Data Management II – Walkthough 1: Reshaping Data To Make it Tidy

Alix Ziff, Gaea Morales, Zachary Johnson, Natalie Song, Marie Zaragoza, and Ben Graham.

January 2023

This walkthrough provides an introduction to reshaping datasets to make them "tidy", using the dplyr and tidyr packages, both part of the tidyverse. We will work with data drawn from a source often used in international relations research, the Correlates of War. The two files you need are cow_milex_untidy_1.rds and cow_milex_tidy_2.rds.

Before you get started, write a detailed header in your R script and save your R script. Then set your working directory to the Training Data folder.

First, load in your packages and read the data.

*Note: If we use read.csv here, instead of read_csv, it will add an X to the start of all of our column names, which will actually make our lives a little easier when we go to reshape these data.

```
library(tidyverse)
#library(dplyr)

cow1 <- read.csv("cow_milex_untidy_1.csv")
cow2 <- read.csv("cow_milex_untidy_2.csv")
```

First, we'll look at cow1.

```
View(cow1)
```

In this dataset, each row is a country (indicated by ccode), and each column is a year containing information about the country's military expenditures in the given year.

Our goal is to reshape our data to make it "tidy".

Recall that in a "tidy" dataframe:
1.) Each row is a unique observation (e.g., a country-year like the U.S. in 1985 or Zimbabwe in 2019)
2.) Each column is a variable with information about the observations (e.g., information about the country-years)
3.) There is only one type of observation in each dataframe (i.e. we don't have information about country-years and country-days in the same dataframe)

To better understand what a tidy dataframe looks like, we can take a look at cow_example.

```
cow_example <- read_csv("cow_example.csv")
head(cow_example)
```

```
## # A tibble: 6 x 5
##    ...1 ccode  year    milex milper
##   <dbl> <dbl> <dbl>    <dbl>  <dbl>
## 1     1     2  1967 75448000   3380
## 2     2     2  1968 80732000   3550
## 3     3     2  1969 81446000   3460
```

```
## 4     4     2  1970 77827008   3070
## 5     5     2  1971 74862000   2720
## 6     6     2  1972 77639000   2323
```

In cow_example, each row is a unique observation (in this case, country in a particular year), and each variable holds information about that observation (military expenditures or military personnel). This dataframe has only one type of observation (country-year data only).

Let's look at another example. This is from a YouGov public poll in October 2020. Participants were asked if they support the creation of the LEWIS registry, a national registry of police officers fired for misconduct.

```
lewis_example <- read_csv("lewis_example.csv")
head(lewis_example)
```

```
## # A tibble: 6 x 5
##    ...1 birthyr gender race  pid3
##   <dbl>   <dbl> <chr>  <chr> <chr>
## 1     1    1985 Male   White Democrat
## 2     2    2002 Female Black Democrat
## 3     3    1994 Male   Asian Democrat
## 4     4    1960 Male   White Independent
## 5     5    1960 Female White Democrat
## 6     6    1943 Female White Republican
```

In lewis_example, we see that each row is a unique observation (in this case, the poll participant), and each variable holds information about the observation (birth year, gender, race, political affiliation). The dataframe only has one type of observation (poll participant). Thus, this data is tidy!

We can make dataframes tidy by reshaping data. We can do this by using the pivot_longer() or pivot_wider() functions.

pivot_longer() makes datasets longer by increasing the number of rows and decreasing the number of columns. We use pivot_longer() in situations where there are multiple observations in each row of data. We only want information about one observation in each row.

```
longData <- pivot_longer(cow1, # this is the dataframe we are going to reshape
  names_to = "year", # naming the new column, which has the old column names as values
  values_to = "milex", # naming the new column that will have the values in it
  c(starts_with("X"))) %>% # the columns we want to pull down as rows.
  # We need to grab these columns as a single object. Concatenate is one way to do this.
  select(ccode, year, milex) # keep just the variables we want

names(cow1) <- sub('^X', '', names(cow1)) # takes out X

head(longData)
```

```
## # A tibble: 6 x 3
##   ccode year     milex
##   <int> <chr>    <int>
## 1     2 X            1
## 2     2 X1967 75448000
## 3     2 X1968 80732000
## 4     2 X1969 81446000
## 5     2 X1970 77827008
## 6     2 X1971 74862000
```

So now the data_origin column tells us where the information comes from and milex gives us the values for military expenditures, which come from the milex_wdi and milex_cow variables. There is only one piece of

information per row for each observation.

If we want to do the reverse, we use pivot_wider(). pivot_wider() makes datasets wider by decreasing the number of rows and increasing the number of columns. pivot_wider() is not used as often as pivot_longer(), but it can be used to tidy dataframes that have information from multiple variables for each observation. We only want one row of information for each observation.

```
head(cow2)
```

```
##   X ccode year variable    value
## 1 1     2 1967    milex 75448000
## 2 2     2 1967   milper     3380
## 3 3     2 1968    milex 80732000
## 4 4     2 1968   milper     3550
## 5 5     2 1969    milex 81446000
## 6 6     2 1969   milper     3460
```

Take a look at cow2. It is already tidy, but we want to expand it so that there are nformation from multiple varibles for each observation.

```
wideData <- pivot_wider(cow2,
  names_from = variable,
  values_from = value)

head(wideData)
```

```
## # A tibble: 6 x 5
##       X ccode  year    milex milper
##   <int> <int> <int>    <int>  <int>
## 1     1     2  1967 75448000     NA
## 2     2     2  1967       NA   3380
## 3     3     2  1968 80732000     NA
## 4     4     2  1968       NA   3550
## 5     5     2  1969 81446000     NA
## 6     6     2  1969       NA   3460
```

We can use the pivot_longer() and pivot_wider() functions to reshape data frames. Most often, we will use pivot_longer() to make data "tidy", meaning each row is a unique observation and each column is a variable with information about the observations. We make data tidy because it is easier to compare data and create data visualizations in this format.