Learn to Pivot!

David John Baker

20/03/2020

- Title Slide

But before getting started, let's import what we need today.

```
# Import Needed Data and Libraries
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(knitr)
study_data <- read_csv("../data/study_data.csv")
```

```
## Parsed with column specification:
## cols(
##   participant_number = col_double(),
##   ldl_time_1 = col_double(),
##   ldl_time_2 = col_double(),
##   ldl_time_3 = col_double()
## )
```

```
study_data_long <- read_csv("../data/study_data_long.csv")
```

```
## Parsed with column specification:
## cols(
##   participant_number = col_double(),
##   timepoints = col_character(),
##   ldl = col_double()
## )
```

```r
student_data <- read_csv("../data/student_data.csv")
```

```
## Parsed with column specification:
## cols(
##   Student = col_character(),
##   Monday = col_double(),
##   Tuesday = col_double(),
##   Wednesday = col_double(),
##   Thursday = col_double(),
##   Friday = col_double()
## )
```

```r
store_data <- read_csv("../data/store_data.csv")
```

```
## Parsed with column specification:
## cols(
##   product = col_character(),
##   November = col_double(),
##   December = col_double(),
##   January = col_double(),
##   February = col_double(),
##   March = col_double(),
##   Department = col_character(),
##   Location = col_character()
## )
```

Today we continue learning about the idea of tidy data. We'll build on some of concepts from last time so you can start to manipulate some of your own data and change the way they are organised with only a few lines of code.

The Shape of Data

- Slide of Brain Pic as computer

How many people here have often heard that your brain is a computer?

This is partially true based on how you define computation and how far you are willing to stretch the metaphor, but even though people say your brain *computes* things, human brains do not work exactly like the computer that's sitting in front of you.

If your brain and computers worked exactly the same, you'd imagine that both you and your computer would always want things in the same format to make it easier for both parties involved. If you've worked with computers a lot before, you may know this isn't always the case.

In practice (and reality), humans and computers often need to be flexible in how they think about things. What works well for one situation for a brain or a computer might not work well in another.

Take a look at the following table of data.

- Table in Slides

```r
kable(study_data)
```

participant_number	ldl_time_1	ldl_time_2	ldl_time_3
1	99	104	95
2	176	180	190
3	120	124	122
4	188	210	201
5	132	144	135

A scientist is running a study where they need to collect data at multiple time points on a study tracking someone's LDL (bad cholesterol) and need to enter those values in by hand or maybe a teacher needs to record daily grades, this is a way that is familiar to many people who work with spreadsheets.

The problem is that when the data is structured like this, in order to do some sort of calculation like count the total number of cigarettes smoked or the average grade per student, most of the software we have seen thus far needs the data to operate over columns as opposed to rows.

- Table with rows highlighted

For example, look at this new data frame. It contains the exact same data as the table before, but has a different shape to it.

```r
kable(study_data_long)
```

participant_number	timepoints	ldl
1	ldl_time_1	99
1	ldl_time_2	104
1	ldl_time_3	95
2	ldl_time_1	176
2	ldl_time_2	180
2	ldl_time_3	190
3	ldl_time_1	120
3	ldl_time_2	124
3	ldl_time_3	122
4	ldl_time_1	188
4	ldl_time_2	210
4	ldl_time_3	201
5	ldl_time_1	132
5	ldl_time_2	144
5	ldl_time_3	135

- Click to show columns highlighted

Both shapes of data are tidy but one is probably easier to enter data into and the other is in a format that works well with a lot of software that makes our jobs easier. Often we need to change our data from one of these shapes to the other. And importantly, each of these shapes of data has a specific name.

## New Vocabulary

When some of the columns of a data set have certain values, rather than the explicit name of a variable, the data is called *wide data*. When an observation is scattered across many rows, the data is called *long data*.

- Slide with half def
- Slide with defs and question

Take a second to talk with your partner and answer this question:

Given the data set shown here, which of the following choices best describes the current state of the data using terminology we just learned.

- A. The data set is wide and tidy
- B. The data set is long and tidy
- C. The data set is wide and not tidy
- D. The data set is long and not tidy.

## Pivoting

Now that we're on the same page, let's learn how to change between the two. When you want to change the shape of data in the way that we just talked about in the tidyverse, you use the `pivot` functions. If you have wide data and want to make it long, you use `pivot_longer()`; if you have long data and want to make it wide, you use `pivot_wider()`. We're going to first look at `pivot_longer()` then in another lesson we can explore `pivot_wider()`.

If you are going to use `pivot_longer()`, we can assume that your data is in a wide format. Since your data is in a wide format we know that some of the column names are actually *values* that look like variables. In order to get these as rows, we need to give our function four different arguments.

Four Arguments

There can be a lot of terms to keep track of here so let's start one by one.

