidy_3 <- co2_data_tidy_2 %>%
  mutate(year = as.double(year)) %>%
  filter(
    year >= 2000, year<=2020,
    `Country Name` %in% c(
      "South Africa", "United States", "Sub-Saharan Africa",
      "European Union", "Germany", "China")
  ) %>%
  rename(country = `Country Name`)
```

We could have done everything in one call as well:

```r
co2_data_tidy <- co2_data_raw %>%
  select(-c(
    "Indicator Name", "Indicator Code",
    # unique() tells you there is only one indicator
    "Country Code", # Not needed
    "V69" # Sometimes such erroneous columns are part of what you download
  )) %>%
  tidyr::pivot_longer(
    cols = -"Country Name",
    names_to = "year",
    values_to = "co2_percap") %>%
  mutate(year = as.double(year)) %>%
  filter(
    year >= 2000, year<=2020,
    `Country Name` %in% c(
      "South Africa", "United States", "Sub-Saharan Africa",
      "European Union", "Germany", "China")
  ) %>%
  rename(country = `Country Name`)
```

Then think about a useful location to store the data and do something like:

```r
fwrite(co2_data_tidy, file = here("data/tidy/co2_tidy.csv"))
```

3 Data wrangling I

Please make sure you use the `here`-package and adjust the relative paths of the following code:

Compute, for each country, the percentage change of the spending from the year 2010 to the year 2020 and save this as a variable called `perc_change`.

```
educ_exercise_data_raw <- fread("education_income.csv")

educ_exercise_data <- educ_exercise_data_raw %>%
  dplyr::select(-c("income", "GDPpc")) %>%
  dplyr::filter(year %in% c(2010, 2020)) %>%
  tidyr::pivot_wider(
    names_from = "year",
    values_from = "EducationSpending"
    ) %>%
  dplyr::mutate(
    perc_change = ((`2020`-`2010`)/`2010`)*100
    ) %>%
  dplyr::filter(!is.na(perc_change))
head(educ_exercise_data)
```

```
# A tibble: 6 x 4
  iso3c `2010` `2020` perc_change
  <chr>  <dbl>  <dbl>       <dbl>
1 ALB     3.41   3.34       -2.07
2 AND     2.98   2.86       -4.05
3 AGO     3.42   2.74      -19.8
4 ARG     5.02   5.28        5.18
5 ARM     3.25   2.71      -16.7
6 AUS     5.54   5.61        1.27
```

Then think about a useful location to store the data and do something like:

```
data.table::fwrite(
  x = educ_exercise_data,
  file = here("data/tidy/educ_perc_change.csv"))
```

4 Data wrangling II

We use `educ_exercise_data_raw` as imported above as a starting point and proceed as follows:

Compute for each income group the average expense of education over the whole period. Make sure missing values are ignored.

Save the new data set under a useful name in an adequate location.

```r
educ_exercise_summarized <- educ_exercise_data_raw %>%
  dplyr::summarise(
    EducExpense_avg = mean(EducationSpending, na.rm = TRUE),
    .by = "income")
educ_exercise_summarized
```

```
               income EducExpense_avg
1          Low income        3.342121
2 Upper middle income        4.723520
3 Lower middle income        4.534556
4         High income        4.644050
```

Then think about a useful location to store the data and do something like:

```r
data.table::fwrite(
  x = educ_exercise_summarized,
  file = here("data/tidy/educ_perc_income-groups.csv"))
```

5 Visualization and Quarto

The quarto header should look like this:

```
title: "Sessions 12 and 13: Recap and Practice"
author: "Claudius Gräbner-Radkowitsch"
format:
  html:
    number-sections: true
    table-of-contents: true
    toc-location: body
execute:
  echo: false
  warning: false
  message: false
```

6 Visualization and Quarto

To read in the data set do something as the following, but make sure you are using the `here`-package and set the path accordingly.

```
child_mortality <- data.table::fread("child_mortality.csv")
head(child_mortality)
```

