Data Wrangling (1)

Haohan Chen

Last update: October 05, 2023

Objectives of this Lecture

This lecture introduces data wrangling with R. Using V-Dem data as an example, we will learn how to use the wrangle data with a set of `tidyverse` functionality. Specifically, we will focus on functions...

1. to import and export data: `read_csv` , `write_csv` (with a brief introduction to other data import/export functions from `readr`).
2. to take a subset of *columns* in the existing data: `select`
3. to rename columns: `rename`
4. to take a subset of *rows* by some simple conditions: `slice_`
5. to take a subset of *rows* by some more complicated conditions: `filter`
6. to sort the rows based on the value of one or multiple columns: `arrange`
7. to perform (4) (5) (6) group by group: `group_by`, `ungroup`
8. to create new columns in the data: `group_by`, `mutate`, `ungroup`
9. to summarize the data: `group_by`, `summarise`, `ungroup`

Outline of In-Class Demo

To demonstrate the above functionality, we will use real-world political data from V-Dem. Specifically, we will use the above function to explore the state of global economic development from 1984 to 2022. Our effort will take the following step (with one-on-one mappings with the above tools).

1. Read a part of pre-processed V-Dem data into R: 1984-2022 "external" data in the V-Dem dataset.
2. Consulting the dataset's codebook and take a **subset** of indicators of *economic development* (along with country-year identifiers).
    - *See a list of country-yer identifiers on p. 5 of the codebook (under "1.7 Identifier Variables in the V-Dem Datasets").*
    - *See a list of development indicators on p. 23 of the codebook (under "9. Background Factors").*
3. Rename the column to name their names informative to readers.
4. Find the country-year with the *highest* and *lowest* level of economic development. In addition, create a dataset containing a random sample of country-year in the dataset.
5. Create a dataset focusing on the economic development of Asian countries and regions; Create a dataset that contains only countries/ regions whose development level pass certain threshold.

6. Create a dataset whose rows are sorted by the development level of country-year.
7. Create a dataset that contains the year of the higest development level for each country/ region respectively.
8. Add the following economic indicators to the data:
    1. Country-year development level with reference to that of 1984.
    2. Year-on-year economic growth.
9. Perform a data availability/ integrity check. Then aggregate the data into a new country-level dataset which contains the following indicators:
    1. Average development level from 1984 to 2022.
    2. Magnitude of growth from 1984 to 2022.

In-Class Exercise

The quality of education has a decisive effect on a country's future development. Applying the data wrangling tools we introduce in this lecture, perform the following task:

1. **Coodbook lookup**. Look up the codebook, answer the following questions:
    1. What indicators regarding the quality of education are available in the V-Dem datasets?
    2. What are the data's coverage (i.e., for which countries and years do we have data?)
    3. What are their sources? Provide the link to least 1 source.
2. **Subset by columns**
    1. Create a dataset containing only the country-year identifiers and indicators of education quality.
    2. Rename the columns of education quality to make them informative.
3. **Subset by rows**
    1. List 5 countries-years that have the highest education level among its population.
    2. List 5 countries-years that suffer from the most severe inequality in education.
4. **Summarize the data**
    1. Check data availability: For which countries and years are the indicators of education quality available?
    2. Create two types of country-level indicators of education quality
        1. Average level of education quality from 1984 to 2022
        2. Change of education quality from 1984 to 2022
    3. Examine the data and *briefly* discuss: Which countries perform the best and the worst in terms of education quality in the past four decades?

Submission requirement: You will submit your outputs through Moodle. In your submission:

1. Attach a PDF document rendered by Rmarkdown
2. In the text field of your submission, include the link to the corresponding Rmarkdown file in your *DaSPPA portfolio* GitHub repo.

Due: October 6, 2023

Note: Please only use the functions we cover in this lecture for this exercise. There is absolutely no need to perform any data visualization for this exercise... We will get there in later lectures.

Further reading

- R for Data Science (2e) Chapters 4, 5, 8: https://r4ds.hadley.nz/
- `readr` documentation (note: read the "cheatsheet"): https://readr.tidyverse.org/
- `dplyr` documentation (note: read the "cheatsheet"): https://dplyr.tidyverse.org/
- V-Dem documentation: https://v-dem.net/

Demo

0. Load the `tidyverse` Packages

This section loads the packages we need in this lecture.

```
library(tidyverse)
```

1. Import and Export the V-Dem Data

This section loads the VDEM dataset and describe its basic information

```
d <- read_csv("_DataPublic_/vdem/1984_2022/vdem_1984_2022_external.csv")
```

```
## Rows: 6789 Columns: 211
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr    (3): country_name, country_text_id, histname
## dbl  (207): country_id, year, project, historical, codingstart, codingend, c...
## date   (1): historical_date
##
## 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.
```

