SPEC Lab R Resources: Data Management I with tidyverse – Groupwork

Alix Ziff, Jasmine Chu, Annie Street, Yuchen Gong, Claudia Salas Gimenez, Ben Graham

Summer 2024

This groupwork assignment serves as your first practical experience in data management using the `tidyverse` and `dplyr` packages. In earlier materials, we introduced essential functions and techniques for managing data, such as selecting variables, filtering, and using piping for cleaner, more efficient code, and now you'll have the chance to apply those skills using a real-world dataset: the `IDC_training_2021.rds`, a country-year dataset compiled by the SPEC Lab.

Understanding the Dataset

Before diving into the coding exercises, it's important to understand the dataset and its context. Often, when learning to code, it can be overwhelming to work with a dataset from a research paper if you're not familiar with the paper itself—but that's completely normal! To help you navigate this, here are some steps to make the process easier.

Get to Know the Research Paper

Start by familiarizing yourself with the research question and goals of the study associated with the dataset. This will provide you with a broader understanding of why the data was collected and what it represents. Understanding the purpose of the study can help you better interpret the variables in your dataset and guide your analysis.

The `IDC_training_2021.rds` dataset is used in Graham, Benjamin A.T., Michael K. Miller, and Kaare W. Strøm. (2017). "Safeguarding Democracy: Powersharing and Democratic Survival." *American Political Science Review,* 111(4): 686–704.

Navigating the Codebook

The codebook is your go-to resource in the research paper's supplementary resources for understanding the specifics of the dataset. Think of it as a guide or a cheatsheet that explains how each variable is coded and what it represents. As you work through the data in R, keep the codebook handy to ensure you're correctly interpreting each variable.

Attached is the "Safeguarding Democracy: Powersharing and Democratic Survival" research paper, the codebook for better understanding the specifics of the `IDC_training_2021.rds` dataset. Now, let's get started!

Initial Setup

First, set up your workspace in R. Begin by writing a header for your R script and saving it in your personal folder. Set your working directory to the folder containing the training data, and load the necessary libraries: `tidyverse` and `readr`. The **`readr`** library in R is used for reading and writing data files efficiently. It provides fast functions for importing data from various file formats into R, such as:

- **read_csv()**: For reading comma-separated values (CSV) files.
- **read_tsv()**: For reading tab-separated values (TSV) files.
- **read_fwf()**: For reading fixed-width files.

```
# Set working directory
#setwd("YourFolderPath")

# Load required libraries
library(tidyverse)
library(readr)
```

Next, load the dataset. Since the `IDC_training_2021.rds` file is in `.rds` format, you'll need to use the `readRDS()` function to import it.

```
# Load the dataset
dt <- readRDS("IDC_training_2022.rds")

## This code saves the dataset as object 'dt'. You can name your data object
## whatever you like, but "dt" is nice and simple.
```

Exercise 1: Identify Variables

To become more familiar with the data, identify the second variable in the dataset from left to right. Also, what is the eighth variable?

```
# Identify 2nd and 8th variables in the dataset
colnames(dt[2]) ## The second variable in the dataset is "gwno"
```

```
## [1] "gwno"
```

```
colnames(dt[8])  ## The eighth variable in the dataset is "constsusp_IDC"
```

```
## [1] "constsusp_IDC"
```

Key Function Recap

Let's do a quick recap of some of the essential functions you'll be using throughout this assignment:

- `select()`: Keeps only the specified variables in your dataset, dropping the rest.
- `rename()`: Changes the names of variables without removing others.
- `starts_with()`, `ends_with()`, `num_range()`, `matches()`: Helper functions within `select()` for dynamic variable selection by patterns, ranges, or regular expressions.
- `filter()`: Finds rows or cases based on specified conditions.
- `mutate()`: Adds new variables while keeping existing ones intact. Ensure new variable names are unique.
- `transmute()`: Similar to `mutate()`, but only keeps newly created variables, dropping the rest.
- **%>%** (piping): Chain multiple functions together for cleaner, more readable code.

Let's Practice!

Before starting the exercises, remember that in R there are often multiple ways to accomplish the same task. This means there's no single correct method, and the code you've written may work just as well as other

approaches.

Exercise 2: Select Key Variables

Select only the variables related to reserved seats and reserved government positions for minority groups, as well as those related to subnational policy authorities. These include:

- Reserved seats (Mandated)
- Reserved Executive Positions (Mandated)
- Reserved Executive Positions (Implemented)
- Subnational Education Authority
- Subnational Tax Authority
- Subnational Police Authority

Also, retain the three variables that include country names, country codes, and the years.

