Introduction to Frames in Stata & Efficient Commands for Data Visualization in Large Datasets

Stata UK Conference, 2023

Jan Kabatek
The University of Melbourne, CentER, IZA, LCC & Netspar

September 7, 2023

Environmental imperative of software efficiency I.

- **Have you ever wondered about the environmental costs of your work?**
  - We spend most of our working days analyzing large quantities of data, which requires **lots of computing power**.
  - Computing power translates into electricity consumption and that translates into **carbon footprint**.
  - But how large is this footprint? And what can we do about it?

Environmental imperative of software efficiency II.

Carbon footprint of computing

According to Stevens *et al.* (2020), the carbon footprint of computing among Australian astronomers is $\sim$ **22 tCO_2e/yr per researcher.**
This is the single highest contributing factor to their net carbon emissions.

- In contrast, the UK average carbon footprint per person is **6-12 tCO_2e/yr**
- Our carbon emissions are bound to differ from those of astronomers, although it is not clear whether they would be much lower.
- Plus, factoring in the much-larger population of Stata users, we are definitely in the zone of non-trivial emissions.

Environmental imperative of software efficiency III.

- **Taking all that into account, we should be putting considerable effort into making our code & software packages efficient!**

  (but by all means, get that compost bin, too...)

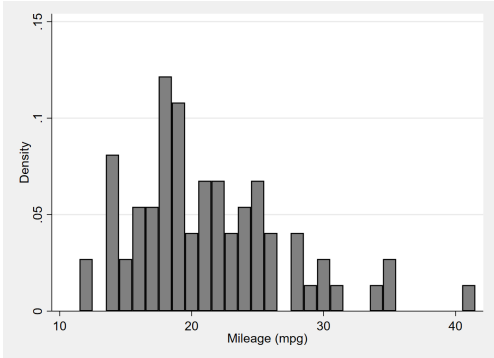
- This presentation serves multiple purposes:

  1. It aims to highlight the environmental imperative of efficiency in statistical computing.
  2. It introduces the concept of **frames** in Stata.
  3. Using my "**plotsuite**" of graphing commands, it illustrates the efficiency improvements that can be achieved in Stata.

Illustrative example I.

Stata code I.

```
sysuse auto
hist mpg
```

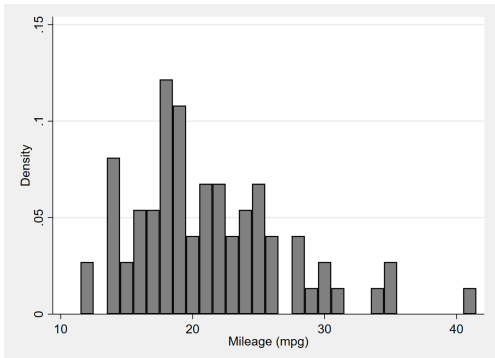


Illustrative example I.

Stata code I.

```
sysuse auto
hist mpg
```

Command 'hist' takes 0.8 secs to run.



Illustrative example I.

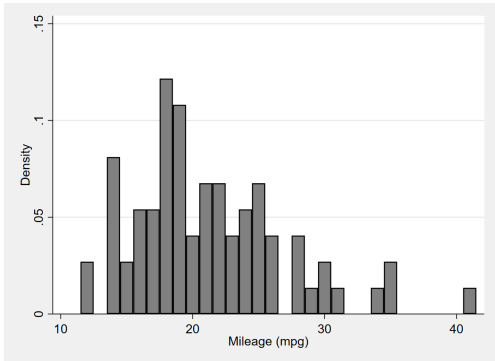
Stata code I.

```
sysuse auto
hist mpg
```

Command 'hist' takes 0.8 secs to run.

Stata code II.

```
sysuse auto
expand 4000000 //4M duplicates
hist mpg
```



Illustrative example I.

Stata code I.

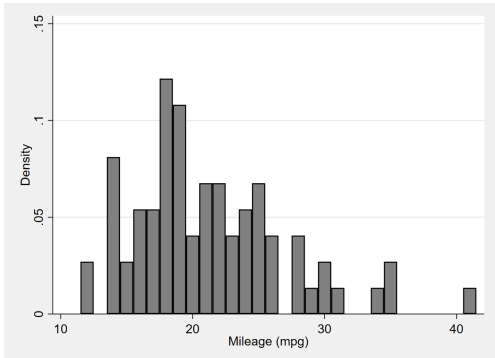
```
sysuse auto
hist mpg
```

Command 'hist' takes 0.8 secs to run.