2. Select economic development indicators

We start by examining the dataset. `name()` is almost always the first function I apply to a dataset. It gives us the names of all the columns

```
names(d)
```

```
##   [1] "country_name"                "country_text_id"
##   [3] "country_id"                  "year"
##   [5] "historical_date"             "project"
##   [7] "historical"                  "histname"
##   [9] "codingstart"                 "codingend"
##  [11] "codingstart_contemp"         "codingend_contemp"
##  [13] "codingstart_hist"            "codingend_hist"
##  [15] "gapstart1"                   "gapstart2"
##  [17] "gapstart3"                   "gapend1"
##  [19] "gapend2"                     "gapend3"
```

```
##  [21] "gap_index"                 "COWcode"
##  [23] "e_v2x_api_3C"              "e_v2x_api_4C"
##  [25] "e_v2x_api_5C"              "e_v2x_civlib_3C"
##  [27] "e_v2x_civlib_4C"           "e_v2x_civlib_5C"
##  [29] "e_v2x_clphy_3C"            "e_v2x_clphy_4C"
##  [31] "e_v2x_clphy_5C"            "e_v2x_clpol_3C"
##  [33] "e_v2x_clpol_4C"            "e_v2x_clpol_5C"
##  [35] "e_v2x_clpriv_3C"           "e_v2x_clpriv_4C"
##  [37] "e_v2x_clpriv_5C"           "e_v2x_corr_3C"
##  [39] "e_v2x_corr_4C"             "e_v2x_corr_5C"
##  [41] "e_v2x_cspart_3C"           "e_v2x_cspart_4C"
##  [43] "e_v2x_cspart_5C"           "e_v2x_delibdem_3C"
##  [45] "e_v2x_delibdem_4C"         "e_v2x_delibdem_5C"
##  [47] "e_v2x_EDcomp_thick_3C"     "e_v2x_EDcomp_thick_4C"
##  [49] "e_v2x_EDcomp_thick_5C"     "e_v2x_egal_3C"
##  [51] "e_v2x_egal_4C"             "e_v2x_egal_5C"
##  [53] "e_v2x_egaldem_3C"          "e_v2x_egaldem_4C"
##  [55] "e_v2x_egaldem_5C"          "e_v2x_elecoff_3C"
##  [57] "e_v2x_elecoff_4C"          "e_v2x_elecoff_5C"
##  [59] "e_v2x_execorr_3C"          "e_v2x_execorr_4C"
##  [61] "e_v2x_execorr_5C"          "e_v2x_feduni_3C"
##  [63] "e_v2x_feduni_4C"           "e_v2x_feduni_5C"
##  [65] "e_v2x_frassoc_thick_3C"    "e_v2x_frassoc_thick_4C"
##  [67] "e_v2x_frassoc_thick_5C"    "e_v2x_freexp_3C"
##  [69] "e_v2x_freexp_4C"           "e_v2x_freexp_5C"
##  [71] "e_v2x_freexp_altinf_3C"    "e_v2x_freexp_altinf_4C"
##  [73] "e_v2x_freexp_altinf_5C"    "e_v2x_gencl_3C"
##  [75] "e_v2x_gencl_4C"            "e_v2x_gencl_5C"
##  [77] "e_v2x_gencs_3C"            "e_v2x_gencs_4C"
##  [79] "e_v2x_gencs_5C"            "e_v2x_gender_3C"
##  [81] "e_v2x_gender_4C"           "e_v2x_gender_5C"
##  [83] "e_v2x_genpp_3C"            "e_v2x_genpp_4C"
##  [85] "e_v2x_genpp_5C"            "e_v2x_jucon_3C"
##  [87] "e_v2x_jucon_4C"            "e_v2x_jucon_5C"
##  [89] "e_v2x_libdem_3C"           "e_v2x_libdem_4C"
##  [91] "e_v2x_libdem_5C"           "e_v2x_liberal_3C"
##  [93] "e_v2x_liberal_4C"          "e_v2x_liberal_5C"
##  [95] "e_v2x_mpi_3C"              "e_v2x_mpi_4C"
##  [97] "e_v2x_mpi_5C"              "e_v2x_partip_3C"
##  [99] "e_v2x_partip_4C"           "e_v2x_partip_5C"
## [101] "e_v2x_partipdem_3C"        "e_v2x_partipdem_4C"
## [103] "e_v2x_partipdem_5C"        "e_v2x_polyarchy_3C"
## [105] "e_v2x_polyarchy_4C"        "e_v2x_polyarchy_5C"
## [107] "e_v2x_pubcorr_3C"          "e_v2x_pubcorr_4C"
## [109] "e_v2x_pubcorr_5C"          "e_v2x_suffr_3C"
## [111] "e_v2x_suffr_4C"            "e_v2x_suffr_5C"
## [113] "e_v2xcl_rol_3C"            "e_v2xcl_rol_4C"
## [115] "e_v2xcl_rol_5C"            "e_v2xcs_ccsi_3C"
## [117] "e_v2xcs_ccsi_4C"           "e_v2xcs_ccsi_5C"
## [119] "e_v2xdd_dd_3C"             "e_v2xdd_dd_4C"
## [121] "e_v2xdd_dd_5C"             "e_v2xdl_delib_3C"
## [123] "e_v2xdl_delib_4C"          "e_v2xdl_delib_5C"
## [125] "e_v2xeg_eqdr_3C"           "e_v2xeg_eqdr_4C"
## [127] "e_v2xeg_eqdr_5C"           "e_v2xeg_eqprotec_3C"
```

```
## [129] "e_v2xeg_eqprotec_4C"          "e_v2xeg_eqprotec_5C"
## [131] "e_v2xel_frefair_3C"           "e_v2xel_frefair_4C"
## [133] "e_v2xel_frefair_5C"           "e_v2xel_locelec_3C"
## [135] "e_v2xel_locelec_4C"           "e_v2xel_locelec_5C"
## [137] "e_v2xel_regelec_3C"           "e_v2xel_regelec_4C"
## [139] "e_v2xel_regelec_5C"           "e_v2xlg_legcon_3C"
## [141] "e_v2xlg_legcon_4C"            "e_v2xlg_legcon_5C"
## [143] "e_v2xme_altinf_3C"            "e_v2xme_altinf_4C"
## [145] "e_v2xme_altinf_5C"            "e_v2xps_party_3C"
## [147] "e_v2xps_party_4C"             "e_v2xps_party_5C"
## [149] "e_boix_regime"                "e_democracy_breakdowns"
## [151] "e_democracy_omitteddata"      "e_democracy_trans"
## [153] "e_fh_cl"                      "e_fh_pr"
## [155] "e_fh_rol"                     "e_fh_status"
## [157] "e_wbgi_cce"                   "e_wbgi_gee"
## [159] "e_wbgi_pve"                   "e_wbgi_rle"
## [161] "e_wbgi_rqe"                   "e_wbgi_vae"
## [163] "e_lexical_index"              "e_uds_median"
## [165] "e_uds_mean"                   "e_uds_pct025"
## [167] "e_uds_pct975"                 "e_coups"
## [169] "e_legparty"                   "e_autoc"
## [171] "e_democ"                      "e_p_polity"
## [173] "e_polcomp"                    "e_polity2"
## [175] "e_bnr_dem"                    "e_chga_demo"
## [177] "e_ti_cpi"                     "e_vanhanen"
## [179] "e_peaveduc"                   "e_peedgini"
## [181] "e_area"                       "e_regiongeo"
## [183] "e_regionpol"                  "e_regionpol_6C"
## [185] "e_cow_exports"                "e_cow_imports"
## [187] "e_gdp"                        "e_gdp_sd"
## [189] "e_gdppc"                      "e_gdppc_sd"
## [191] "e_miinflat"                   "e_pop"
## [193] "e_pop_sd"                     "e_total_fuel_income_pc"
## [195] "e_total_oil_income_pc"        "e_total_resources_income_pc"
## [197] "e_radio_n"                    "e_miferrat"
## [199] "e_mipopula"                   "e_miurbani"
## [201] "e_miurbpop"                   "e_pefeliex"
## [203] "e_peinfmor"                   "e_pelifeex"
## [205] "e_pematmor"                   "e_wb_pop"
## [207] "e_civil_war"                  "e_miinteco"
## [209] "e_miinterc"                   "e_pt_coup"
## [211] "e_pt_coup_attempts"
```