```
    iso3c  year ChildMortality    GDPpc
   <char> <int>          <num>    <num>
1:    AFG  2017           64.6 2096.093
2:    AFG  2014           73.4 2110.830
3:    AFG  2016           67.2 2023.835
4:    AFG  2012           80.3 1958.448
5:    AFG  2021           55.7 1673.144
6:    AFG  2007          100.0 1287.064
```

To summarize the data:

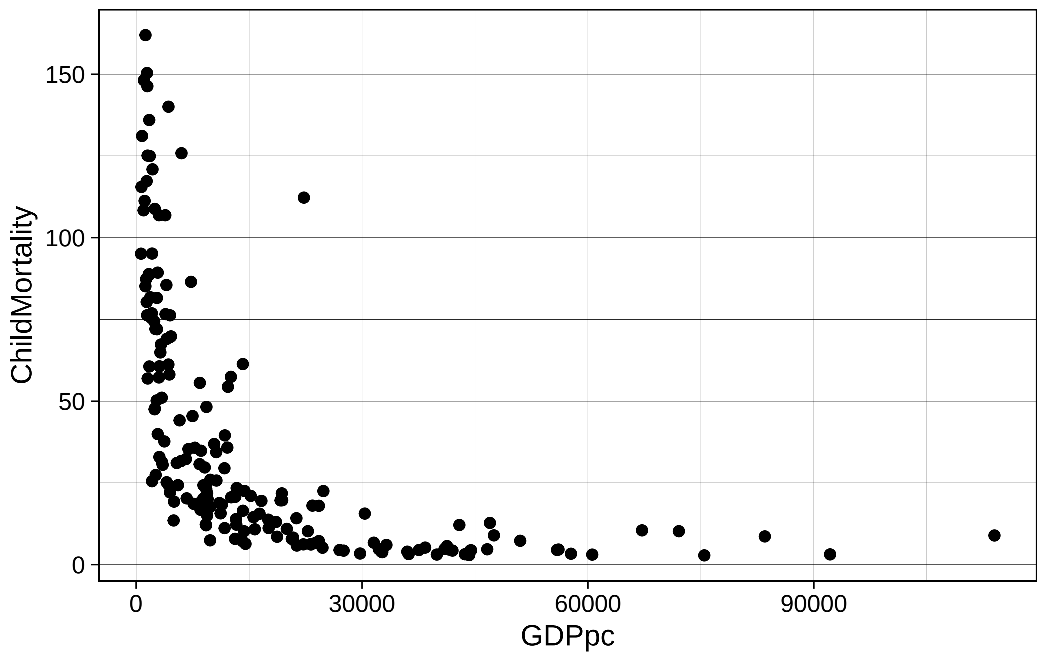
```
child_mortality_summarized <- child_mortality %>%
  dplyr::summarise(
    ChildMortality = mean(ChildMortality, na.rm = TRUE),
    GDPpc = mean(GDPpc, na.rm = TRUE),
    .by = "iso3c")
head(child_mortality_summarized)
```

```
  iso3c ChildMortality     GDPpc
1   AFG      88.459091  1660.568
2   ALB      15.031818  9437.101
3   DZA      29.486364 11735.174
4   ASM            NaN       NaN
5   AND       4.718182       NaN
6   AGO     125.831818  6029.127
```

We can then directly create a simple scatter plot:

```
ggplot2::ggplot(
  data = child_mortality_summarized,
  mapping = aes(x = GDPpc, y = ChildMortality)
  ) +
```

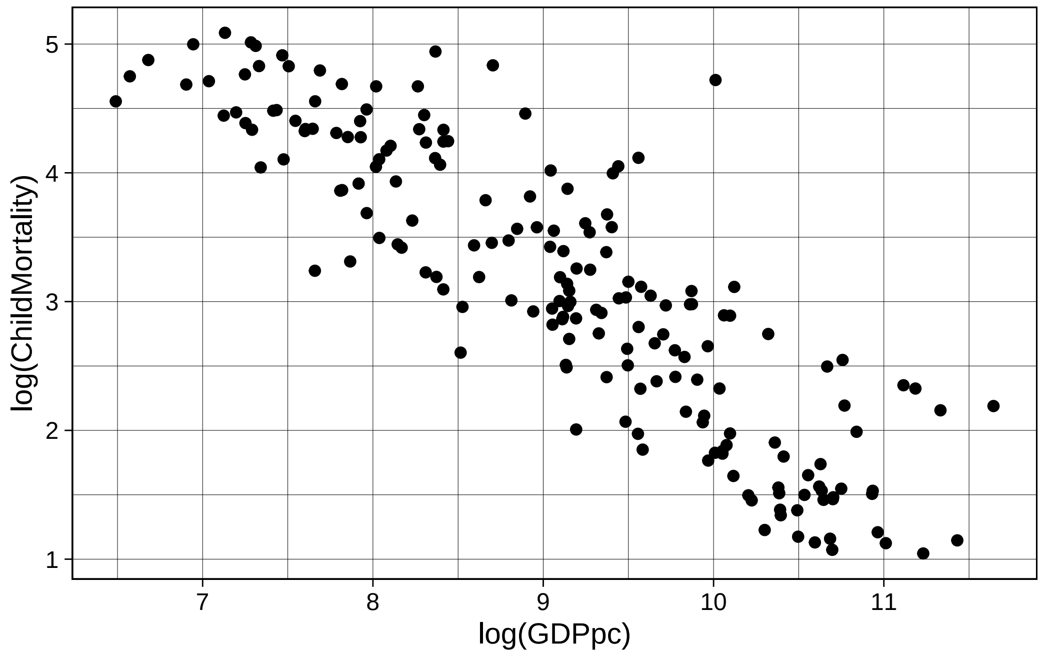
```r
  geom_point() +
  theme_linedraw()
```



We see a clear non-linear relationship.

We now plot the data in logarithms. You can do this by changing the underlying data, rescale an axis, or make the change directly in the `data` argument of `ggplot2::ggplot()`:

```r
ggplot2::ggplot(
  data = child_mortality_summarized,
  mapping = aes(x = log(GDPpc), y = log(ChildMortality))
  ) +
  geom_point() +
  theme_linedraw()
```



The relationship now becomes almost linear. This is typical for relationships that are exponential. We can say: an increase in GDP per capita by one percent is on average associated with a reduction of child mortality by 0.83 per cent (the latter value is given by a regression, but we come to this later).