9 Reshaping Data

Repeated Measures Data: Wide & Long formats

In human research we often take measurements on subjects at multiple timepoints. Often the most convenient way to collect data is to add columns of observations for each additional timepoint and keep all information for any one subject in a single row. Variable names can be appended with some indicator of timepoint, for example weight_t0, weight_t1, weight_t2, etc. This organizational scheme is called **wide format** because the data frames grow wider with each additional timepoint.

Strictly speaking, the wide format does not conform to tidy data principals. A column such as weight_t0 contains information on two variables, the weight measurement and the time measurement. While the wide format occasionally has advantages in data analysis, the format also presents limitations because the timepoint variable does not exist in a single column.

Long format organizes all the timepoint information into a single column and each repeated measures variable into a single column. Now, every column contains information on a single variable. If the data involve 3 timepoints, the long formatted data will have three times as many rows as the wide format. Additional timepoints make the long format longer, but do not change the width.

While the long format conforms to tidy data principals and is more versatile for data analysis, the format is not ideal for data collection. Information on each subject is spread across multiple rows, and subject identifers (name, age, subject ID, etc.) and other variables that are observed only once must still be recorded at each timepoint.

A common strategy is to collect data in wide format and use a program like R to convert the data into long format for analysis. We call the process of converting between wide and long formats **reshaping** or **pivoting**. These conversions can be straightforward or difficult depending on the complexity of the data and the variable names used. While it may be tempting to simply retype wide data into a new long-formatted spreadsheet, even for small data sets this process is tedious and error prone and creates duplicate versions of the raw data. Furthermore, to make our work transparent and repeatable, we need a record of everything we have done to our data. This is accomplished by using code for data manipulations, and is jeopardized by using spreadsheet applications such as Excel that save the final product but not the steps taken to get there.

Reshaping with a single repeated measures variable

Let's begin with a small dataset from a study of cholesterol levels in humans consuming two different types of margarine. For this study, cholesterol levels were obtained at three timepoints and the data were collected in wide format.

We will use the pivot_longer command from the tidyverse collection (dplyr package) to reshape the data. Pivot_longer can take many different arguments, but for cases involving a single repeated measures variable, we often use the following three arguments:

1. cols = identify the repeated measures columns
2. names_to = identify the name of a new column that will contain the old column names
3. values_to = identify the name of a new column that will contain the observations (values)

Lastly, because R creates character variables by default whenever string entries (letters) exist, we pipe the output into mutate and change timepoint to a factor variable.

```r
# Clear environment
rm(list = ls())

# Read in data
df <- read_csv("data/margarine.csv")

# Reshape to wide format
df.l <- df %>%
    pivot_longer(cols = 2:4,
                 names_to = "timepoint",
                 values_to = "cholesterol") %>%
    mutate(timepoint = factor(timepoint))
```

Reshaping with multiple repeated measures variables

We begin by importing and re-coding the friends cholesterol data we have been using in earlier lessons. For simplicity, we will forego re-coding the data and focus only on reshaping.

```r
rm(list = ls())

df <- read_csv(file = "data/Friends_Cholesterol.csv",
            col_types = "cnfnfnnnnnn")
```

Because we now have more than one repeated measures variable, we will use a slightly different group of arguments with pivot_longer. The arguments will take advantage of the underscore "_" column separator to distinguish the two parts of each variable name and will create a new collection of column names from those parts.

1. cols = identify the repeated measures columns
2. names_sep = identify a character that separates the wide-formatted names into two parts
3. names_to = identify names of the new columns in long format and sort the data accordingly.

This time the names_to argument is used to direct what will happen to the parts of each variable name in wide format, the part that came before the underscore and the part that came after the underscore. Before the underscore, we have our repeated measured variables (weight, hdl, ldl, tc, and bmi). After the underscore we have our timepoints. We will use a special command, ".value", that tells R to create a new column of observations for every unique name that came before the underscore. We will tall R to collect the timepoint information into a new column that we cleverly choose to call "timepoint".

The above description is difficult to conceptualize without seeing the command in action. Run the code below, view the df_l dataframe and work through the logic of the code again.

```r
df_l <- df %>%
  pivot_longer(cols = 6:11,
               names_sep = "_",
               names_to = c(".value", "timepoint")) %>%

# Convert timepoint to a factor variable and relabel
    mutate(timepoint = factor(timepoint),
           timepoint = fct_recode(timepoint,
                                  "initial" = "i",
                                  "final" = "f"))
```

Advanced Reshaping: Tricky dataframes

Sometimes data are produced without the foresight of pivoting and other data wrangling challenges. Often the variable names don't lend themselves to easy reshaping, because they don't use "_" as a separator or they don't use any separator at all. There is always a solution though! Below are solutions to some of the reshaping challenges you may encounter, but many more (and far worse) scenarios exist. Reviewing the pivot_longer help page will often provide solutions to problematic dataframes.

Reshaping with problematic character separators(escaping special characters)

Below we import a different version of the friends cholesterol data that uses a "." in the variable names. This is a poor choice because periods can have other meanings in R and we must use an escape character, two backslashes, for R to interpret the period as text.

```r
# Clear environment
rm(list = ls())

# Read in data
df <- read_csv("data/Friends.Cholesterol.csv")

# Import data & Reshape
df.l <-
    df %>%
    pivot_longer(cols = weight.i:ldl.f,
                 names_sep = "\\.",
                 names_to = c(".value", "timepoint"))
```

Reshaping with no separator

Using character indexes In the next two examples, we use a version of the data that has no character separator. Variable names are simply appended with a letter, i or f, that codes for initial and final timepoints. Fortunately, the names_sep argument can also break apart variable names based on an index value. In this case, we can use -1 as the index to send the last character of each variable name to the timepoint variable.

```r
# Clear environment
rm(list = ls())

# Import
df <- read_csv("data/FriendsCholesterol.csv")

# Reshape
df.l <- df %>%
    pivot_longer(cols = weighti:ldlf, # variable names used instead of indexes
                 names_sep = -1,
                 names_to = c(".value", "timepoint"))
```

Renaming variables In this example, the variable names use no separator and instead of appending with single characters, the words initial and final are simply appended to the variable names. This was another poor choice because there is no easy way to separate the parts of these variable names. Perhaps the quickest solution is to simply create new names in R that use an underscore. We can then reshape as before using the names_sep = "_" argument.

```r
# Clear environment
rm(list = ls())

# Here I read the data into a "temp" dataframe so that students can view the
# data and understand the problem before continuing.
temp <- read_csv("data/FriendsCholesterol.rn.csv")

df <- temp %>%
  rename(weight_initial = weightinitial,
         hdl_initial = hdlinitial,
         ldl_initial = ldlinitial,
         weight_final = weightfinal,
         hdl_final = hdlfinal,
         ldl_final = ldlfinal)

# Reshape to long format
df.l <-
  df %>%
  pivot_longer(cols = weight_initial:ldl_final,
               names_sep = "_",
               names_to = c(".value", "timepoint"))
```