We may use some alternative functions that provides information about the dataset. The `str()` provides not only variable names, but also their data types and a few example data points.

```r
# Warning: If you have many variables, the output of str() will be lengthy!
str(d)
```

```
## spc_tbl_ [6,789 x 211] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ country_name               : chr [1:6789] "Mexico" "Mexico" "Mexico" "Mexico" ...
##  $ country_text_id            : chr [1:6789] "MEX" "MEX" "MEX" "MEX" ...
##  $ country_id                 : num [1:6789] 3 3 3 3 3 3 3 3 3 3 ...
```

```
##  $ year                        : num [1:6789] 1984 1985 1986 1987 1988 ...
##  $ historical_date             : Date[1:6789], format: "1984-12-31" "1985-12-31" ...
##  $ project                     : num [1:6789] 0 0 0 0 0 0 0 0 0 0 ...
##  $ historical                  : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ histname                    : chr [1:6789] "United Mexican States" "United Mexican States" "United
##  $ codingstart                 : num [1:6789] 1789 1789 1789 1789 1789 ...
##  $ codingend                   : num [1:6789] 2022 2022 2022 2022 2022 ...
##  $ codingstart_contemp         : num [1:6789] 1900 1900 1900 1900 1900 1900 1900 1900 1900 1900 ...
##  $ codingend_contemp           : num [1:6789] 2022 2022 2022 2022 2022 ...
##  $ codingstart_hist            : num [1:6789] 1789 1789 1789 1789 1789 ...
##  $ codingend_hist              : num [1:6789] 1920 1920 1920 1920 1920 1920 1920 1920 1920 1920 ...
##  $ gapstart1                   : num [1:6789] NA NA NA NA NA NA NA NA NA NA ...
##  $ gapstart2                   : num [1:6789] NA NA NA NA NA NA NA NA NA NA ...
##  $ gapstart3                   : num [1:6789] NA NA NA NA NA NA NA NA NA NA ...
##  $ gapend1                     : num [1:6789] NA NA NA NA NA NA NA NA NA NA ...
##  $ gapend2                     : num [1:6789] NA NA NA NA NA NA NA NA NA NA ...
##  $ gapend3                     : num [1:6789] NA NA NA NA NA NA NA NA NA NA ...
##  $ gap_index                   : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ COWcode                     : num [1:6789] 70 70 70 70 70 70 70 70 70 70 ...
##  $ e_v2x_api_3C                : num [1:6789] NA NA NA NA 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ e_v2x_api_4C                : num [1:6789] 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0
##  $ e_v2x_api_5C                : num [1:6789] 0.5 0.5 0.5 0.5 0.75 0.75 0.75 0.75 0.75 0.75 ...
##  $ e_v2x_civlib_3C             : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ e_v2x_civlib_4C             : num [1:6789] 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0
##  $ e_v2x_civlib_5C             : num [1:6789] 0.5 0.5 0.5 0.5 0.5 0.75 0.75 0.75 0.75 0.75 ...
##  $ e_v2x_clphy_3C              : num [1:6789] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 1 1 1 ...
##  $ e_v2x_clphy_4C              : num [1:6789] 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0.667 0.667 0
##  $ e_v2x_clphy_5C              : num [1:6789] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ e_v2x_clpol_3C              : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ e_v2x_clpol_4C              : num [1:6789] 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0
##  $ e_v2x_clpol_5C              : num [1:6789] 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 ...
##  $ e_v2x_clpriv_3C             : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ e_v2x_clpriv_4C             : num [1:6789] 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 1
##  $ e_v2x_clpriv_5C             : num [1:6789] 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 ...
##  $ e_v2x_corr_3C               : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ e_v2x_corr_4C               : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ e_v2x_corr_5C               : num [1:6789] 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 ...
##  $ e_v2x_cspart_3C             : num [1:6789] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ e_v2x_cspart_4C             : num [1:6789] 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0
##  $ e_v2x_cspart_5C             : num [1:6789] 0.25 0.25 0.25 0.25 0.25 0.5 0.25 0.25 0.25 0.25 ...
##  $ e_v2x_delibdem_3C           : num [1:6789] 0 0 0 0 0 0.5 0.5 0.5 0.5 0.5 ...
##  $ e_v2x_delibdem_4C           : num [1:6789] 0 0 0 0 0 0.333 0.333 0.333 0.333 0.333 ...
##  $ e_v2x_delibdem_5C           : num [1:6789] 0 0 0 0.25 0.25 0.25 0.25 0.25 0.25 0.25 ...
##  $ e_v2x_EDcomp_thick_3C       : num [1:6789] 0 0 0 0 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ e_v2x_EDcomp_thick_4C       : num [1:6789] 0 0 0 0 0.333 0.333 0.333 0.667 0.667 0.667 ...
##  $ e_v2x_EDcomp_thick_5C       : num [1:6789] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ e_v2x_egal_3C               : num [1:6789] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ e_v2x_egal_4C               : num [1:6789] 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0
##  $ e_v2x_egal_5C               : num [1:6789] 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 ...
##  $ e_v2x_egaldem_3C            : num [1:6789] 0 0 0 0 0 0 0 0 0 0 ...
##  $ e_v2x_egaldem_4C            : num [1:6789] 0 0 0 0 0 0 0 0 0 0 ...
##  $ e_v2x_egaldem_5C            : num [1:6789] 0 0 0 0 0 0 0 0.25 0.25 0.25 ...
##  $ e_v2x_elecoff_3C            : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ e_v2x_elecoff_4C            : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ e_v2x_elecoff_5C          : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ e_v2x_execorr_3C          : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ e_v2x_execorr_4C          : num [1:6789] 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0
##  $ e_v2x_execorr_5C          : num [1:6789] 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 ...
##  $ e_v2x_feduni_3C           : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ e_v2x_feduni_4C           : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ e_v2x_feduni_5C           : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ e_v2x_frassoc_thick_3C    : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ e_v2x_frassoc_thick_4C    : num [1:6789] 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0
##  $ e_v2x_frassoc_thick_5C    : num [1:6789] 0.5 0.5 0.5 0.75 0.75 0.75 0.75 0.75 0.75 0.75 ...
##  $ e_v2x_freexp_3C           : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ e_v2x_freexp_4C           : num [1:6789] 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0
##  $ e_v2x_freexp_5C           : num [1:6789] 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 ...
##  $ e_v2x_freexp_altinf_3C    : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ e_v2x_freexp_altinf_4C    : num [1:6789] 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0
##  $ e_v2x_freexp_altinf_5C    : num [1:6789] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.75 0.75 0.75 ...
##  $ e_v2x_gencl_3C            : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ e_v2x_gencl_4C            : num [1:6789] 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0
##  $ e_v2x_gencl_5C            : num [1:6789] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ e_v2x_gencs_3C            : num [1:6789] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 1 ...
##  $ e_v2x_gencs_4C            : num [1:6789] 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0
##  $ e_v2x_gencs_5C            : num [1:6789] 0.25 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ e_v2x_gender_3C           : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ e_v2x_gender_4C           : num [1:6789] 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0
##  $ e_v2x_gender_5C           : num [1:6789] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ e_v2x_genpp_3C            : num [1:6789] 1 1 1 1 1 1 1 1 1 1 ...
##  $ e_v2x_genpp_4C            : num [1:6789] 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0.667 0
##  $ e_v2x_genpp_5C            : num [1:6789] 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.5 0.5 0.5 ...
##  $ e_v2x_jucon_3C            : num [1:6789] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ e_v2x_jucon_4C            : num [1:6789] 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0
##  $ e_v2x_jucon_5C            : num [1:6789] 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 ...
##  $ e_v2x_libdem_3C           : num [1:6789] 0 0 0 0 0 0 0 0 0 0 ...
##  $ e_v2x_libdem_4C           : num [1:6789] 0 0 0 0 0 0 0 0 0 0 ...
##  $ e_v2x_libdem_5C           : num [1:6789] 0 0 0 0 0 0 0 0.25 0.25 0.25 ...
##  $ e_v2x_liberal_3C          : num [1:6789] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ e_v2x_liberal_4C          : num [1:6789] 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0
##  $ e_v2x_liberal_5C          : num [1:6789] 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 ...
##  $ e_v2x_mpi_3C              : num [1:6789] 0 0 0 0 0 0 0 0 0 0 ...
##  $ e_v2x_mpi_4C              : num [1:6789] 0 0 0 0 0 0 0 0 0 0 ...
##  $ e_v2x_mpi_5C              : num [1:6789] 0 0 0 0 0 0 0 0 0 0.25 ...
##  $ e_v2x_partip_3C           : num [1:6789] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ e_v2x_partip_4C           : num [1:6789] 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0.333 0
##   [list output truncated]
##  - attr(*, "spec")=
##   .. cols(
##   ..   country_name = col_character(),
##   ..   country_text_id = col_character(),
##   ..   country_id = col_double(),
##   ..   year = col_double(),
##   ..   historical_date = col_date(format = ""),
##   ..   project = col_double(),
##   ..   historical = col_double(),
##   ..   histname = col_character(),
##   ..   codingstart = col_double(),
```

