Introduction to Computing for Sociologists
Neustadtl

Stata `by` and `egen` commands

by and *bysort*

Many Stata commands can be executed on a group-by-group basis. Earlier we looked at how the Stata `by` command can be used as a prefix for statistical commands (see `help by`). For example the following Stata code will execute the summarize command for each unique value of marital (married, widowed, etc.):

```
by marital: summarize sociability
```

The `by` construct can also be used to create new variables. The `by` prefix steps through each unique value of a variable, and treats each set of observations as a separate dataset. The data must be sorted by values of the by-group variable before using `by`. Suppose we wanted to calculate the minimum value of age for each marital status. We could:

```
sort marital age
by marital: gen minage=age[1]
```

The first line orders all the cases by marital status and within marital status by age. The second line looks only within each marital group and assigns the value of age to the first observation which because of sorting is the minimum value, to the new variable *minage*. The *_n* variable which records the current observation number resets within each by-group, i.e. each group is treated like its own little dataset.

You can combine these two steps with the `bysort` command (abbreviated `bys`):

```
bysort marital (age): gen minage=age[1]
```

where the variables after `bysort` but before the parentheses are the variables you want to perform the command by, and the variables inside the parentheses specify the sort order of each little dataset you are performing the command on.

egen

One of Stata's most powerful and useful commands is `egen`. Like `generate`, it is used to create new variables, but it is much more than that. Using `egen` difficult and tedious variables can be created easily. Some examples are variables whose values are the mean of another variable for each group such as sociability for males and females. You can also use `egen` to create other variables that count the number of observations that fit a certain criteria, or even simply number observations. The only way to truly see how powerful `egen` can be is to show a few examples and then have you explore the other available functions on your own. See `help egen` for a complete listing of all of these commands.

`egen age_cat1 = cut(age), ///`
`   at(18, 25, 35, 45, 55, 65, 75, 90)`
 	
`egen age_cat2 = cut(age), group(6)`
	The `cut` function is useful for collapsing variables. You specify the lowest value for each new group with the `at()` option. Any observations with a value less than 18 will be given a missing value for age_cat, and all observations with a value greater than 90 will be placed in the “90” age_cat group. Or simply specify the number of groups you want with the `group()` option.

`egen sexfreq2 = mean(sexfreq1), by(marital)`
 	
`egen sexfreq3 = mean(sexfreq1), by(sex marital)`
	This example creates a variable that is the mean of *sexfreq1* for each marital group. In addition to the mean, other statistics can be calculated (e.g. min, max, sd).

`egen numobs1 = count(sexfreq1), by(year)`
 	
`egen numobs2 = count(sexfreq1), ///`
`   by(year marital)`
	The `count` function counts the number of non-missing observations of *sexfreq1* within each year creating a constant for each case. The second example counts the non-missing values of *sexfreq1* within year and within marital status.

`egen sexmar=group(sex marital), label`
 	
`egen marsex=group(marital sex), label`
	The `group` function numbers the groups formed by crossing *sex* and *marital*. The groups are numbered consecutively which makes this a good variable to use in analysis. The `label` option causes Stata to use the value labels (if any) of *sex* and *marital*.

`egen sexmarcon=concat(sex marital), punct(/)`	The `concat` function is useful when you have two or more variables that you want to combine to form one variable but adding or multiplying them would not make sense. The `p()` option allows you to put a separator character between the values.

1. Use the `rowmiss()` function to create a new variable called *socmiss* for the sociability component measures (*socrel, socommun, socfrend,* and *socbar*). Use `tabulate` to show the distribution of missing values for these variables
2. Use the `group()` function to create a new categorical variable for every combination of *race* (by) *sex*. Restrict race to white and black respondents (`if` can be used with `egen`). Use graph bar and plot the 25th and 75th percentiles of *sociability* for these categories (look at the `over()` option).
3. Create a new variable *avgsoc* that is the average sociability within *year* and *sex*.