Linguistics 337 : Methods of Design+Analysis

R workshop 2: Tidy data + data wrangling

K.Bott / K. Becker

15 October 2020

Warm up: review and exploring a new dataset

You have just been given a new dataset from a linguistics study focusing on conversations with speakers in the Pacific Northwest.

Note that in this week's folder there are *many* datafiles. As you work through these exercises, make sure you are using the correct datafiles for each step.

begin: dataset `pnw_extract.csv`

Load your libraries, load the data, and take a look at the data. Keep in mind that you will need to change the filepath for working on your computer.

```r
library(tidyverse)
pnw <- read_csv("/Users/bottk/Downloads/pnw_extract.csv")
View(pnw)
```

In this class we have highlighted some data wrangling verbs as tools that you may use in your work this term.

(At the bottom of this document, I have placed review exercises and some additional exploration for thie dataset. If you get done with lab early and are feeling adventurous, take a look.)

Today we will focus on a new area of data wrangling: data *tidying*.

Tidy data

"Happy families are all alike; every unhappy family is unhappy in its own way" - Leo Tolstoy "Tidy datasets are all alike, but every messy dataset is messy in its own way" - Hadley Wickham

Tidy requirements

1. Each observation has its own row
2. Each variable has its own a column
3. Each measurement has its own cell

R/RStudio and all `tidyverse` tools work most smoothly with "tidy data." Your data might come to you as not-tidy and you will need to tidy it. There also may be reasons why you might wish to "un-tidy" your data.

In this class, we will use the PNW data as an example for how to reshape data - to both "wider" and "longer" formats, and then do some other dataset-level wrangling.

First, look at the PNW data and the above rules. Is this data "tidy"?

```r
View(pnw)
```

reshape with `pivot_longer()`

Technically, the values that are listed under F1, F2, and F3 are *hertz* values for a particular *formant*. The resident Data Witch, MeanBott (she/her), has declared that you MUST have columns for both of these variables.

While MeanBott is stomping in the corner and mumbling *"Not tidy! Not tidy!"*, take a moment to skim the `pivot_longer()` function documentation.

```
?pivot_longer
```

. . . then, using `pivot_longer`, make two new variables, "Formant" and "Hz", to Properly Tidy this data and calm down MeanBott.

```
pnw_tidy <- pnw %>%
  pivot_longer(cols=starts_with("F"),
               names_to = "formant",
               values_to = "Hz")
```

Take a look at your data – what happened? Spend a minute with both `pnw` and `pnw_tidy` dataframes to make sure that you understand the data reformatting.

```
View(pnw)
View(pnw_tidy)
```

(Note: You may see an older relative of this, `spread()`; the more recent `pivot_longer` has friendlier syntax.)

With the *longer* data format (`pnw_tidy`), you could evaluate the Hz values across F1, F2, and F3 – or easily filter out by formant. This might be interesting to someone who focuses on Hz readings in a non-linguistic context (physicist?), but makes little linguistic sense.

reshape with `pivot_wider()`

For linguists, it is more correct to consider formants as variables within an observation (vowel). No matter. While MeanBott isn't looking, we can put the data back like it was before:

```
pnw_wider <- pnw_tidy  %>%
  pivot_wider(names_from = "formant",
              values_from = "Hz")
```

. . . and, compare your three datasets

```
View(pnw)
View(pnw_tidy)
View(pnw_wider)
```

(Note: Similar to what we saw with `pivot_longer()`, `pivot_wider()` is an updated relative of `gather()`.)

Expanding datasets with `bind_rows()` and `bind_cols()`

Sometimes you might have reason to expand your dataset – either adding additional observations or additional variables.

[*psst* :: If your data is *tidy* . . . additional *observations* mean more – rows or columns? . . . additional *variables* would be more . . . which?]

adding observations with `bind_rows()`

I *thought* I was done with data collection - but I received a one-week extension on my project, so I collected additional data.