```
##   ..   codingend = col_double(),
##   ..   codingstart_contemp = col_double(),
##   ..   codingend_contemp = col_double(),
##   ..   codingstart_hist = col_double(),
##   ..   codingend_hist = col_double(),
##   ..   gapstart1 = col_double(),
##   ..   gapstart2 = col_double(),
##   ..   gapstart3 = col_double(),
##   ..   gapend1 = col_double(),
##   ..   gapend2 = col_double(),
##   ..   gapend3 = col_double(),
##   ..   gap_index = col_double(),
##   ..   COWcode = col_double(),
##   ..   e_v2x_api_3C = col_double(),
##   ..   e_v2x_api_4C = col_double(),
##   ..   e_v2x_api_5C = col_double(),
##   ..   e_v2x_civlib_3C = col_double(),
##   ..   e_v2x_civlib_4C = col_double(),
##   ..   e_v2x_civlib_5C = col_double(),
##   ..   e_v2x_clphy_3C = col_double(),
##   ..   e_v2x_clphy_4C = col_double(),
##   ..   e_v2x_clphy_5C = col_double(),
##   ..   e_v2x_clpol_3C = col_double(),
##   ..   e_v2x_clpol_4C = col_double(),
##   ..   e_v2x_clpol_5C = col_double(),
##   ..   e_v2x_clpriv_3C = col_double(),
##   ..   e_v2x_clpriv_4C = col_double(),
##   ..   e_v2x_clpriv_5C = col_double(),
##   ..   e_v2x_corr_3C = col_double(),
##   ..   e_v2x_corr_4C = col_double(),
##   ..   e_v2x_corr_5C = col_double(),
##   ..   e_v2x_cspart_3C = col_double(),
##   ..   e_v2x_cspart_4C = col_double(),
##   ..   e_v2x_cspart_5C = col_double(),
##   ..   e_v2x_delibdem_3C = col_double(),
##   ..   e_v2x_delibdem_4C = col_double(),
##   ..   e_v2x_delibdem_5C = col_double(),
##   ..   e_v2x_EDcomp_thick_3C = col_double(),
##   ..   e_v2x_EDcomp_thick_4C = col_double(),
##   ..   e_v2x_EDcomp_thick_5C = col_double(),
##   ..   e_v2x_egal_3C = col_double(),
##   ..   e_v2x_egal_4C = col_double(),
##   ..   e_v2x_egal_5C = col_double(),
##   ..   e_v2x_egaldem_3C = col_double(),
##   ..   e_v2x_egaldem_4C = col_double(),
##   ..   e_v2x_egaldem_5C = col_double(),
##   ..   e_v2x_elecoff_3C = col_double(),
##   ..   e_v2x_elecoff_4C = col_double(),
##   ..   e_v2x_elecoff_5C = col_double(),
##   ..   e_v2x_execorr_3C = col_double(),
##   ..   e_v2x_execorr_4C = col_double(),
##   ..   e_v2x_execorr_5C = col_double(),
##   ..   e_v2x_feduni_3C = col_double(),
##   ..   e_v2x_feduni_4C = col_double(),
```

```
##   ..   e_v2x_feduni_5C = col_double(),
##   ..   e_v2x_frassoc_thick_3C = col_double(),
##   ..   e_v2x_frassoc_thick_4C = col_double(),
##   ..   e_v2x_frassoc_thick_5C = col_double(),
##   ..   e_v2x_freexp_3C = col_double(),
##   ..   e_v2x_freexp_4C = col_double(),
##   ..   e_v2x_freexp_5C = col_double(),
##   ..   e_v2x_freexp_altinf_3C = col_double(),
##   ..   e_v2x_freexp_altinf_4C = col_double(),
##   ..   e_v2x_freexp_altinf_5C = col_double(),
##   ..   e_v2x_gencl_3C = col_double(),
##   ..   e_v2x_gencl_4C = col_double(),
##   ..   e_v2x_gencl_5C = col_double(),
##   ..   e_v2x_gencs_3C = col_double(),
##   ..   e_v2x_gencs_4C = col_double(),
##   ..   e_v2x_gencs_5C = col_double(),
##   ..   e_v2x_gender_3C = col_double(),
##   ..   e_v2x_gender_4C = col_double(),
##   ..   e_v2x_gender_5C = col_double(),
##   ..   e_v2x_genpp_3C = col_double(),
##   ..   e_v2x_genpp_4C = col_double(),
##   ..   e_v2x_genpp_5C = col_double(),
##   ..   e_v2x_jucon_3C = col_double(),
##   ..   e_v2x_jucon_4C = col_double(),
##   ..   e_v2x_jucon_5C = col_double(),
##   ..   e_v2x_libdem_3C = col_double(),
##   ..   e_v2x_libdem_4C = col_double(),
##   ..   e_v2x_libdem_5C = col_double(),
##   ..   e_v2x_liberal_3C = col_double(),
##   ..   e_v2x_liberal_4C = col_double(),
##   ..   e_v2x_liberal_5C = col_double(),
##   ..   e_v2x_mpi_3C = col_double(),
##   ..   e_v2x_mpi_4C = col_double(),
##   ..   e_v2x_mpi_5C = col_double(),
##   ..   e_v2x_partip_3C = col_double(),
##   ..   e_v2x_partip_4C = col_double(),
##   ..   e_v2x_partip_5C = col_double(),
##   ..   e_v2x_partipdem_3C = col_double(),
##   ..   e_v2x_partipdem_4C = col_double(),
##   ..   e_v2x_partipdem_5C = col_double(),
##   ..   e_v2x_polyarchy_3C = col_double(),
##   ..   e_v2x_polyarchy_4C = col_double(),
##   ..   e_v2x_polyarchy_5C = col_double(),
##   ..   e_v2x_pubcorr_3C = col_double(),
##   ..   e_v2x_pubcorr_4C = col_double(),
##   ..   e_v2x_pubcorr_5C = col_double(),
##   ..   e_v2x_suffr_3C = col_double(),
##   ..   e_v2x_suffr_4C = col_double(),
##   ..   e_v2x_suffr_5C = col_double(),
##   ..   e_v2xcl_rol_3C = col_double(),
##   ..   e_v2xcl_rol_4C = col_double(),
##   ..   e_v2xcl_rol_5C = col_double(),
##   ..   e_v2xcs_ccsi_3C = col_double(),
##   ..   e_v2xcs_ccsi_4C = col_double(),
```

```
##   ..   e_v2xcs_ccsi_5C = col_double(),
##   ..   e_v2xdd_dd_3C = col_double(),
##   ..   e_v2xdd_dd_4C = col_double(),
##   ..   e_v2xdd_dd_5C = col_double(),
##   ..   e_v2xdl_delib_3C = col_double(),
##   ..   e_v2xdl_delib_4C = col_double(),
##   ..   e_v2xdl_delib_5C = col_double(),
##   ..   e_v2xeg_eqdr_3C = col_double(),
##   ..   e_v2xeg_eqdr_4C = col_double(),
##   ..   e_v2xeg_eqdr_5C = col_double(),
##   ..   e_v2xeg_eqprotec_3C = col_double(),
##   ..   e_v2xeg_eqprotec_4C = col_double(),
##   ..   e_v2xeg_eqprotec_5C = col_double(),
##   ..   e_v2xel_frefair_3C = col_double(),
##   ..   e_v2xel_frefair_4C = col_double(),
##   ..   e_v2xel_frefair_5C = col_double(),
##   ..   e_v2xel_locelec_3C = col_double(),
##   ..   e_v2xel_locelec_4C = col_double(),
##   ..   e_v2xel_locelec_5C = col_double(),
##   ..   e_v2xel_regelec_3C = col_double(),
##   ..   e_v2xel_regelec_4C = col_double(),
##   ..   e_v2xel_regelec_5C = col_double(),
##   ..   e_v2xlg_legcon_3C = col_double(),
##   ..   e_v2xlg_legcon_4C = col_double(),
##   ..   e_v2xlg_legcon_5C = col_double(),
##   ..   e_v2xme_altinf_3C = col_double(),
##   ..   e_v2xme_altinf_4C = col_double(),
##   ..   e_v2xme_altinf_5C = col_double(),
##   ..   e_v2xps_party_3C = col_double(),
##   ..   e_v2xps_party_4C = col_double(),
##   ..   e_v2xps_party_5C = col_double(),
##   ..   e_boix_regime = col_double(),
##   ..   e_democracy_breakdowns = col_double(),
##   ..   e_democracy_omitteddata = col_double(),
##   ..   e_democracy_trans = col_double(),
##   ..   e_fh_cl = col_double(),
##   ..   e_fh_pr = col_double(),
##   ..   e_fh_rol = col_double(),
##   ..   e_fh_status = col_double(),
##   ..   e_wbgi_cce = col_double(),
##   ..   e_wbgi_gee = col_double(),
##   ..   e_wbgi_pve = col_double(),
##   ..   e_wbgi_rle = col_double(),
##   ..   e_wbgi_rqe = col_double(),
##   ..   e_wbgi_vae = col_double(),
##   ..   e_lexical_index = col_double(),
##   ..   e_uds_median = col_double(),
##   ..   e_uds_mean = col_double(),
##   ..   e_uds_pct025 = col_double(),
##   ..   e_uds_pct975 = col_double(),
##   ..   e_coups = col_double(),
##   ..   e_legparty = col_double(),
##   ..   e_autoc = col_double(),
##   ..   e_democ = col_double(),
```

```
##   ..   e_p_polity = col_double(),
##   ..   e_polcomp = col_double(),
##   ..   e_polity2 = col_double(),
##   ..   e_bnr_dem = col_double(),
##   ..   e_chga_demo = col_double(),
##   ..   e_ti_cpi = col_double(),
##   ..   e_vanhanen = col_double(),
##   ..   e_peaveduc = col_double(),
##   ..   e_peedgini = col_double(),
##   ..   e_area = col_double(),
##   ..   e_regiongeo = col_double(),
##   ..   e_regionpol = col_double(),
##   ..   e_regionpol_6C = col_double(),
##   ..   e_cow_exports = col_double(),
##   ..   e_cow_imports = col_double(),
##   ..   e_gdp = col_double(),
##   ..   e_gdp_sd = col_double(),
##   ..   e_gdppc = col_double(),
##   ..   e_gdppc_sd = col_double(),
##   ..   e_miinflat = col_double(),
##   ..   e_pop = col_double(),
##   ..   e_pop_sd = col_double(),
##   ..   e_total_fuel_income_pc = col_double(),
##   ..   e_total_oil_income_pc = col_double(),
##   ..   e_total_resources_income_pc = col_double(),
##   ..   e_radio_n = col_double(),
##   ..   e_miferrat = col_double(),
##   ..   e_mipopula = col_double(),
##   ..   e_miurbani = col_double(),
##   ..   e_miurbpop = col_double(),
##   ..   e_pefeliex = col_double(),
##   ..   e_peinfmor = col_double(),
##   ..   e_pelifeex = col_double(),
##   ..   e_pematmor = col_double(),
##   ..   e_wb_pop = col_double(),
##   ..   e_civil_war = col_double(),
##   ..   e_miinteco = col_double(),
##   ..   e_miinterc = col_double(),
##   ..   e_pt_coup = col_double(),
##   ..   e_pt_coup_attempts = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Usually, the second step of my data inquiry is having an overview of the *identifiers* of data points. In our case, the identifiers are country names, country IDs, and years. Using the `distinct()` function can effectively identify the distinct levels of *identifiers*

