Week 5 Walkthrough
Subsetting Variables

James Robertson

January 31, 2025

Subsetting Variables

Related to lazy coders are lazy (read: efficient) computers. You always want a computer to do the *least* it possibly can to accomplish a given task. When I ask my computer to open up my web browser, I want it to do so as quickly as possible. I *do not* want it to take detours and side paths or use too much of my RAM. Related to this idea of laziness (efficiency) is not storing or manipulating more data than you need to.

Continuing with the survey data from the RENAME and LABEL document, this dataset actually has *a lot* more variables in it. It has respondents' names, emails, IP addresses, and a whole lot more! But the thing is, *I don't really care about those.* More than that, if I were to share this data, I would get in *a lot of trouble* if I included personally identifying information like that. So to cover my `liability` and free up storage space I need to not have those columns.

Using DROP and KEEP

Whenever we want to remove columns we have two options: DROP and KEEP. You may be able to imagine the distinction, but let me make it explicit:

- KEEP is used to specify *desired* columns, i.e. columns we would choose to *keep*
- DROP is used to specify *unwanted* columns, i.e. columns we would choose to *drop*

That's easy enough to say, but how should we actually use these? And how do we choose *which* to use?
In terms of how, KEEP and DROP are written the same way, but do totally opposite things. Consider the example below:

```
DATA lib.survey_anonymized1;
    SET lib.survey2;
    DROP firstName lastName eMail IPAddress
         date streetAddress DLN shoeSize;
RUN;

DATA lib.survey_anonymized2;
    SET lib.survey2;
    KEEP duration finish
         q1_2 q1_3 q1_4 q1_5;
RUN;
```

These two DATA steps produce the exact same data[1]! So why one over the other? The answer is simple: *I am lazy.* I don't want to type more than I have to, so I choose whichever leads to less typing. In this case KEEP is *way* less typing. KEEP also does a better job of helping me cover my `liability` because I may forget `streetAddress` was in the data and not successfully exclude it if I used DROP. Which would get me *super* fired. Additionally, if I had to change the names with RENAME initially, it would be convenient if I only renamed the variables I *wanted* and then used KEEP on those.

Similar to RENAME and LABEL, you can also have these as **options** in your SET statement! The syntax is similar.

```
DATA lib.survey_anonymized1;
    SET lib.survey2 (DROP = (firstName lastName eMail IPAddress
                             date streetAddress DLN shoeSize));
RUN;
```

[1] Trust me when I say there's no columns which aren't listed in either the KEEP or DROP **statement**. If there were other columns, these would *not* be equivalent!

```
DATA lib.survey_anonymized2;
    SET lib.survey2 (KEEP = (duration finish
                             q1_2 q1_3 q1_4 q1_5));
RUN;
```

Doing these sorts of manipulations as **options** can meaningfully improve runtimes if the dataset is large! If you go on to take ST445 (required for stat majors) you'll hear more about efficiency there. Or we can talk about it in office hours!