SOC 561: Programing for the Social Sciences
Community Resource – Postfile
Steven Lizzol

A QUICK STATA GUIDE: POSTFILE

Additional Resources:

- Stata Online Help Documentation
- The Stata Journal – Stata Tip 54: Post your results
- Introduction to postfile commands in Stata (Jang 2016)
- Introduction to postfile in Stata (Abromaviciute 2014)

Postfile: What is it?

Post is a command structure in Stata that allows the user to pull specific observations or statistical estimations from a dataset and store that information in a separate file. For example, post will allow a researcher to create a subset of desired observations. Additionally, a user might implement the post command to produce different statistical estimations based on certain desired parameters and store those estimations in a different file. This guide will first provide an overview of the postfile command structure followed by different examples.

Basic Command Template

To ensure you perform the post command accurately each time, it is recommended to store a command template, one that you can reference whenever a situation lends itself to the use of postfile. Here is a very basic postfile template:

```
1. tempname memhold
2.     postfile `memhold' var_name1 str10 var_name2… using "file_name.dta", replace
3.     post `memhold' (var_name1_type) (var_name1_type)
4. postclose `memhold'
```

Let's discuss what each part of the template means.

Line 1: Here temporary object named "`memhold`" is created to store the specific information you are trying to gather. For consistency, it is recommended that you always use this same syntax. Deviations from "`memhold`" are allowed; however, the user must make sure and use the same title throughout the command to avoid unwanted errors.

Line 2: This dictates the desired new variables within the new file. The `postfile `memhold'` specifies the temporary file from line 1, while "`var_name1`" (a numeric variable) and "`str10 var_name2`" (a string variable) addressing the variables to be listed in the file. Notice that the format of the second variable is identified as a string with the name of the variable after the "`str10`" format designation. Stata will automatically assume a numeric variable, therefore a format designator is not needed for the first variable and required for the second. The last part of this line identifies the new name of the file that will be created; "replace" may be good to use here given the trial and error aspect of writing more complex code. However, be mindful before you run the code of overwriting any files you desire to keep.

Line 3: Here an observation is posted to the new dataset using the desired variables from line 2. The data for each variable is placed in parentheses in the order dictated in line 2.

Basic Command Example

Using the Stata data file auto.dta, we will go through a quick example to what type of outputs are created with the basic commands.

```
. sysuse auto
(1978 Automobile Data)

. list
```

	make	price
1.	AMC Concord	4,099
2.	AMC Pacer	4,749
3.	AMC Spirit	3,799
4.	Buick Century	4,816
5.	Buick Electra	7,827

What results will we get if we run the following code?:

```
tempname memhold
    postfile `memhold' price str18 make using "auto_post.dta", replace
    post `memhold' (price) (make)
postclose `memhold'
```

First, let's examine what is being requested. The second line addresses the creation of the numeric variable "price" and the string variable "make" and saving that information to a file called "auto_post.dta." The third line tells Stata to pull data from the variable "price" and "make" to populate the identified variables in line two. Stata will do this for the first observation for each variable providing the following output:

```
. use auto_post.dta

. list
```

	price	make
1.	4099	AMC Concord

Capturing Statistical Estimations

The above example is for demonstration purposes only and should not represent the typical type of outputs desired from the postfile command. Ideally the command should be used to store multiple estimates from a statistical model. To further investigate this function using the auto.dta dataset, let's say we were interested in regressing price (price) on miles per gallon (mpg), the car's weight (weight), and whether or not the model possessed a foreign label (foreign). A Stata regression command will provide the following results:

```
. reg price mpg weight foreign
```

Source	SS	df	MS	Number of obs	=	74
				F(3, 70)	=	23.29
Model	317252881	3	105750960	Prob > F	=	0.0000
Residual	317812515	70	4540178.78	R-squared	=	0.4996
				Adj R-squared	=	0.4781
Total	635065396	73	8699525.97	Root MSE	=	2130.8

price	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
mpg	21.8536	74.22114	0.29	0.769	-126.1758	169.883
weight	3.464706	.630749	5.49	0.000	2.206717	4.722695
foreign	3673.06	683.9783	5.37	0.000	2308.909	5037.212
_cons	-5853.696	3376.987	-1.73	0.087	-12588.88	881.4934

We could store both the beta (_b) and the standard error (_se) for each independent variable in the model using the post command:

```
tempname memhold
    postfile `memhold' mpg_b mpg_se weight_b weight_se foreign_b foreign_se using "auto_post2.dta", replace
    reg price mpg weight foreign
    post `memhold' (_b[mpg]) (_se[mpg]) (_b[weight]) (_se[weight]) (_b[foreign]) (_se[foreign])
postclose `memhold'
```

Notice for the six variable names in command line two, we have six corresponding data requests in command line four that captures the desired betas and standard errors for the independent variables. A new file "auto_post2.dta" becomes available after running the command and provides the following output:

```
. use auto_post2.dta

. list
```

	mpg_b	mpg_se	weight_b	weight~e	foreig~b	foreig~e
1.	21.8536	74.22114	3.464706	.630749	3673.06	683.9783

These are the exact same betas and standard errors from the previous regression command.

