Reshaping data frames with dplyr

CRI R Workshop

Why, and what?

Sometimes you may have several different data frames with data on the same set of samples, and you'd like to put them together into one data frame. If these separate data frames are set up differently from each other, you may need to restructure one or both to more easily align them. Or, your data may be in one data frame, but not in a form that works with the analysis functions that you want to use (e.g., for statistics).

In the next two lessons, we will introduce you to reshaping functions from the package `tidyr` and join functions from the package `plyr`, which you've already had some experience with. These functions will help you organize your data for analysis both in terms of making your data "tidy" as well as putting all observations or variables into the same data frame.

Let's start by installing the package `tidyr`.

```
install.packages("tidyr")   ## install the package
```

```
library("tidyr")   ## load the package
```

Data formats and tidy data

Data frames can be in either long format or wide format.

To demonstrate, let's create a data frame in long format. There are three different measurements for each sample: ammonium, nitrate, and nitrite. All of the values are in a single column, and the measurement is specified by a value in a second column. In `tidyr` and other similar packages, this variable (`Nutrient`) is called the *key*.

```
nutrients_long <- data.frame(SampleID = rep(c(1, 2, 3), 3)
                          ,Nutrient = c(rep("Ammonium",3), rep("Nitrate",3), rep("Nitrite",3))
                          ,Concentration = c(8.2, 6.9, 12.1, 1.7, 3.6, 2.8, 0.4, 1.5, 0.8)
                          )
nutrients_long
```

```
##   SampleID Nutrient Concentration
## 1        1 Ammonium           8.2
## 2        2 Ammonium           6.9
## 3        3 Ammonium          12.1
## 4        1  Nitrate           1.7
## 5        2  Nitrate           3.6
## 6        3  Nitrate           2.8
## 7        1  Nitrite           0.4
## 8        2  Nitrite           1.5
## 9        3  Nitrite           0.8
```

Here is the same data frame, using color to help you visualize the structure:

SampleID	Nutrient	Concentration
1	Ammonium	8.2
2	Ammonium	6.9
3	Ammonium	12.1
1	Nitrate	1.7
2	Nitrate	3.6
3	Nitrate	2.8
1	Nitrite	0.4
2	Nitrite	1.5
3	Nitrite	0.8

But what if we want all of the data from one sample to be in a single row, with each nutrient in a separate column? This would be called wide format, and it would look like this:

SampleID	Ammonium	Nitrate	Nitrite
1	8.2	1.7	0.4
2	6.9	3.6	1.5
3	12.1	2.8	0.8

This is an example of "tidy" data (Wickham 2014). It has the following characteristics:

1. Each variable forms a column.
2. Each observation forms a row.

Formatting your data in this way can streamline downstream analyses. A consistent data format makes it easier to pass data to functions and analyses.

Of course, there are other structures and formats that may be more convenient for other purposes, but this one tends to be the most effective for analysis in R.

The data you work with is likely more complex than in this example, and it can take some thought to decide what constitutes a variable.

If the structure of your experiment involves groups between which you want to compare aggregate data (mean, maximum, etc.), these groups should usually be specified by a variable (i.e., a column), with each sample in each group in a separate row. For example, say you have two treatments applied to each `SampleID`, and you want to compare nutrient concentrations between treatments. Ideally, you would include a `Treatment` variable in your data frame, and each `SampleID` would be associated with 2 rows, one for each `Treatment`. The structure would be the same if you had multiple sampling dates instead of multiple treatments. It would also be similar if you had two `Treatments` that each had 3 replicates, with the variable name `Replicate` replacing `SampleID`.

Treatment *or* SamplingDate	**SampleID *or* Replicate**	**Ammonium**	**Nitrate**	**Nitrite**
1	1	8.2	1.7	0.4
1	2	6.9	3.6	1.5
1	3	12.1	2.8	0.8
2	1	10.5	0.4	0.7
2	2	8.6	2.7	1.2
2	3	7.8	4.1	0.9

This is a tidy dataset!

In contrast, the following "messy" dataset has separate columns for different replicates of each treatment. This will make things much more difficult later and should be avoided. However, if your data comes this way, e.g. as output from an analytical instrument, it is straightforward to use reshaping functions to put it into a tidier form!