Your turn

1. Load in the `pnw_extract_extra.csv` data and name it `pnw_extra`
2. Note how many observations you have in the "extra" data
3. Take a look at the data using `View()`

This "extra" data has *the exact same structure* as my main `pnw` data, so I can append it to my `pnw` dataset using the tidy tool `bind_rows()`. (Think about *vertically* taping together two spreadsheet printouts.)

```
pnw_all <- bind_rows(pnw, pnw_extra)
```

Take a look at your combined data. How many observations do you have in the final dataset? Does that match your expectation?

adding variables with `bind_cols()`

Similarly, you can add columns (variables) to your dataset. The tool `bind_cols()` matches observations *by position* . In this case, I have duration values for each of my vowel observations, and I would like to add these to my dataset.

Start by reading in `pnw_extract_dur.csv`

```
pnw_dur <- read_csv("/Users/bottk/Downloads/pnw_extract_dur.csv")
```

. . . then take a look at the data

```
View(pnw_dur)
```

. . . and use `bind_cols()` to combine the datasets

```
pnw_final <- bind_cols(pnw_all, pnw_dur)
```

. . . and take a look at the resulting data

```
View(pnw_final)
```

Other data wrangling tools . . .

There are SO MANY (!!!) tools in R/RStudio for working with data, and it is not possible for us to cover every tool. I think what is *most important* is that you know that for **whatever** task you might need to do to your project data, *R can help you get that done.*

. . . for splitting up data

For example, perhaps you have data where all of your responses are snuggled together in one cell. (This isn't your fault; blame Google Sheets or Qualtrics or whatever your collection instrument was.) There a *number* of different ways you can tell R "break that data into different cells" – depending on what the data look like, you may use `separate()` or some function from package `stringr`.

. . . for combining data

We went through `bind_rows()` and `bind_cols()`; you can also combine datasets from different spreadsheets using a number of different types of *joins* (learn more with `?join`). You can combine data *within* a spreadsheet a number of ways. You can use `mutate()`, you can use `unite()` . . .

. . . for working with particular *kinds* of output

Folks who set up surveys using Qualtrics, your data will look Pretty Messy when you download the *.csv . There are special packages for *just* working with Qualtrics, which might be the way to go depending on what you want to do.

. . . for *your data in particular*

Recommendation: when you get your data together, work with Kara (to make an overall goal/plan) and KBott (for specifics of how to get to that goal, *kbott@reed.edu*) on how to clean up *your particular dataset.* The best way to learn these skills is with a real-world example, and each of your projects will likely need slightly different approaches.

additional bonus section / review: tasks + challenges

Working with the `pnw_final` dataset we created above:

group_by() + summarize()

Overall F1 and F2 values aren't terribly useful; we want to work with the data broken into *groups* by vowel and then calculate *average* F1 and F2 values. Below, we *group by vowel* and then *calculate means* for both F1 and F2. (The `arrange()` command orders our output by `vowel`.)

```
vowel_summary <- pnw_final %>%
  group_by(vowel) %>%
  summarise(meanF1 = mean(F1), meanF2 = mean(F2)) %>%
  arrange(vowel)
```

your turn

1. calculate mean values of variable `duration` for different vowels in the pnw_final data
2. calculate mean values of `F1` and `F2` for different *words* in the pwn_final data

mutate()

Often, it is necessary to create additional variables from raw data. Example: F1 and F2 are raw data; it is more common to work with normalized formants.

your turn

3. Use verb `mutate()` to create a new variable, `F2norm`, which is equal to the difference between F2 and F1.

filter() and select()

Say it with me – "FilteR works on Rows, seleCt works on Columns…"

your turn

4. Create a new dataset only containing observations related to vowel "AA"
5. Create a new dataset containing onbsrevations of vowel "AA" *and*" vowel "AH" (hint: for #5, think about your "and" logic … and "or" logic. Most people set this up backwards the first time through; feel free to check-in with KBott)
6. Create a new dataset that does not contain F3.
7. Create a new dataset that contains everything *except* F3. (Or, "Accomplish #6 again, but using different code.")