The first argument for any `tidyverse` function is going to be the data we want to manipulate. We should know what the structure of our data is before trying to change its shape! Let's start using our example from before.

```
study_data
```

```
## # A tibble: 5 x 4
##   participant_number ldl_time_1 ldl_time_2 ldl_time_3
##                <dbl>      <dbl>      <dbl>      <dbl>
## 1                  1         99        104         95
## 2                  2        176        180        190
## 3                  3        120        124        122
## 4                  4        188        210        201
## 5                  5        132        144        135
```

- In the example below from before, which of our columns here look like variables, but are really values?

  In this case, the it would be the columns ldl_time_1, ldl_time_2, and ldl_time_3

We have now identified our columns that are hiding as variables but are *really* values that we might want to use in our analysis. Identifying columns that look like variables, but are really values is the first step in moving data from wide to format.

If you remember this, you will remember that this is the second argument that the `pivot_longer()` function takes `id_cols =`. The function `pivot_longer()` makes it easy for us and allows for us to use `select()` like notation to select out what columns we have identified.

If you need a refresher on all the things `select()` can do, check that out here.

Now that we have our data and identified what columns are values hiding as variables, we can now tell `pivot_longer()` what we have found. We now need to tell `pivot_longer()` both what kind of data was in our column names as well as what kind of data is in the column with the value as a header.

Let's fill that in with our example.

- Note here that our `cols` argument works like select, so we can use the `:` operator to select consecutive columns.
- Also be aware that the `names_to` and `values_to` arguments need to be `"charater"` arguments since you are creating the *names* of your new columns.

```
pivot_longer(data = study_data,
             cols = ldl_time_1:ldl_time_3,
             names_to = "timepoints",
             values_to = "ldl")
```

Take notice of the fact that we have given the `names_to` argument what kind of data was in our columns. Take notice of the fact that we have given the `values_to` argument what kind of data lived was in the columns with values as headers.

We now have everything we need to move from wide data to long. Let's run this to see if it worked.

```
pivot_longer(data = study_data,
             cols = ldl_time_1:ldl_time_3,
             names_to = "timepoints",
             values_to = "ldl")
```

```
## # A tibble: 15 x 3
##    participant_number timepoints   ldl
##                 <dbl> <chr>      <dbl>
##  1                  1 ldl_time_1    99
##  2                  1 ldl_time_2   104
##  3                  1 ldl_time_3    95
##  4                  2 ldl_time_1   176
##  5                  2 ldl_time_2   180
##  6                  2 ldl_time_3   190
##  7                  3 ldl_time_1   120
##  8                  3 ldl_time_2   124
##  9                  3 ldl_time_3   122
## 10                  4 ldl_time_1   188
## 11                  4 ldl_time_2   210
## 12                  4 ldl_time_3   201
## 13                  5 ldl_time_1   132
## 14                  5 ldl_time_2   144
## 15                  5 ldl_time_3   135
```

Great! We've done it.

Let's try another one as a group. Take a look at this new data and see if you can fill in the blanks in order to get this new data from long format to wide. In this data set we have data from a few different students and their daily grades from one week of the year. We want every row to contain the name of the student, the day of the week, and their grade.

Remember to first uncomment the line so you can run it!

```
student_data
```

```
## # A tibble: 5 x 6
##   Student Monday Tuesday Wednesday Thursday Friday
##   <chr>    <dbl>   <dbl>     <dbl>    <dbl>  <dbl>
## 1 Ioana       78      65        66       69     98
## 2 Banji       89      98        88       77     78
## 3 Zara        88      78        90       89     79
## 4 Tina        90      79        90       79     99
## 5 Chris       69      89        70       82     91
```

```
# pivot_longer(data = ______ , cols = _______, names_to = ______, values_to = ______)
```

To end, let's try one more, but this time I am only going to give you the data and tell you that I want you to take this data and make it longer! The data here contains sales of products from a family owned store over the past few months. Here we want to take the sales from the different months and make it so that we have a variable called month and a variable called number of sales. There are a few other variables in there too, but just focus on the ones you want to pivot and `pivot_longer()` will take care of the rest.

```
store_data
```

```
## # A tibble: 8 x 8
##   product     November December January February March Department Location
##   <chr>          <dbl>    <dbl>   <dbl>    <dbl> <dbl> <chr>      <chr>
## 1 hammer            25       23      56        7    46 Home       Milwaukee
## 2 nail             105      452     454      563   222 Home       Milwaukee
## 3 screwdriver       34       67      43       32    52 Home       Chicago
## 4 pliars            24       23      23       24    39 Home       Milwaukee
## 5 drywall           58       78      56       34    64 Home       Chicago
## 6 stapler           21       11      18       31    33 Office     Chicago
## 7 printer ink       42       43      64       53    55 Office     Chicago
## 8 posit             97       75      37       97    64 Office     Milwaukee
```

```
# Your Code Here!
```

We covered a lot of ground in the last 15 minutes. We reviewed the idea of tidy data and learned two new terms to describe the shape of data. What are they?

> Wide and Long

We also looked at the first of two `pivot` functions, `pivot_longer()` which takes wide data and makes it long. The function `pivot_longer()` needs four arguments at minimum. What are they?

> Data, cols, names_to, values_to

Great, let's take a quick break, then do `pivot_wider()`...