SPEC REU R Resources: Reshaping Data To Make it Tidy – Group Work

Alix Ziff, Jasmine Chu, Claudia Salas Gimenez and Ben Graham

January 2025

In previous assignments, we worked with data already organized in a country-year format. Now, this groupwork assignment will invite you to think about tidying and merging data—key steps in data management for research—using concepts covered in the walkthrough documents. By the end of this exercise, you'll gain hands-on experience using functions like `pivot_longer()`, `pivot_wider()`, or `full_join()` to tidy and merge data.

To complete this assignment, you'll need the following files: `LATVIA_NJ_fortraining.csv` and `FINLAND_JC_fortraining.csv`

Initial Setup

Start by setting up your R script with a clear header and setting your working directory to the folder where your data files are located. Clear any previous objects from your environment, load the necessary R packages (like `tidyverse`), and load the datasets.

Helpful Hint: Using `read.csv()` to load data will add "X"s in front of column names, while `read_csv()` will leave column names as numbers, representing years. Refer to DM II: Walkthrough 1 if you need a reminder on differences between these functions.

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

# Load libraries
library(tidyverse)
library(readr)

# Load the datasets using read_csv()
latvia <- read_csv("LATVIA_NJ_fortraining.csv")
finland <- read_csv("FINLAND_JC_fortraining.csv")
```

Key Function Recap

The `tidyr` package provides functions crucial for reshaping data:

- `pivot_longer()` changes data to a long format, turning multiple columns into rows. This increases the number of rows while consolidating observations, allowing you to analyze each year separately.
- `pivot_wider()` converts data from long to wide format by spreading values across columns, which is useful for having one row per observation.

Let's Practice!

Exercise 1: Tidy the Data

Transform each dataset to a tidy format, where variables are represented in columns and each row represents a year. Include a country identifier for each dataset to distinguish Finland's and Latvia's data when merged.

```
# Clean and reshape the Finland data
finland <- finland %>%
  mutate_at(2:11, as.numeric) %>%
  # Convert columns 2 through 11 to numeric
  filter(COUNTRY != "") %>%
  # Filter out any empty values in the column 'COUNTRY'
  pivot_longer(cols = `2010`:`2019`,
               names_to = "year",
               values_to = "value") %>%
  # Spread rows into columns based on country names
  pivot_wider(names_from = COUNTRY,
              values_from = value,
              id_cols = year) %>%
  # Spread rows into columns based on country names
  mutate(year = as.numeric(year)) %>%
  # Convert year to numeric
  mutate(country = "Finland")
  # Add a new column 'country' with all values set to 'Finland'
```

```
# Clean and reshape the Latvia data
latvia <- latvia %>%
  mutate_at(2:11, as.numeric) %>%
  # Convert columns 2 through 11 to numeric
  pivot_longer(cols = `2010`:`2019`,
               names_to = "year",
               values_to = "value") %>%
  # Spread rows into columns based on country names
  pivot_wider(names_from = COUNTRY,
              values_from = value,
              id_cols = year) %>%
  # Spread rows into columns based on country names
  mutate(year = as.numeric(year)) %>%
  # Convert year to numeric
  mutate(country = "Latvia")
  # Add a new column 'country' with all values set to 'Latvia'
```

Exercise 2: Conversion to Numeric

Ensure all variables are numeric.

Helpful Hint: Use the `str()` function to confirm the data structure.

```
# Convert year columns to numeric for both datasets
finland<- finland %>%
  mutate_at(vars(year:jrevman), as.numeric)

latvia<- latvia %>%
  mutate_at(vars(year:jrevman), as.numeric)
```

```r
# Check the structure of the data to verify numeric conversion
str(finland)
str(latvia)
```

Exercise 3: Merge the Datasets

Combine the two datasets into a single dataframe.

```r
# Merge the datasets into a single dataframe
full_data <- full_join(latvia, finland)
```

Exercise 4: Variable Renaming

Rename a subset of six variables to simple, descriptive names. Keep all variables—avoid subsetting.

```r
# Rename selected variables
full_data <- full_data %>%
  # Use the piping command to rename multiple variables
  rename(c("police" = "subpolice")) %>%
  rename(c("rel_est" = "relestablish")) %>%
  rename(c("seats" = "resseats")) %>%
  rename(c("martiallaw" = "martiallaw_binary")) %>%
  rename(c("religion" = "relrestrict")) %>%
  rename(c("party" = "partynoethnic"))
```

Exercise 5: Save Your Work

Save the final dataset as an `.rds` file in your homework submission folder and keep an annotated copy of your R script for reference.

```r
# Save the final merged dataset as an .rds file
saveRDS(full_data, file = "M3_2_GroupWork_DM2_012025.rds")
```

Bonus Question: Exploring Different Ways to Merge Data

Try merging the Latvia and Finland datasets using different merging approaches, and see how each one shapes the final dataset. Notice any differences? Look at the number of rows and columns, and think about what's included or left out in each merged version. For additional guidance on what formula to use, refer back to DM II: Walkthrough 2 if needed.