Stata code II.

```
sysuse auto
expand 4000000 //4M duplicates
hist mpg
```

Now, 'hist' takes 31 mins to run!



Illustrative example II.

- **BUT:** Longer execution time of the second code is **not** fully attributable to the larger dataset requiring more computing time to put the data into histogram bins!
- Rather, it relates to one of the **most pernicious bottlenecks** of native Stata commands.

Illustrative example II.

- **BUT:** Longer execution time of the second code is **not** fully attributable to the larger dataset requiring more computing time to put the data into histogram bins!
- Rather, it relates to one of the **most pernicious bottlenecks** of native Stata commands.
- Command 'hist' does the following:

Illustrative example II.

- **BUT:** Longer execution time of the second code is **not** fully attributable to the larger dataset requiring more computing time to put the data into histogram bins!
- Rather, it relates to one of the **most pernicious bottlenecks** of native Stata commands.
- Command 'hist' does the following:
  1. **preserves** the original dataset

Illustrative example II.

- **BUT:** Longer execution time of the second code is **not** fully attributable to the larger dataset requiring more computing time to put the data into histogram bins!
- Rather, it relates to one of the **most pernicious bottlenecks** of native Stata commands.
- Command 'hist' does the following:
  1. **preserves** the original dataset
  2. calculates the histogram bins

Illustrative example II.

- **BUT:** Longer execution time of the second code is **not** fully attributable to the larger dataset requiring more computing time to put the data into histogram bins!
- Rather, it relates to one of the **most pernicious bottlenecks** of native Stata commands.
- Command 'hist' does the following:
  1. **preserves** the original dataset
  2. calculates the histogram bins
  3. stores them as a **temporary dataset**

Illustrative example II.

- **BUT:** Longer execution time of the second code is **not** fully attributable to the larger dataset requiring more computing time to put the data into histogram bins!
- Rather, it relates to one of the **most pernicious bottlenecks** of native Stata commands.
- Command 'hist' does the following:
  1. **preserves** the original dataset
  2. calculates the histogram bins
  3. stores them as a **temporary dataset**
  4. displays the corresponding graph

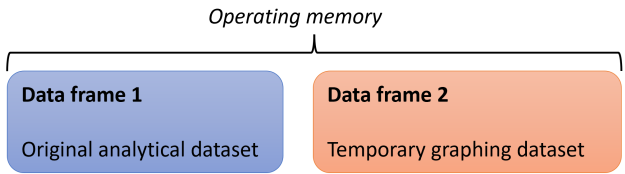
Illustrative example II.

- **BUT:** Longer execution time of the second code is **not** fully attributable to the larger dataset requiring more computing time to put the data into histogram bins!
- Rather, it relates to one of the **most pernicious bottlenecks** of native Stata commands.
- Command 'hist' does the following:
  1. **preserves** the original dataset
  2. calculates the histogram bins
  3. stores them as a **temporary dataset**
  4. displays the corresponding graph
  5. and **restores** the original dataset

Legacy bottlenecks

- Now, **preserving** and **restoring** big datasets takes ages...
- ...and, as of Stata 16, it is wholly redundant!

- Instead of making operations on a single dataset, we can leverage the new environment of **data frames** and hold **both** the original **and** the temporary graph data in memory at the same time (no restoring required).



