Data Management for Visualization: Walkthrough

Alix Ziff, Miriam Barnum, Ben Graham, Abigail Longstreth, Yuchen Gong, Annie Street

9/12/2023

Data Management for Visualization

In this module, we will continue working with our IDC powersharing data to hone our data management skillset. We will use the package `dplyr` to work with country-year data. While group_by() and summarise() are introduced in Data Management II, this training goes more in depth.

```
library(tidyverse)
```

```
idc_controls <- readRDS(file.path("./IDC_training_2022.rds"))
```

Collapsing Country-Year Data

univariate

First, we want to collapse the country-year data down to annual global averages. In other words, we need to `group_by(year)`. We'll start by focusing on a single variable: resource rents.

Helpful Hint: the new data object has much less information than the data you started with, so give it a new name, rather than overwriting the data you started with

`na.rm = T` tells R to run a command ignoring the missing values. This allows R to still calculate things like averages using just the non-missing values. NOTE: This means that the average for a year may be affected by what countries have missing values for that year. For example, if a bunch of poor countries with higher infant mortality rates have missing data, dropping those observations when calculating the global average will make the global average look artificially high.

```
rents_global <- idc_controls %>%
  group_by(year) %>%
  summarise(natresource_rents_mean = mean(natresource_rents_WDI, na.rm = T))
View(rents_global)
```

multivariate

Let's get global averages for multiple variables at once. We may want to compare resource rents with trade. To take a look, we'll do the same thing as above, but will use the `summarise_at()` function to bring in our second variable. We'll call this new object trade_rents_global.

```
trade_rents_global <- idc_controls %>%
  group_by(year) %>%
  summarise_at(vars(natresource_rents_WDI, trade_WDI), mean, na.rm = T)
View(trade_rents_global)
```

As social scientists, we know that we can expect lots of regional variation. This is especially likely when looking at things like resources and trade. In order to look at our regional averages, we need to `mutate()` our data and a region variable and then pipe together our code.

```
rents_regional <- idc_controls %>%
  group_by(region, year) %>%
  summarise(natresource_rents_mean = mean(natresource_rents_WDI, na.rm = T))

View(rents_regional)
```

Summarizing & Simplifying

multivariate

Let's try to simplify our data a bit so we can identify broad patterns. We'll start by collapsing our country-year data on resource rents down to country-decade data. Then, we'll take a look at our averages, minimum, maximum, and median.

Helpful Hint we'll add a decade variable by mutating our dataset and subtracting ten from our start date. `Year %% 10` gives us the last digit in the year, then we subtract from the year to get the decade.

```
rents_decade <- idc_controls %>%
  mutate(decade = year - year %% 10) %>%
  group_by(gwno, decade) %>%
  summarise(rents_mean = mean(natresource_rents_WDI, na.rm = T),
            rents_median = median(as.numeric(natresource_rents_WDI), na.rm = T),
                   rents_max = max(natresource_rents_WDI, na.rm = T),
                   rents_min = min(natresource_rents_WDI, na.rm = T),)
```

an in-depth look at the modulo operator (%%)

The modulo operator, %%, performs division between two values and returns the remainder. For example:
1977 %% 10 = 7
{first: 1977 / 10 = 197.7
then: 0.7 x the divisor = remainder
0.7 x 10 = 7}

the modulo operator performs both of these steps in one operation.
So, when we assign (1977 - 1977 %% 10) to our decade object, we're essentially calculating 1977 - 7, which gives us 1970.

univariate

We'll do the same thing with our global_rents data on trade and resource rents. We use the `lst()` function from the tibble package as a helper function here. `tibble::` allows us to call a function from the tibble package without first loading that package into memory via `library()`.

```
trade_rents_decade <- idc_controls %>%
  mutate(decade = year - year %% 10) %>%
  group_by(gwno, decade) %>%
  summarise_at(vars(natresource_rents_WDI, trade_WDI), tibble::lst(mean, median, max, min),
               na.rm = T)
View(trade_rents_decade)
```

a note on the lst() function: `lst()` can be paired with `tibble()` to apply functions in a list to objects in a data frame. The `lst()` function used here allows us to create a column in our data frame for each summary statistic (for each object). Because we have included it within the `summarise_at()` command, R will automatically apply the functions in our `lst()` to each object in `summarise_at()`. Then, `lst()` also applies new column names for us.

before we can make pretty pictures. . . we prep!

making maps

In order to make map figures, we need a single value per country. To do this, we'll pull out a single cross-sectional year either using base R or dplyr.

```
#base R
idc_2006 <- idc_controls[idc_controls$year == 2006,]
#dplyr
idc_2006 <- idc_controls %>%
  filter(year == 2006)
```

line plots summarizing change over time for countries, regions, and the world

We may also want a dataset that includes the annual values for some individual countries, one region, and the global average. We already have our global averages, we just need to add a region name to it.

```
trade_rents_global$region <- "Global"
```

Say we want to compare countries in North America. We'll first need to get regional averages by continent and then filter so we just keep the Americas.

```
trade_rents_americas <- idc_controls %>%
  group_by(region, year) %>%
  filter(region == "Americas") %>%
  summarise_at(vars(natresource_rents_WDI, trade_WDI), mean, na.rm = T)
```

Next, we want to pull out annual data for the US, Canada, and Mexico.

```
idc_sub <- idc_controls %>%
  filter(country %in% c("United States of America", "Canada", "Mexico")) %>%
  select(region = country, year, natresource_rents_WDI, trade_WDI)
View(idc_sub)
```

Let's finish by binding it all into one dataframe. Since variables should have the same names across dataframes we'll just use `bind_rows`, which is the tidyverse version of `rbind`.

Helpful Hint: It is also possible to assign gwno numbers to the regions and the global average. For example, we could assign North America a gwno of 1000 and the world a gwno of 10000. If we do this, we could use a full_join() function to merge these dataframes together, just as if we were merging together a couple of different country-year datasets. This is what we do in the lecture video you watched.

```
df <- idc_sub %>%
  bind_rows(trade_rents_americas) %>%
  bind_rows(trade_rents_global)
View(df)
```

And now we have a dataframe prepared for visualization. In this case, we could make a nice line plot with year on the X axis and either natural resource rents or trade volume on the Y axis. We could plot,for example, how trade volume has evolved over time in the these three countries, the americas combined, and the world overall.