```
d |> select(country_name, country_id, year) |> distinct()
```

```
## # A tibble: 6,789 x 3
##    country_name country_id  year
##    <chr>             <dbl> <dbl>
##  1 Mexico                3  1984
```

```
##   2 Mexico                  3  1985
##   3 Mexico                  3  1986
##   4 Mexico                  3  1987
##   5 Mexico                  3  1988
##   6 Mexico                  3  1989
##   7 Mexico                  3  1990
##   8 Mexico                  3  1991
##   9 Mexico                  3  1992
##  10 Mexico                  3  1993
## # ... with 6,779 more rows
```

```
# Which countries are in this dataset
d |> select(country_name) |> distinct()
```

```
## # A tibble: 181 x 1
##    country_name
##    <chr>
##  1 Mexico
##  2 Suriname
##  3 Sweden
##  4 Switzerland
##  5 Ghana
##  6 South Africa
##  7 Japan
##  8 Burma/Myanmar
##  9 Russia
## 10 Albania
## # ... with 171 more rows
```

```
d |> select(year) |> distinct()
```

```
## # A tibble: 39 x 1
##     year
##    <dbl>
##  1  1984
##  2  1985
##  3  1986
##  4  1987
##  5  1988
##  6  1989
##  7  1990
##  8  1991
##  9  1992
## 10  1993
## # ... with 29 more rows
```

Select both the country identifiers, GDP, and GDP per capita.