Treatment	Ammonium_1	Nitrate_1	Nitrite_1	Ammonium_2	Nitrate_2	Nitrite_2	Ammonium_3	Nitrate_3	Nitrite_3
1	8.2	1.7	0.4	6.9	3.6	1.5	12.1	2.8	0.8
2	10.5	0.4	0.7	8.6	2.7	1.2	7.8	4.1	0.9

For more on tidy data, here is Hadley Wickham's informal version of his full tidy data paper: https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html

Converting between long and wide formats

Depending on the initial format of the data that you are working with, it may be necessary to convert it between long and wide format (and possibly in both directions) to make it tidy.

A simple way to convert your data from long to wide format is to use the function `spread` in `tidyr`. Here, we'll start with the data frame `nutrients_long` and convert it to wide format. As a reminder, here is the `nutrients_long` data frame:

```
##   SampleID Nutrient Concentration
## 1        1 Ammonium           8.2
## 2        2 Ammonium           6.9
## 3        3 Ammonium          12.1
## 4        1  Nitrate           1.7
## 5        2  Nitrate           3.6
## 6        3  Nitrate           2.8
## 7        1  Nitrite           0.4
## 8        2  Nitrite           1.5
## 9        3  Nitrite           0.8
```

```
nutrients_wide <- nutrients_long %>%
  spread(key = Nutrient, value = Concentration)

nutrients_wide
```

```
##   SampleID Ammonium Nitrate Nitrite
## 1        1      8.2     1.7     0.4
## 2        2      6.9     3.6     1.5
## 3        3     12.1     2.8     0.8
```

In the function `spread`, the `key` argument is the name of the column that contains a list of the data descriptors (or measurement names) that we want to separate, and these descriptors will become column names in wide format. The `value` argument indicates the values that will populate the data frame.

Tip: When you're figuring out how to restructure your data for a downstream analysis, it can be helpful to write out a quick sketch of how you want the data to be organized, in terms of the names of the variables and the observations. This may help you figure out what input to give to each argument in the reshaping function.

We can convert wide format data back to long format using the function `gather`.

```
nutrients_wide %>%
  gather(key = "Nutrient", value = "Concentration", Ammonium, Nitrate, Nitrite)
```

```
##   SampleID Nutrient Concentration
## 1        1 Ammonium           8.2
## 2        2 Ammonium           6.9
## 3        3 Ammonium          12.1
## 4        1  Nitrate           1.7
## 5        2  Nitrate           3.6
## 6        3  Nitrate           2.8
## 7        1  Nitrite           0.4
## 8        2  Nitrite           1.5
## 9        3  Nitrite           0.8
```

In this case, the argument `key` gives the *name* of the column that will have the data descriptors in it (here, "Nutrient"), and the argument `value` has the *name* of the column with the values (here, "Concentration"). The remaining input tells the function which variables to gather. In this case, this is a list of columns. For

each column, the column name will be used to populate the named key column, and the values will be used to populate the named value column. SampleID is not listed and is maintained as an identifier for each observation - i.e., it is not gathered.

You can accomplish the same task by telling R that you want to gather all of the columns besides SampleID, as follows:

```
nutrients_wide %>%
  gather(key = "Nutrient", value = "Concentration", -SampleID)
```

You can also specify these columns using column numbers. Try these yourself to demonstrate that they give the same output.

```
nutrients_wide %>%
  gather(key = "Nutrient", value = "Concentration", 2:4)

nutrients_wide %>%
  gather(key = "Nutrient", value = "Concentration", 2, 3, 4)
```

What happens if you only select two of the three columns of data?

Challenge

- If a column of a dataset is a character vector that includes information on two different variables, you can separate the information using the `separate` function. Use `gather`, `spread`, and `separate` to tidy the following "messy" data set so that it looks like the tidy dataset pictured above, i.e., `nutrients_wide` with separate rows for each replicate. (Look at the help file for `separate` - paying attention to the arguments `col`, `into`, and `sep` - to learn how to use it.)

```
nutrients_messy <- read.csv(file="../Data/Experiment_nutrients_messy.csv")
nutrients_messy
```

```
##   Treatment Ammonium_1 Nitrate_1 Nitrite_1 Ammonium_2 Nitrate_2 Nitrite_2
## 1         1        8.2       1.7       0.4        6.9       3.6       1.5
## 2         2       10.5       0.4       0.7        8.6       2.7       1.2
##   Ammonium_3 Nitrate_3 Nitrite_3
## 1       12.1       2.8       0.8
## 2        7.8       4.1       0.9
```

Resources

- Hadley Wickham on tidy data