Basics of frames

1. Stata starts in the *default* frame:

```
sysuse bplong
// we are in a frame called default
```

Basics of frames

1. Stata starts in the *default* frame:

```
sysuse bplong
// we are in a frame called default
```

2. Create a frame called *other* and load another dataset:

```
frame create other
frame change other
sysuse auto
hist mpg
```

Basics of frames

1. Stata starts in the *default* frame:

```
sysuse bplong
// we are in a frame called default
```

2. Create a frame called *other* and load another dataset:

```
frame create other
frame change other
sysuse auto
hist mpg
```

3. Call the same histogram command from the *default* frame

```
frame change default
frame other: hist mpg
```

Basic uses of frames

- Great for running **post-estimation commands** (e-class & r-class) that use preserve and restore workflows (*e.g.*, **outreg2**).
  - Estimate the model, switch to an empty frame and run the commands there!
- Excellent for **merging-in** datasets that require some initial adjustment (pre-merge).
  - Load them into a separate frame, adjust, save as a tempfile, and merge into the default frame (tempfiles are not frame-specific).
- Also great for making **customized & automated** output tables.
  - Load the output table (in .dta format) as a separate frame and adjust as needed.
  - This can prove **invaluable** in Remote Access environments (ABS DataLab) for generating multi-sheet Excel output tables.
    (See https://github.com/jankabatek/replication_FBOE_FFE/)

Advanced frames I.

3-dataset linkage: Data setup

```
// Frame 1 stores people's time-invariant characteristics (Obs = 1k)
frame create FrameFix
frame change FrameFix
use id sex cob using DatPersFixed

// Frame 2 stores their monthly gross wages, disability status & state of
residence in 2020 (Obs =12k)
frame create FrameVar
frame change FrameVar
use id state wage disab using DatPersVari

// Frame 3 stores chars of their states of residence (Obs=52)
frame create FrameSt
frame change FrameSt
use state taxrate democrat using DatState
```

Advanced frames II.

3-dataset linkage II: Run wage regressions!

```
frame change FrameVar
// FrameVar contains wages, disability status & personal + state IDs
frlink m:1 id, frame(FrameFix)
frlink m:1 state, frame(FrameSt)

// construct net wages, accounting for state-level tax rate:
gen netwage = wage - wage*frval(FrameSt,taxrate)
reg netwage disab if frval(FrameFix,sex) == 2

// put variable from one frame into another:
frget TR = taxrate, from(FrameSt)

// using linked variables as regressors does NOT work:
reg netwage disab frval(FrameFix,sex)
```

Advanced frames III.

- Linked structures hold great promise, but it is still early days.
- Regardless, linkages can aid in dataset construction and heterogeneity analyses, allowing us to bypass memory constraints.

- What remains an annoying bottleneck for Stata is that standard visualization commands (histogram, twoway, bar, etc.) have not yet embraced data frames.
  - (they're stuck in the preserve/restore loop because they're not post-estimation commands).

PLOT family of commands

- Downloadable from github.com/jankabatek/plotsuite
- Or from SSC: **ssc install plottabs**

- Commands: **plottabs, plotmeans, plotarea, & plotbetas**
- Aside from the speed improvements, they also operationalize some features I was missing in the standard visualization commands

  - Critically, the commands store the plotted data in a dedicated frame
  - This data frame can store multiple plots at once, thereby enabling visualizations of complex data systems.

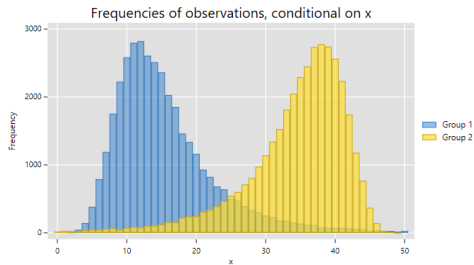
plottabs: one-way and two-way frequency plots

Stata code

```
webuse plotdata, clear
plottabs if gr==1, over(x1) clear
plottabs if gr==2, over(x1) gr(bar)
```

- this example is equivalent to a **twoway** graph combining two histograms with discrete bins, using the option **frequency**

- **plottabs** can also plot conditional rates, similar to the output of `tabulate twoway, row nofreq`



plotmeans: conditional mean plots

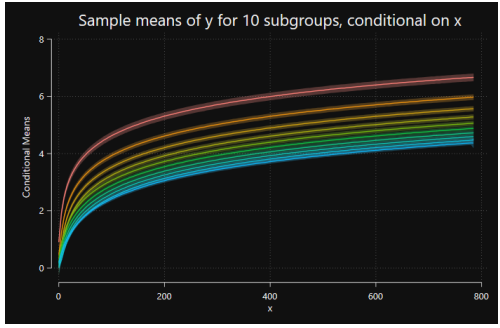
Stata code

```
webuse plotdata, clear
forvalues g = 1/10 {
  plotmeans y if gr10==‘g’, over(x)
}
```

- This example plots means of variable y conditional on a specific value of $x \in [0, 800]$

- Each curve consists of conditional means corresponding to a distinct subgroup of observations g

- Applications: avg wages over time, avg years of education by age, etc.



