FRAMES IN STATA

Frames represent a complete shift in how one thinks about coding in Stata. They allow you to seamlessly work with multiple datasets at the same time. Instead of just holding one dataset in memory, Stata will now hold multiple frames in memory. This allows you to work with a large number of variables at the same time while still remaining below the upper limit and not having to store them in different datasets on your computer. It can be incredibly useful in streamlining existing work as well as opening opportunities for working in more efficient but previously not possible ways. This feature was added in Stata 16.

However, there is a fixed cost of learning the syntax, and it can be hard to see the benefits if you've used the one-data-set-only version of Stata for a long time. This short post only covers the basics of frames. There are other commands such as deleting or renaming frames as well as ways of adding observations to frames when doing simulations or similar activities. For more details see this Stata help file.

Using frames — the basics

When you open Stata, you also automatically open a frame, called default. It is important to understand that whenever you're using Stata you're working in a frame.

You can create a new frame called newframename with variables given in varlist using:

```
frame create newframename varlist
```

You can also create a frame by copying an existing frame using frame

```
copy existing_frame new_frame
```

There are three ways to change which frame you're working in. To permanently switch to framename use frame

```
change framename
```

To execute a command on data in framename use frame

```
framename : command
```

To execute a block of code on data in framename use

```
frame framename {
        commands
}
```

You can also make links between frames in a similar fashion to how you would merge datasets. To link the existing frame to a new frame use

```
frlink 1:1 link_variable , frame(link_frame)
```

Now that the frames are linked, you can copy variables from one to the other. To copy newvar from link frame to the current active frame use:

```
frget newvar , from(link_frame)
```

Example

Consider the following example where we import a dataset, create a new frame in which we collapse the dataset, create and plot a variable of importance, and then merge it back into the main frame. Using frames allows us to not have to save the data in a different file and makes it easy to switch between the main and collapsed versions.

We will first load the sample dataset:

```
clear all

sysuse nlsw88.dta
```

Now, we will rename default frame:

```
frame rename default nlsw
```

We will now create graph of wages by industry. However, in order to do this, we will collapse the data by storing it inside a new frame. Note that the previous dataset will not be lost and we can switch back and forth whenever needed. We do not have to store them as separate dta files and merge them but only switch across different frames.

```
frame copy nlsw nlsw_industry_wage

frame nlsw_industry_wage {

      keep wage industry

      collapse (mean) wage , by(industry)

      rename wage av_wage

      graph bar av_wage , over(industry , lab(angle(45))) ///

      graphregion(color(white)) ylab(, angle(0) nogrid)

}
```

Now, we will merge the mean industry wage from the collapsed dataset (stored in the new frame) to the original frame, and then use it in an individual level regression-

```
frlink m:1 industry , frame(nlsw_industry_wage)

frget av_wage , from(nlsw_industry_wage)

reg wage union age av_wage
```

Frames also have other uses. For example, if you have a dataset containing several variables and only want to work with a few at a time, you can store those variables in a new frame, work with them, and then switch back to the old frame and merge these cleaned variables in from the new one.

Luke Milsom, DPhil candidate in Economics, University College

14 October 2021