```
d_gdp <- d |>
  select(country_name, country_id, year, e_gdp, e_gdppc)

d_gdp
```

```
## # A tibble: 6,789 x 5
##    country_name country_id  year   e_gdp e_gdppc
##    <chr>             <dbl> <dbl>   <dbl>   <dbl>
##  1 Mexico                3  1984  93563.    11.7
##  2 Mexico                3  1985  94259.    11.5
##  3 Mexico                3  1986  92750.    11.1
##  4 Mexico                3  1987  93220.    10.9
##  5 Mexico                3  1988  94687.    10.8
##  6 Mexico                3  1989  98145.    11.0
##  7 Mexico                3  1990 103254.    11.4
##  8 Mexico                3  1991 107374.    11.6
##  9 Mexico                3  1992 111533.    11.9
## 10 Mexico                3  1993 114611.    12.0
## # ... with 6,779 more rows
```

3. Rename Columns to Make Names Informative

```r
d_gdp <- d_gdp |>
  rename("GDP" = "e_gdp", "GDP_per_capita" = "e_gdppc",
         "Country" = "country_name", "ID" = "country_id",
         "Year" = "year")

d_gdp
```

```
## # A tibble: 6,789 x 5
##    Country    ID  Year     GDP GDP_per_capita
##    <chr>   <dbl> <dbl>   <dbl>          <dbl>
##  1 Mexico      3  1984  93563.           11.7
##  2 Mexico      3  1985  94259.           11.5
##  3 Mexico      3  1986  92750.           11.1
##  4 Mexico      3  1987  93220.           10.9
##  5 Mexico      3  1988  94687.           10.8
##  6 Mexico      3  1989  98145.           11.0
##  7 Mexico      3  1990 103254.           11.4
##  8 Mexico      3  1991 107374.           11.6
##  9 Mexico      3  1992 111533.           11.9
## 10 Mexico      3  1993 114611.           12.0
## # ... with 6,779 more rows
```

4. Subset Rows of the Data Using `slice_`

The set of `slice_` functions will become handy when you want to take a subset of rows based on some simple rules.

If you would like to get 10 obervations (countries-years) with the maximum `GDP`, use `slice_max`:

```r
# Want countries-years with highest GDP
d_gdp |> slice_max(order_by = GDP, n = 10)
```

```
## # A tibble: 10 x 5
##    Country                          ID   Year      GDP GDP_per_capita
```

```
##    <chr>                    <dbl> <dbl>    <dbl>          <dbl>
##  1 China                      110  2019 2279809.           15.4
##  2 China                      110  2018 2205730.           14.9
##  3 China                      110  2017 2136176.           14.5
##  4 United States of America    20  2019 2118706.           60.6
##  5 United States of America    20  2018 2077898.           59.6
##  6 China                      110  2016 2039529.           13.9
##  7 United States of America    20  2017 2023242.           58.5
##  8 United States of America    20  2016 1980809.           57.6
##  9 China                      110  2015 1953127.           13.3
## 10 United States of America    20  2015 1942092.           56.7
```

Similiarily, if you want a subset of countries-years with mimnimal GDP, use `slice_min`:

```r
# Get countries-years with the lowest GDP
d_gdp |> slice_min(order_by = GDP, n = 10)
```

```
## # A tibble: 10 x 5
##    Country                  ID  Year   GDP GDP_per_capita
##    <chr>                 <dbl> <dbl> <dbl>          <dbl>
##  1 Sao Tome and Principe   196  1988  24.0           2.04
##  2 Sao Tome and Principe   196  1987  24.0           2.08
##  3 Sao Tome and Principe   196  1986  24.4           2.17
##  4 Sao Tome and Principe   196  1984  24.7           2.29
##  5 Sao Tome and Principe   196  1985  24.9           2.26
##  6 Sao Tome and Principe   196  1989  25.0           2.06
##  7 Sao Tome and Principe   196  1990  25.2           2.03
##  8 Sao Tome and Principe   196  1992  25.2           1.95
##  9 Sao Tome and Principe   196  1991  25.3           1.99
## 10 Sao Tome and Principe   196  1993  25.5           1.93
```

Finally, if you wish to take a random sample of observations in the data, use `slice_sample`. Note that you may tell R the exact sample size you want:

```r
set.seed(52)
d_gdp |> slice_sample(n = 10) # Sample 10 observations
```

```
## # A tibble: 10 x 5
##    Country         ID  Year     GDP GDP_per_capita
##    <chr>        <dbl> <dbl>   <dbl>          <dbl>
##  1 Cape Verde      70  1988    76.5           2.18
##  2 Oman           187  1991   2955.          14.7
##  3 Romania        190  2010  30202.          14.0
##  4 South Korea     42  2001 124701.          24.6
##  5 Mozambique      57  2012   3589.           1.41
##  6 Bulgaria       152  1992   8739.           9.53
##  7 Morocco         90  2001  15549.           5.03
##  8 Vietnam         34  1990  10537.           1.47
##  9 Canada          66  1985  83713.          30.4
## 10 Serbia         198  1987  17430.           7.64
```

Or you may define the sample size as a poroportion of the original data size:

```r
set.seed(52)
d_gdp |> slice_sample(prop = 0.1)
```

```
## # A tibble: 678 x 5
##    Country        ID  Year     GDP GDP_per_capita
##    <chr>       <dbl> <dbl>   <dbl>          <dbl>
##  1 Cape Verde     70  1988    76.5           2.18
##  2 Oman          187  1991   2955.          14.7
##  3 Romania       190  2010  30202.          14.0
##  4 South Korea    42  2001 124701.          24.6
##  5 Mozambique     57  2012   3589.           1.41
##  6 Bulgaria      152  1992   8739.           9.53
##  7 Morocco        90  2001  15549.           5.03
##  8 Vietnam        34  1990  10537.           1.47
##  9 Canada         66  1985  83713.          30.4
## 10 Serbia        198  1987  17430.           7.64
## # ... with 668 more rows
```

The `set.seed` function specify a random seed with which the system uses to generate the "random sample." Long story short, "random" stuff generated by a machine are never really random. Instead, the random outputs (in our case, a random subset of the data) are results of the computer input some "random seed" to some complicated formula. When you define a random seed, you can guarantee that you obtain the same random sample every time you run the program – this makes your data science research reproducible. As we have discussed, reproducibility is a desired feature of a data science project. So I would strongly recommend setting a random seed every time.

5. Subset Rows of the Data Using `filter`

For example, we may take the observations whose `Year` variable ranges from 2000 to 2005.

