Data Organization

`capture` -- https://www.stata.com/manuals13/pcapture.pdf

Why it is useful: Allows subsequent command to be executed while suppressing output (specifically any error messages)

Example: execute a close log command in case you have a log open, without getting an error if you actually don't have an open log

```
capture log close
```

`ds` -- https://www.stata.com/manuals13/dds.pdf

Why it is useful: generates a list of variables matching name patterns.

Example: make a list of all variables in your file, but do not include cpsidp

```
ds cpsidp, not
```

`levelsof` -- https://www.stata.com/manuals/plevelsof.pdf

Why it is useful: creates a list of all unique values of a variable that can be used elsewhere

Example: find all values of YEAR in the data file and save to a local macro

```
levelsof year, local(years_in_file)
```

`local` -- https://www.stata.com/manuals13/pmacro.pdf

Why it is useful: local macros store information that can be used throughout the do file

Example: store the extract number in a local macro near the top of the file so that if your extract number changes, you only have to make the change once at the top of the file. The value stored in the local macro gets inserted in the do file wherever the local macros is called.

```
local extract_num 00200
do cps_`extract_num'.do
```

Is equivalent to:

```
do cps_00200.do
```

`numlabel, add` -- https://www.stata.com/manuals13/dlabelbook.pdf

Why it is useful: prefixes numeric labels to the value labels

Example: see numeric and descriptive values for the variable UNION

```
numlabel, add
tab union
```

`rename` -- https://www.stata.com/manuals13/drename.pdf

Why it is useful: allows use of wildcards to rename variables matching name patterns

Example: rename all variables ending with a number so that they will work with reshape (e.g., rename hrhhid2 to hrhhid2_, etc.)

```
rename var# var#_
```

`scalar` -- https://www.stata.com/manuals13/pscalar.pdf

Why it is useful: stores a single number or string rather than creating a variable attached to every observation in the dataset

Example: identify the maximum age difference allowable between two time points

```
scalar define maxagediff = 2
```

Data Manipulation

`bysort` -- https://www.stata.com/manuals/dby.pdf

Why it is useful: combines by prefix and sort option and can be used with other commands (i.e., gen, egen) to summarize information across records or values on a record.

Example: find an individual's minimum age while in CPS

```
bysort cpsid: egen minage = min(age)
```

Example: number each observation for a person within a linked data file

```
bysort cpsidp: gen _countvar = _n
```

`egen` -- https://www.stata.com/manuals13/degen.pdf

Why it is useful: summarize across records or values on a record

Example: find an individual's minimum age while in CPS

```
sort cpsidp
egen minage=min(age), by(cpsidp)
```

Example: find an individual's maximum age while in CPS

```
sort cpsidp
egen maxage=max(age), by(cpsidp)
```

`foreach` -- https://www.stata.com/manuals13/pforeach.pdf

Why it is useful: allows you to loop through a set of values, variables, etc without having to repeat the code

Example:

```
foreach var in sex race age {
      gen `var'_match=1
      }
```

Is equivalent to

```
gen sex_match=1
gen race_match=1
gen age_match=1
```

`_n and _N` -- https://stats.idre.ucla.edu/stata/seminars/notes/counting-from-_n-to-_n/

Why it is useful: _n and _N are Stata system variables that can be used to generate a unique code for each observation (_n) or to identify the total number of observations (_N). They can also be used to identify these values for an observation within a group of records or values on a record.

Example: Identify the times a person appears within a linked data file

```
by cpsidp, sort: gen _numobs = _n
```

Example: Identify the last observation for a person within a linked data file

```
by cpsidp, sort: gen _finalobs = _N
```

`reshape` -- https://www.stata.com/manuals13/dreshape.pdf

Why it is useful: converts the data from long to wide and vice versa

Example: convert a long CPS extract to wide format

```
reshape wide age sex race mish month year, i(cpsidp)
j(time)
```

**note: time-varying variables (almost all in your data) need to be dropped before performing a reshape or specified as variables you want to reshape*

Weighting

`svyset` and `svy` -- https://www.stata.com/manuals13/svysvytabulateoneway.pdf

Why it is useful: these commands allow you to produce weighted estimates of survey data. Svyset declares the data as survey data while prefixing your commands with svy: produces weighted estimates.

Example: Declare the data as survey data representative of a population using sampling weights (pweights), and estimate tabulations with weighted counts and columns.

```
svyset[pweight=wtfinl]
svy: tab year, count format(%10.0f)
svy: tab year, col row cell
```