Use this and upcoming exercises as practice to go back to the codebook and see how the variables are coded and the names they are identified by so you can use them to run your code.

Helpful Hint: Try using the `contains()`, `matches()` or `starts_with()` functions to select variables for reserved positions and subnational policy authorities efficiently. You can also use the concatenate function `c()` to create a vector of the variables you want to select, and then use the `all_of()` helper function to grab all the items in the vector.

```
# Select specific variables
## Option A: create a vector with all variables needed and use 'select(all_of())'
vars_1 <- c("gwno", "country", "year", "resseats_IDC", "resman_IDC",
            "resimp_IDC", "subed_IDC", "subtax_IDC", "subpolice_IDC")

dt_revised <- dt %>%
  select(all_of(vars_1))

## Option B: use 'select(matches())' to get 'sub' and 'res' variables
dt_vars <- dt %>%
  select("gwno", "country", "year", matches('sub|res'))%>%
  select(!resseatsimp_IDC)

## Option C: use 'select(starts_with))' to get 'sub' and 'res' variables
dt_vars <- dt %>%
  select("gwno", "country", "year", starts_with(c("res", "sub"))) %>%
  select(!resseatsimp_IDC)

## Option D: use 'select(contains))' to get 'sub' and 'res' variables
dt_vars <- dt %>%
  select("gwno", "country", "year", contains(c("res", "sub"))) %>%
  select(!resseatsimp_IDC)

## Option E: use 'select()' without helper functions
dt_revised2 <- dt %>%
  select(gwno, country, year, resseats_IDC, resman_IDC, resimp_IDC,
         subed_IDC, subtax_IDC, subpolice_IDC)
```

Exercise 3: Filter by Subnational Authority

Create a subset of the data where countries have subnational authority over either education or taxation.

Helpful Hint: `subtax` and `subed` are binary variables: they take a value of 1 if countries have subnational authority over either education or taxation, and 0 otherwise.

```
# Filter by subnational authority
df_sub <- dt %>%
    filter(subtax_IDC == 1 | subed_IDC == 1)
    ## Filter the dataset to have subtax_IDC == 1 and subed_IDC == 1
```

Exercise 4: Filter by Reserved Seats

Identify a country-year where more than 20% of legislative seats are reserved for minority groups.

```
# Filter for countries with more than 20% reserved seats
dt_resseats <- dt %>%
  filter(resseats_IDC > 0.2)
  ## Filter the dataset to have resseats_IDC > 0.2

## In the year 2012, the country Barbados saw more than 20% of the legislative seats
## reserved for members of minority groups.
```

Exercise 5: Create a Binary Variable

Create a binary variable named `resseats_10` capturing whether at least 10% of legislative seats are reserved for minority groups.

```
# Create binary variable 'resseats_10'
dt_revised <- dt_revised %>%
  mutate(resseats_10 = resseats_IDC >= 0.1)
  ## Create binary variable through mutate
```

Exercise 6: Summarize Over Time

Now, let's say that we want to know whether imposed reserved executive positions that have been getting more common over time. Calculate the global proportion of countries that have this type of protection (i.e. the global average of imposed reserved executive positions) by year.

Helpful Hint: You may have to remove missing values coded as `NA` or `-44`.

```
# Claculate the global average of resimp by year
dt_revised <- dt_revised %>%
  group_by(year)%>%
  ## Group by year
  filter(resimp_IDC != -44 & resimp_IDC != "NA")%>%
  ## Take out -44 and NA values
  summarise(mean_resimp = mean(resimp_IDC, na.rm=TRUE))
  ## Find global average of resimp
```

Exercise 7: Combine with Piping

Use piping to combine steps from exercises 2 and 5 into one streamlined command. Also, make sure to remove any missing values.

```
# Put it all together with %>%
final <- dt %>%
```

```r
  select("resseats_IDC", "resman_IDC", "resimp_IDC",
         "subed_IDC", "subtax_IDC", "subpolice_IDC", "gwno",
         "country", "year") %>%
  ## Select key variables
  mutate(resseats_10 = resseats_IDC >= 0.1) %>%
  ## Create 'resseats_10' binary variable
  filter(resimp_IDC != -44 & resimp_IDC != "NA")
  ## Take out -44 and NA values
```

Conclusion

This groupwork assignment is designed to give you hands-on practice with data management techniques in R. By working with the `IDC_training_2021.rds` dataset, you'll gain valuable experience in handling real-world data, applying functions, and thinking critically about data management. Keep in mind that when learning to code, being proactive in seeking out resources and regularly practicing the commands you've learned is key to improving your coding skills.