```r
# Want: 2000-2005 data
d_gdp |> filter(Year >= 2000 & Year <= 2005)
```

```
## # A tibble: 1,062 x 5
##    Country     ID  Year     GDP GDP_per_capita
##    <chr>    <dbl> <dbl>   <dbl>          <dbl>
##  1 Mexico       3  2000 145206.          13.7
##  2 Mexico       3  2001 146993.          13.6
##  3 Mexico       3  2002 148549.          13.6
##  4 Mexico       3  2003 151035.          13.7
##  5 Mexico       3  2004 156578.          14.1
##  6 Mexico       3  2005 162094.          14.3
##  7 Suriname     4  2000    383.           7.67
##  8 Suriname     4  2001    402.           7.93
##  9 Suriname     4  2002    423.           8.25
## 10 Suriname     4  2003    451.           8.67
## # ... with 1,052 more rows
```

We may subset observations whose `Country` variable, a `character` variable, equals to the text `"China"`.

```r
d_gdp |> filter(Country == "China")
```

```
## # A tibble: 39 x 5
##    Country     ID  Year     GDP GDP_per_capita
##    <chr>    <dbl> <dbl>   <dbl>          <dbl>
##  1 China      110  1984 243976.           2.21
##  2 China      110  1985 265805.           2.36
##  3 China      110  1986 285707.           2.50
##  4 China      110  1987 308227.           2.65
##  5 China      110  1988 322596.           2.73
##  6 China      110  1989 327739.           2.74
##  7 China      110  1990 315683.           2.63
##  8 China      110  1991 329836.           2.71
##  9 China      110  1992 359817.           2.90
## 10 China      110  1993 393449.           3.15
## # ... with 29 more rows
```

We may also stack multiple `filter` functions. For example, you may do the following if you want to look at a subset of the data whose `Year` ranges from 2000 to 2005 and `Country` equals to `"China"`:

```r
# Want: 2000 - 2005 from China
d_gdp |>
  filter(Year >= 2000 & Year <= 2005) |>
  filter(Country == "China")
```

```
## # A tibble: 6 x 5
##   Country     ID  Year     GDP GDP_per_capita
##   <chr>    <dbl> <dbl>   <dbl>          <dbl>
## 1 China      110  2000 633740.           4.74
## 2 China      110  2001 682141.           5.05
## 3 China      110  2002 738393.           5.43
## 4 China      110  2003 798702.           5.83
## 5 China      110  2004 871314.           6.31
## 6 China      110  2005 956102.           6.89
```

6. Sort the Data based on Values of Rows using `arrange`

Now we will try to sort the dataset `d_gdp` by the value of GDP per capita using the `arrange`. We may have country-year with small values of `GDP_per_capita` appearing first and those with larger values of `GDP_per_capita` coming after them.

```r
# Want: sort the row by GDP per capita
d_gdp |> arrange(GDP_per_capita)
```

```
## # A tibble: 6,789 x 5
##    Country                              ID  Year   GDP GDP_per_capita
##    <chr>                             <dbl> <dbl> <dbl>          <dbl>
##  1 Liberia                              86  1995  62.3          0.286
##  2 Liberia                              86  1994  65.5          0.307
##  3 Liberia                              86  1996  70.6          0.309
##  4 Liberia                              86  1993  81.5          0.383
```

```
##  5 Liberia                            86  1997  107.            0.429
##  6 Liberia                            86  1992  113.            0.53
##  7 Democratic Republic of the Congo  111  2002 2966.            0.538
##  8 Democratic Republic of the Congo  111  2001 2890.            0.54
##  9 Liberia                            86  1998  147.            0.543
## 10 Democratic Republic of the Congo  111  2003 3141.            0.552
## # ... with 6,779 more rows
```

Want the countries-years with larger values of `GDP_per_capita` appear first? Simply reverse the value using `-GDP_per_capita`. Alternatively, you may replace `desc(GDP_per_capita)`.

```
d_gdp |> arrange(-GDP_per_capita)
```

```
## # A tibble: 6,789 x 5
##    Country                 ID  Year    GDP GDP_per_capita
##    <chr>                <dbl> <dbl>  <dbl>          <dbl>
##  1 United Arab Emirates   207  1984 16817.          115.
##  2 United Arab Emirates   207  1985 15946.          103.
##  3 Qatar                   94  2012 23055.          101.
##  4 Qatar                   94  2011 21273.          100.
##  5 Qatar                   94  2013 24074.           98.9
##  6 United Arab Emirates   207  1991 20567.           96.5
##  7 United Arab Emirates   207  1992 21506.           95.7
##  8 Qatar                   94  2014 24194.           95.3
##  9 Qatar                   94  2010 18107.           94.4
## 10 United Arab Emirates   207  2000 31871.           93.3
## # ... with 6,779 more rows
```

7. Perform (4) (5) (6) group by group: `group_by`, `ungroup`

Task: Create a dataset that contains the year of the higest development level for each country/ region respectively.

1. Perform a data availability/ integrity check. Then aggregate the data into a new country-level dataset which contains the following indicators:
   1. Average development level from 1984 to 2022.
   2. Magnitude of growth from 1984 to 2022.

```
# INSERT CODE HERE
```

8. Create new columns in the data: `group_by`, `mutate`, `ungroup`

Task: Add the following economic indicators to the data:

1. Country-year development level with reference to that of 1984.
2. Year-on-year economic growth.

```
# INSERT CODE HERE
```

9. Summarize the data: `group_by`, `summarise`, `ungroup`

Task: Perform a data availability/ integrity check. Then aggregate the data into a new country-level dataset which contains the following indicators:

1. Average development level from 1984 to 2022.
2. Magnitude of growth from 1984 to 2022.

```r
# INSERT CODE HERE
```

Final Notes

Pipe `|>`

What is a pipe?

> R now provides a simple native forward pipe syntax `|>`. The simple form of the forward pipe inserts the left-hand side as the first argument in the right-hand side call.

Let's elaborate this definition

```r
# What we have used
d_gdp |> filter(Country == "China")
```

```
## # A tibble: 39 x 5
##    Country    ID  Year    GDP GDP_per_capita
##    <chr>   <dbl> <dbl>  <dbl>          <dbl>
##  1 China     110  1984 243976.          2.21
##  2 China     110  1985 265805.          2.36
##  3 China     110  1986 285707.          2.50
##  4 China     110  1987 308227.          2.65
##  5 China     110  1988 322596.          2.73
##  6 China     110  1989 327739.          2.74
##  7 China     110  1990 315683.          2.63
##  8 China     110  1991 329836.          2.71
##  9 China     110  1992 359817.          2.90
## 10 China     110  1993 393449.          3.15
## # ... with 29 more rows
```