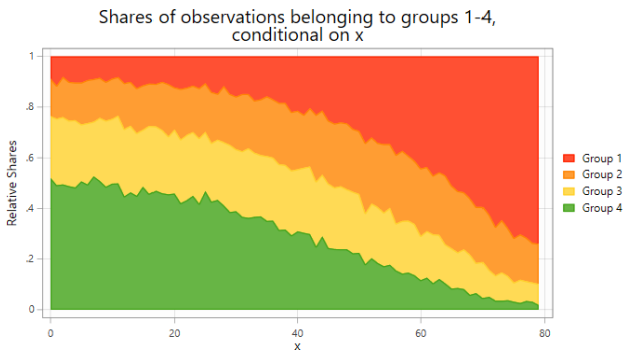
plotarea: conditional share plots

Stata code

```
webuse plotdata, clear
plotarea g, over(x)
```

- This example plots the conditional shares of observations belonging to one of the $g \in [1, 4]$ mutually exclusive categories.

- Applications: highest level of education by age, shares of felonies over time / pop. density, etc.



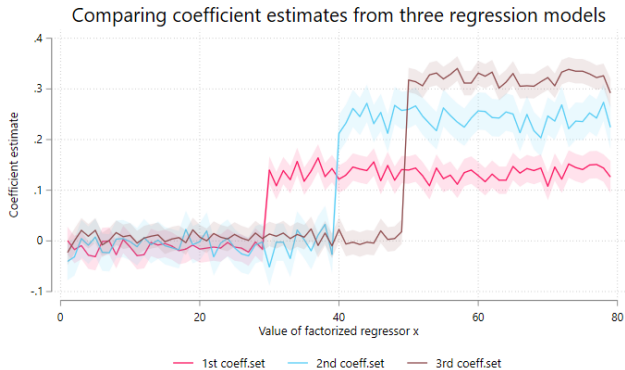
plotbetas: plot sets of coefficient estimates

Stata code

```
webuse plotdata, clear
reg z1 i.x
plotbetas i.x, clear
reg z2 i.x
plotbetas i.x
reg z3 i.x
plotbetas i.x
```

- less verbose alternative to **coefplot**

- Applications: dif-in-dif, RD, heterogeneity analyses, specif. testing & robustness



Benchmarking PLOT against native commands

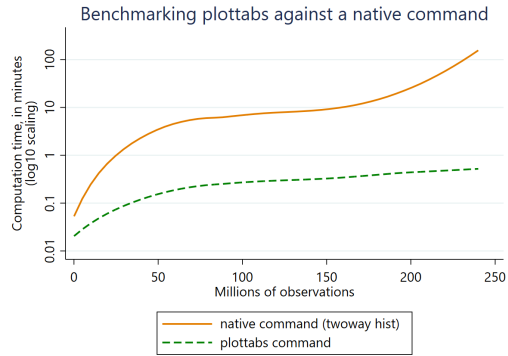
Stata code

```
webuse plotdata, clear
// plottabs
plottabs if gr==1, over(x)
plottabs if gr==2, over(x) gr(bar)

// Twoway native command
twoway (histogram x if gr==1, disc) ///
       (histogram x if gr==2, disc)
```

- the plot comparing the execution times (cond. on sample size) uses the log10 scale

- With large datasets, plottabs finishes **under a minute**, whereas twoway hist takes **almost 2 hrs**



Conclusion

- Efficiency matters! And optimizing popular software packages might just be the most environmentally conscious thing that we (as individuals) will ever do.
- Frames can help a lot with handling large datasets in Stata. They're not perfect just yet, but it's a significant step forward.
- While somewhat quirky, my PLOT commands highlight the efficiency gains that can be reapt by leveraging frames to speed up Stata workflows (other commands can be adjusted in the same way too).

Thank you for your attention!

j.kabatek@unimelb.edu.au
TW: @jankabatek
www.jankabatek.com