Capturing Statistical Estimations with Loops

Given that the postfile command has the ability to pull estimations from a statistical model, the addition of a loop allows postfile to store multiple estimates from multiple model specifications. For example, let's say we were interested in running the previous model for the different categories of repair records using the "rep78" variable. This variable classifies each car into one of five categories. For this example, we are only interested in gathering statistical estimates from cars identified in the 3, 4, and 5 repair categories. To do this without using postfile, we would embark on a tedious track of writing separate model codes for each classification. Rather than running the models separately, we can simplify the process by using a loop and capturing the

estimates through the postfile command. In addition to collecting the different betas and standard errors, we also desire to capture levels of significance for each beta within each distinct model. Stata will not pull p-values like betas or standard errors. To do so we need to set up a local and a nested loop to collect the desired levels of significance within each model:

```
tempname memhold
postfile `memhold' repair_cat mpg_b mpg_se mpg_p weight_b weight_se weight_p foreign_b foreign_se ///
foreign_p using "auto_post3.dta", replace
forval j=3(1)5 {
    local i = 0
    reg price mpg weight foreign if rep78 == `j'
        foreach var in mpg weight foreign {
            local i = `i' + 1
            test `var'
            local p`i' = r(p)
        }
    post `memhold' (`j') (_b[mpg]) (_se[mpg]) (`p1') (_b[weight]) (_se[weight]) (`p2') (_b[foreign]) ///
    (_se[foreign]) (`p3')
}
postclose `memhold'
```

After running the above command, the "auto_post3.dta" file is created with the following estimates:

```
. use auto_post3

. list
```

	repair~t	mpg_b	mpg_se	mpg_p	weight_b	weight~e	weight_p	foreign_b	foreig~e	foreig~p
1.	3	172.7148	167.2526	.3112744	5.472117	1.071669	.0000254	5310.45	1668.798	.0037651
2.	4	165.4922	167.2336	.339172	2.638209	1.020408	.0215798	2807.545	999.851	.0139598
3.	5	101.109	92.72512	.3116359	7.868431	2.115531	.0074623	-827.2197	1158.37	.498265

Comparing the newly created Stata file with the last regression output reveals the estimates for the model specified on the "5" repair category are accurate.

Source	SS	df	MS	Number of obs	=	11
				F(3, 7)	=	11.26
Model	56676683.8	3	18892227.9	Prob > F	=	0.0045
Residual	11745464.2	7	1677923.46	R-squared	=	0.8283
				Adj R-squared	=	0.7548
Total	68422148	10	6842214.8	Root MSE	=	1295.3

price	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
mpg	101.109	92.72512	1.09	0.312	-118.151	320.3691
weight	7.86843	2.115531	3.72	0.007	2.865995	12.87087
foreign	-827.2197	1158.37	-0.71	0.498	-3566.33	1911.891
_cons	-14453.11	6913.032	-2.09	0.075	-30799.83	1893.61

A further check of the other regression outputs (not shown here) shows that the other two rows of model estimations are accurate as well.

Review Questions

1. Running each of the following post codes in Stata will produce an error. What needs to change so the code can run?

    a. "length" is a numeric variable and "category" is a string variable

```
tempname memhold
    postfile `memhold' length category using "new_post.dta", replace
    post `memhold' (length) (category)
postclose `memhold'
```

    b.

```
tempname memhold
    postfile `memhold' mpg_b mpg_se weight_b weight_se foreign_se using "auto_post1.dta", replace
    reg price mpg weight foreign
    post `memhold' (_b[mpg]) (_se[mpg]) (_b[weight]) (_se[weight]) (_b[foreign]) (_se[foreign])
postclose `memhold'
```

    c.

```
tempname memhold
postfile `memhold' repair_cat mpg_b mpg_se weight_b weight_se foreign_b foreign_se using "auto_post3.dta", replace
forval j=3(1)5 {
    reg price mpg weight foreign if rep78 == `j'
    post memhold (`j') _b[mpg] _se[mpg] _b[weight] _se[weight] _b[foreign] _se[foreign]
}
postclose `memhold'
```

2. By now you should have a good understanding why the post command is a useful tool. Now, it's your turn! Open up a dataset and try to create a new file using postfile to store some estimates that interest you. Did you get any errors? How did you resolve them?

Answers:

1a. Since "category" is a string variable, it needs to be identified as such in line 2 (e.g., `str12 category`).

1b. The number of variables identified for the storage file does not match the number of estimations we are requesting for the model. A "`foreign_b`" variable needs to be place before "`foreign_se`" in line 2.

1c. The error occurs in line 5 of the code. First "`memhold`" lacks the correct prefix and suffix designations (e.g., "`` `memhold' ``"). Second, the estimations do not have parentheses around them and will result in an error.