```r
# is equivalent to...
filter(d_gdp, Country == "China")
```

```
## # A tibble: 39 x 5
##    Country    ID  Year    GDP GDP_per_capita
##    <chr>   <dbl> <dbl>  <dbl>          <dbl>
##  1 China     110  1984 243976.          2.21
##  2 China     110  1985 265805.          2.36
##  3 China     110  1986 285707.          2.50
##  4 China     110  1987 308227.          2.65
##  5 China     110  1988 322596.          2.73
```

```
##  6 China     110  1989 327739.           2.74
##  7 China     110  1990 315683.           2.63
##  8 China     110  1991 329836.           2.71
##  9 China     110  1992 359817.           2.90
## 10 China     110  1993 393449.           3.15
## # ... with 29 more rows
```

```r
# ... is equivalent to
d_gdp |> filter(.data = _, Country == "China")
```

```
## # A tibble: 39 x 5
##    Country    ID  Year     GDP GDP_per_capita
##    <chr>   <dbl> <dbl>   <dbl>          <dbl>
##  1 China     110  1984 243976.           2.21
##  2 China     110  1985 265805.           2.36
##  3 China     110  1986 285707.           2.50
##  4 China     110  1987 308227.           2.65
##  5 China     110  1988 322596.           2.73
##  6 China     110  1989 327739.           2.74
##  7 China     110  1990 315683.           2.63
##  8 China     110  1991 329836.           2.71
##  9 China     110  1992 359817.           2.90
## 10 China     110  1993 393449.           3.15
## # ... with 29 more rows
```

```r
# Note: You may use "_" as a placeholder of the object passed down through the pipe. But it should be u
```

Why piping? Pipe is useful when you are conducting a series of operation on your data but want to minimize the number of intermediate outputs produced. To

```r
# STEP 1: Subset variables
d_gdp <- d |> select(country_name, country_id, year, e_gdp, e_gdppc)

# STEP 2: Rename variables
d_gdp_renamed <- d_gdp |>
  rename("GDP" = "e_gdp", "GDP_per_capita" = "e_gdppc",
         "Country" = "country_name", "ID" = "country_id",
         "Year" = "year")

# STEP 3: Filter down to China
d_gdp_china <- d_gdp_renamed |> filter(Country == "China")

# STEP 4: Filter down to 2000 - 2005
d_gdp_china_2000_2005 <- d_gdp_china |> filter(Year >= 2000 & Year <= 2005)

d_gdp_china_2000_2005
```

```
## # A tibble: 6 x 5
##   Country    ID  Year     GDP GDP_per_capita
##   <chr>   <dbl> <dbl>   <dbl>          <dbl>
## 1 China     110  2000 633740.           4.74
## 2 China     110  2001 682141.           5.05
```

```
## 3 China     110  2002 738393.           5.43
## 4 China     110  2003 798702.           5.83
## 5 China     110  2004 871314.           6.31
## 6 China     110  2005 956102.           6.89
```

As programmers, we face trade-offs. We want to work things out step-by-step. In this way, our code will look organized and readable for ourselves and other readers. However, the cost of a a step-by-step approach often is the growing size of intermediate outputs — to pass down results from our intermediate steps, we have to temporarily save intermediate outputs. Doing so consume system resources (as they take up your Memory), makes it hard to navigate through your Environment, and is error-prone.

A pipe helps us maintain the step-by-step approach without creating many intermediate outputs. In our case, I can skip the intermediate outputs `d_gdp`, `d_gdp_renamed` and `d_gdp_china` with pipe.

```
rm(d_gdp, d_gdp_renamed, d_gdp_china, d_gdp_china_2000_2005)

d_gdp_china_2000_2005 <- d |>
  # Subset variables
  select(country_name, country_id, year, e_gdp, e_gdppc) |>
  # Rename variables
  rename("GDP" = "e_gdp", "GDP_per_capita" = "e_gdppc",
         "Country" = "country_name", "ID" = "country_id",
         "Year" = "year") |>
  # Filter only observations from China
  filter(Country == "China") |>
  # Filter 2000 - 2005
  filter(Year >= 2000 & Year <= 2005)

d_gdp_china_2000_2005
```

```
## # A tibble: 6 x 5
##   Country    ID  Year     GDP GDP_per_capita
##   <chr>   <dbl> <dbl>   <dbl>          <dbl>
## 1 China     110  2000 633740.           4.74
## 2 China     110  2001 682141.           5.05
## 3 China     110  2002 738393.           5.43
## 4 China     110  2003 798702.           5.83
## 5 China     110  2004 871314.           6.31
## 6 China     110  2005 956102.           6.89
```

`|>` v.s. `%>%` When you look up online resources, you may see pipe written in a different way: `%>%`. This is the pipe operator that data scientists (including myself) have been familiar with for years. You may use `|>` and `%>%` interchangeably for basic use cases (which is pretty much everything we are doing in this course). For more advanced use cases, `%>%` is more powerful.

Read further: https://www.tidyverse.org/blog/2023/04/base-vs-magrittr-pipe/

To Create a New Object or Not

With pipe `|>`, we can maintain a step-by-step approach but skip some intermediate outputs. However, as our data processing task becomes more and more complicated, intermediate outputs are unavoidable. When do we want to create an intermediate output and when do we want to skip it? This is more arts than science. Here is my take:

1. If I keep repeating some data wrangling steps for many downstream tasks, I would create an intermediate output and use it for all these downstream tasks.
2. If I find some intermediate outputs no longer needed, I will remove them from my environment using `rm()` to keep my environment clean and to make space.
3. Although I'd plan my data wrangling before I start the work, unexpected things happen. Sometimes, I figure I can merge some data wrangling steps to reduce intermediate outputs. Sometime, I suddenly realize some intermediate outputs are essential. This is a trial-and-error process.
4. Perfectionism is unnecessary in data wrangling. Produce replicable code that you and readers can understand. But do not edit your code to make it "pretty" endlessly.

Style

Where should you add *a space*? Where should you add *a line break*? Where should you add *a comment*? Where should you add *a section break*? These are questions concerning the *style* of your R code. Like writing articles, maintaining a good style when you write code helps you better communicate information with your readers and your future self.

Before talking about style, I should stress that the correctness of your syntax should always be prioritized over style. **One common mistake beginners make is to add spaces between functions and their arguments**. The typical error along this line is adding a space between a function and its arguments. For example, `filter(Year >= 2000 & Year <= 2005)` is correct, but `filter (Year >= 2000 & Year <= 2005)` is incorrect. The latter has a space between the function `filter` and its arguments `(Year >= 2000 & Year <= 2005)`. Take another example, `x[, 1]`, an expression that subset the first column of the data frame or matrix `x`, is correct. But `x [, 1]` is incorrect, because a space is added between the object `x` and the command that takes a subset from it `[, 1]`.

As we heavily use `tidyverse`, we will use `tidyverse`'s style guide: https://style.tidyverse.org/. The style guide touches upon several advanced R functionality, for now, we will focus on Sections 1 (Files), 4 (Pipes), and 5(`ggplot2`).