Lab 1: Introduction to Stata

Data: reach.csv (see the reach.pdf file for data dictionary/useful information).

Practical objective: To gain familiarity with implementation in Stata. Today, we will focus on data pre-processing, manipulation, and exploration techniques that will be used during this course.

Scientific objective: To investigate the effect of the REACH/FAMS interventions on hemoglobin A1c (HbA1c) in adults with highly uncontrolled type 2 diabetes. Our outcome will be HbA1c at six months. Our predictor will be treatment (control, REACH, or REACH + FAMS).

Noteworthy commands: Below is a list of some Stata commands (and some shortcuts) that will be helpful for this lab.

- `pwd`
- `cd`
- `import delimited`
- `help`
- `tabulate/tab`
- `generate/gen`
- `replace`
- `rename`
- `label define`
- `label values`
- `drop`
- `keep`
- `misstable summarize`
- `tabstat`
- `ttest`
- `prtest`

.do files: The purpose of a .do file is to keep a record of all the commands you run; they should be commented for your own benefit (*). This is a step to create and save a .do file:

1. File → New → Do-file
2. Complete coding
3. File → Save As… → *your_file_name.do*

Log files: Log files are designed to store your results after running a series of commands. These are the steps to save a log of your results.

1. File → Log → Begin
2. Complete coding
3. File → Log → Close

Working directory: The `pwd` command will allow you to see the present working directory. To change it, use the command `cd`. For instance:

```
cd "/Users/andrewspieker/BIOST 6312 - Spring 2022/DataSets"
```

Reading in data from a .csv file: You can read in a .csv file using the user interface in the expected way, or you can read it in from the command line once you are in the right working directory:

```
import delimited "reach.csv", clear
```

Strictly speaking, the working directory does not need to be the location of the data set. However, your code needs to reflect the correct path:

```
import delimited "~/BIOST 6312 - Spring 2022/DataSets/reach.csv", clear
```

Help files: If you need assistance with a command, your first line of attack is to look at the help files.

```
help [command]
```

You can also search online for Stata's manuals, and feel free to reach out for assistance!

Examples of data processing: In this example, we are going to compare multiple treatment groups. Tabulate the two treatment variables, REACH and FAMS.

```
tabulate reach fams
```

Now is a good time to mention that many Stata commands have shortcuts. The shortcut command for the tabulate command works just as well:

```
tab reach fams
```

Exercise 1: What do you learn from this cross-tabulation?

Let's create a variable that encodes information on the three groups. The `generate` command (shortcut: `gen`) can be used:

```
gen txgrp = 0
replace txgrp = 1 if reach == 1 & fams == 0
replace txgrp = 2 if reach == 1 & fams == 1
```

You can tabulate the newly defined treatment groups and verify that they make sense. You should become comfortable with logical "if," "and," and "or" statements, as well as inequalities.

```
tab txgrp if reach == 1
tab txgrp if reach != 0
tab txgrp if reach == 0 | fams == 1
tab txgrp if reach == 0 & fams == 1
tab txgrp if txgrp >= 1
tab txgrp if txgrp <= 1
tab txgrp if txgrp < 1
```

Exercise 2: Check to make sure the results from the above commands make sense.

Suppose we are not entirely thrilled with a variable's name. That can be changed using the `rename` command, which takes the following format:

```
rename oldname newname
```

Exercise 3: Rename the treatment group variable name to "trtgrp" instead of "txgrp."

Labels can be particularly helpful, especially if the coding is non-intuitive or if there are many unordered categories. While it is not particularly difficult to manage three categories, it's easy to forget the variable coding. Creating labels in Stata is a two-stage procedure. First, you need to define what the labels are, and then you need to assign them to the variable.

```
label define txlbl 0 "Control" 1 "REACH" 2 "REACH+FAMS"
label values trtgrp txlbl
```

Exercise 4: Assign labels to the gender variable for clarity.

The drop and keep commands are useful and serve a few different purposes:

- The syntax `drop` *`var`* drops variables from the data (columns).
- The syntax `drop if` *`condition`* drops observations meeting those conditions (rows).
- The analog is true for the `keep` command.

These commands do not affect the original .csv file (unless you overwrite it – but even then, there is always a copy on the course website). Always keep original files when creating an analytic data set!

Exercise 5: Drop the variable for insulin status (there is no scientific reason to drop it, but this is just for practice since we are not going to use it in this lab).

Exercise 6: Keep only the observations having a baseline HbA1c of at least 9.0%. How else could you have accomplished this?

Examples of data exploration: We have learned a number of ways to explore and describe data in the first set of notes. There are a couple of other commands that are helpful to know about.

It is possible to characterize degree of missingness in data (for all variables or for specific variables):

```
misstable summarize
misstable summarize var1 var2 (etc.)
```

The tabstat command shows you descriptive statistics (you can specify the statistics to show). This is an example of a command in which the *by* option is available to avoid having to write many "if" statements.

```
tabstat a1c0, stat(mean)
tabstat a1c0, stat(n mean sd p50 p25 p75 min max)
tabstat a1c12, stat(n mean sd p50 p25 p75 min max) by(trtgrp)
```

Exercise 7: Run the above commands and be certain you understand what the numbers are telling you. Pay particular attention to *n*.

Exercises: We have learned a number of ways to conduct statistical tests in the first set of notes. Here, you will have the opportunity to practice implementing these tests. The exercises below pertain only to our defined cohort of individuals with highly uncontrolled baseline HbA1c. Recall that when creating a new variable, you must keep missingness associated with the *old* variable in mind.

Exercise 8: Perform an analysis to evaluate whether the mean six-month HbA1c differs between patients receiving REACH only and patients receiving REACH + FAMS.

Exercise 9: Perform an analysis to evaluate whether the mean change in six-month HbA1c from baseline among control patients differs from zero.

Exercise 10: Perform an analysis to evaluate whether the mean change in six-month HbA1c from baseline differs between patients receiving REACH (that is, REACH alone *or* REACH + FAMS) and patients receiving the control.

Exercise 11: Perform an analysis to evaluate whether the proportion of patients with an absolute reduction in HbA1c exceeding 0.5% differs between patients receiving REACH + FAMS and patients receiving the control.