Since version 1.0.0, released in September, the *tidyr* package has a new replacement for the *gather*/*spread* pair of functions, called *pivot\_longer*/*pivot\_wider*. (See the [blog post](https://www.tidyverse.org/blog/2019/09/tidyr-1-0-0/) about the release. It can do a lot of cool things.) Just what we needed, another pair of names for *melt*/*cast*, right?

Yes, I feel like this might just be what we need!

My journey started with *reshape2*, and after a bit of confusion, I internalised the logic of *melt*/*cast*. Look at this beauty:

library(reshape2)

fake\_data <- data.frame(id = 1:20,

variable1 = runif(20, 0, 1),

variable2 = rnorm(20))

melted <- melt(fake\_data, id.vars = "id")

This turns a data frame that looks like this …

id variable1 variable2

1 1 0.10287737 -0.21740708

2 2 0.04219212 1.36050438

3 3 0.78119150 0.09808656

4 4 0.44304613 0.48306900

5 5 0.30720140 -0.45028374

6 6 0.42387957 1.16875579

… into a data frame that looks like this:

id variable value

1 1 variable1 0.10287737

2 2 variable1 0.04219212

3 3 variable1 0.78119150

4 4 variable1 0.44304613

5 5 variable1 0.30720140

6 6 variable1 0.42387957

This is extremely useful. Among other things it comes up all the time [when using *ggplot2*](https://onunicornsandgenes.blog/2014/02/20/books-and-lessons-about-ggplot2/).

To be fair, ”melt” and ”cast” felt equally arbitrary, but by that time I was used to them. Getting the logic of the arguments, the order, what needed quotation marks and not, some starting at examples and a fair bit of trial and error.

Here are some examples. If you’re not used to these functions, just skip ahead, because you will want to learn the pivot functions instead!

library(tidyr)

melted <- gather(fake\_data, variable, value, 2:3)

## Column names instead of indices

melted <- gather(fake\_data, variable, value, variable1, variable2)

## Excluding instead of including

melted <- gather(fake\_data, variable, value, -1)

## Excluding using column name

melted <- gather(fake\_data, variable, value, -id)

Enter the pivot functions. Now, I have never used pivot tables in any spreadsheet software, and in fact, the best way to explain them to me was to tell me that they were like *melt*/*cast* (and *summarise*) … But *pivot\_longer*/*pivot\_wider* are definitely friendlier on first use than *gather*/*spread*. The naming of both the functions themselves and their arguments feel like a definite improvement.

long <- pivot\_longer(fake\_data, 2:3,

names\_to = "variable",

values\_to = "value")

# A tibble: 40 x 3

id variable value

1 1 variable1 0.103

2 1 variable2 -0.217

3 2 variable1 0.0422

4 2 variable2 1.36

5 3 variable1 0.781

6 3 variable2 0.0981

7 4 variable1 0.443

8 4 variable2 0.483

9 5 variable1 0.307

10 5 variable2 -0.450

# … with 30 more rows

We tell it into what column we want the names to go, and into what column we want the values to go. The function is named after a verb that is associated with moving things about in tables [all the way to matrix algebra](https://en.wikipedia.org/wiki/Pivot_element), followed by an adjective (in my opinion the most descriptive, out of the alternatives) that describes the layout of the data that we want.

Or, to switch us back again:

wide <- pivot\_wider(long,

names\_from = "variable",

values\_from = "value")

# A tibble: 20 x 3

id variable1 variable2

1 1 0.103 -0.217

2 2 0.0422 1.36

3 3 0.781 0.0981

4 4 0.443 0.483

5 5 0.307 -0.450

6 6 0.424 1.17

Here, instead, we tell it where we want the new column names taken from and where we want the new values taken from. None of this is self-explanatory, by any means, but they are thoughtful choices that make a lot of sense.

We’ll see what I think after trying to explain them to beginners a few times, and after I’ve fought warning messages involving list columns for some time, but so far: well done, *tidyr* developers!