R: Cleaning and tidying data

Quite often when you work with folks who are R experts or aficionados, you hear the term "data wrangling". To wrangle your data involves 3 steps according to Wickham and Grolemund (2017), the authors of R for Data Science: Importing your data, Tidying your data, and Transforming your data. We have already discussed importing or bringing your data into R, and we have discussed some transformations of the data as well - creating new variables and recoding. However, we have not discussed tidying your data. To me, this is synonymous with cleaning and reshaping your dataset. Depending on your analysis, you may need your data in a long columnar format or you may need your data in a broad and short format.

This section of the workshop we will use data that accompanies the tidyverse package.

Please download the R script used in the cleaning and tidying data.

What in the world is a Tibble?

No, it is not an imaginary creature from Fraggle Rock created by the Jim Henson Company. That's the first thing that comes to my mind :) A tibble in R is a newer way/version of the classic dataframe in R. More efficient way of handling your data, especially when printing it to your screen. There are many more advantages as well, but for now, we just need to recognize that it is a new and more efficient way of working with dataframes in R.

Tidy Data

Tidy data is another term that you may hear with working with data in R. There are 3 "rules" to having tidy data and they are:

1. Each variable must have its own column
2. Each observation must have its one row
3. Each values must have its own cell

To me, this is a clean dataset and nothing too new. New terms for the concept of a clean dataset. But be aware of the term "tidy data" and the 3 caveats - in case someone asks you.

Reshaping our data - Gathering

For many of us that have used SAS for many years, may see this particular aspect of R as transposing our data - going from wide to long. We will be using the table4a available in the package.

Notice that this table lists a country followed by 2 columns one for 1999 and a second for 2000 data. Our goal with this exercise is to create a tidy dataset that contains the following 3 variables: County, Year, Cases.
newtable4a <- gather(table4a,"1999", "2000", key="year", value="cases")

newtable4a - the name of the new tidy dataset we are creating

gather() is the function we are using to gather the data in the original columns labelled 1999 and 2000 into new variables called year and cases

We first need to tell the gather() the name of the original table - table4a in our case

We then list the original column names that we wish to gather into one column - so 1999 and 2000

key="year" - this is the name of the new variable that will contain the values of 1999 and 2000

value="cases" - our new variable that will hold that current values under 1999 and 2000 will be called cases, and their values will be placed here by country and the year it was identified with in the original table.

Try it out and see what happens. As you work through this example, I would like you to think about your own research data. Can you see a situation where this would be helpful for you?

Try replicating our example with the table4b - this table contains the country population.

**Reshaping our data - Spreading**

If I do the comparison again to SAS - spreading is the same as transposing again - but now we are going from a columnar dataset to a wide dataset. Imagine that you have collected your data on some variable for a number of months. Now we want to see if there is a correlation between the measures at each month - in order to perform the correlation analysis, we need each row or observation to contain the measurements taken at each month - so a wide dataset.

In R, we refer to this action as spreading the data. We will again use the sample data that accompanied the tidyverse package. Let's start by looking at table2. This table contains the variables: country, year, type, count. We want to create a table that is a tidy dataset and contains the variables: country, year, cases, and population.

newtable2 <- spread(table2, key=type, value=count)

newtable2 - the name of the new tidy dataset we are creating

spread() is the function we are using to spread the data

Similar to above within our spread() function, we first tell R the name of the original table we want to work with - table2

We then identify the key variable - so the variable that we will be breaking out the data - you'll notice that values in the type variable in table2 are cases and population - and these will be the variable names in our newtable2.
The last piece of information we need to inform R is that the value of our new variables are the values contained in the variable called count in the original table.

Try it out and see what happens. As you work through this example, I would like you to think about your own research data. Can you see a situation where this would be helpful for you?

**Changing Data Types**

When we introduced R in the first section of the workshop, I defined the different types of data that R can use. Depending on how you create your data in R, whether it is by importing it from Excel or a CSV file, or creating it directly in R, there may be situations where you may need to change a variable from one data type to another.

Let's try a few conversion with a new data file that you will first import from Excel. The data file is called woodchips.xlsx and can be downloaded directly from this link. Please import it into your R workspace by a method you prefer. I used the following code:

```r
woodchips <- read_excel("woodchips.xlsx", sheet="Sheet1", col_names=TRUE)
```

To see what is inside the dataset, simply type the name that you assigned to the dataset - in my case that would be:

```r
woodchips
```

You should see 3 variables: sampleID, woodwt, and quality, along with a snippet of the data. You should also see that the sampleID is a <chr> or character data type and both woodwt and quality are <dbl> or double precision floating point number (think of this as your decimal data).

In RStudio, in your Environment tab, you should also see the name of your dataset - woodchips. If you select the arrow to open the contents, you will see the names of the variables along with their structures and a few observation values. We see that SampleID is character and that both woodwt and quality are numeric. Let's replicate this in our Console window by using the str() or Structure function found within the Base R.

```r
str(woodchips)
```

Notice that the results are the same as in the Environment window.

**Changing characters to factors**

Let's take a look at our SampleID variable - it is a character but we want it to be treated as a factor - a classification variable. R is very particular about what types of variables can be used in what analyses. In this case we know we want to use SampleID as a factor. To convert it we need to use the as.factor() function as follows:
as.factor(woodchips$sampleID)

What do we need to do if we want R to replace the current contents of our variable sampleID in the dataset woodchips with the new factor version??

woodchips$sampleID <- as.factor(woodchips$sampleID)

Can we take a shortcut and drop the dataset designation? Why or why not?

Changing numeric to factors

Same concept, except now we have a numeric piece of information we want to be treated as a factor. For illustration purposes, let's use the quality variable. Why would we do this with our research data? When you set out your trial you labelled your plots or your treatments using numbers - we are not going to do any calculations with these numbers, but they will be used as a classification variable. In R, we need to set these as factors in our dataset. Again, for illustration purposes let's use the quality variable and convert it to a factor. Follow the same process we did above.

First check the structure of your variable, then change it to factor using the as.factor() function and check to make sure it worked.

str(woodchips$quality)
woodchips$quality <- as.factor(woodchips$quality)
woodchips$quality
str(woodchips$quality)

Changing factors to numeric

The reverse may be happen as well. We may have a variable in our dataset that we need to go from factor back to numeric. Our little example is a perfect one. In this case you would use the as.numeric() function. Let's try it on our quality variable.

str(woodchips$quality)
woodchips$quality <- as.numeric(woodchips$quality)
woodchips$quality
str(woodchips$quality)

Michelle