The Stata Journal (2023)
23, Number 2, pp. 589–594 DOI: 10.1177/1536867X231175349

Stata tip 152: if and if: When to use the if qualifier and when to use the if command

Nicholas J. Cox
Department of Geography
Durham University
Durham, U.K.
n.j.cox@durham.ac.uk

Clyde B. Schechter
Albert Einstein College of Medicine
Bronx, NY
clyde.schechter@einsteinmed.edu

1 Introduction

Stata has an `if` qualifier and an `if` command. Here we discuss generally when you should use either and specifically flag a common pitfall in using the `if` command. In a nutshell, the pitfall arises from confusing the two constructs: the `if` command does not loop over the data but, at most, looks in the first observation of a dataset. There has long been a StataCorp FAQ on this topic (Wernow 2005), but we and others have usually tried to explain matters otherwise. This tip is intended as a more durable version of the story that should be easier to find than occasional Statalist postings that are vivid when read but hard to find later.

2 The if qualifier

The `if` qualifier is met by most users early in their Stata experience. Its purpose is to select observations (cases, records, or rows in the dataset) for some action. Thus, you could run the following commands to read in a dataset and first `summarize` a variable and then `summarize` that variable again for a subset of observations. Here we suppress the results, but if you are new to Stata and unfamiliar with `summarize`, it would be worth your time to run the code yourself to find out about a valuable command.

```
. sysuse auto
. summarize mpg
. summarize mpg if foreign == 1
```

When the `if` qualifier is used (or, in other words, when an `if` condition is specified), Stata tests the expression given—here `foreign == 1`—in each observation to see whether it is satisfied (is true) in that observation. Observations for which the expression is true are selected for the action. In this example, `foreign` is an indicator variable that is 1 if a car is foreign (made outside the United States) and 0 if a car is domestic (made inside the United States). The operator `==` tests for equality, noting that in Stata the `=` operator typically indicates assignment of a value or values, say, to a variable. Out of 74 cars, 22 qualify as being foreign, so their observations will be `summarize`d for the variable `mpg`.

Stata follows a very widely used convention, running across statistics, mathematics, and computing, that in logical tests, a value of 1 means true and a value of 0 means false. In fact, Stata's rule is more general: Any numeric value that is not 0 means true, while only the numeric value 0 means false. Watch out with missing values because any numeric value that represents missing (whether system missing, `.`, or extended missing values from `.a` to `.z`) is certainly not 0 and so yields true in a logical test.

Logical tests in Stata take two forms. First, and more commonly, some logical operator is used in an expression. Tests for equality, using the `==` operator, may be what you need; otherwise, some test for inequality may be needed. See the help for operators to see the complete list. Thus, in `auto.dta` you could select cars with high `mpg` by, say, `mpg > 25`. Logical tests can combine two or more conditions, but even so the keyword `if` appears only once in any comparison.

Second, you can ask Stata to look inside a numeric variable and check whether its values are 0 or not. In `auto.dta`, `foreign` is only ever 1 or 0 and never missing. So the test `if foreign` is precisely the same test in practice as `if foreign == 1`. Presented with `if foreign`, Stata looks inside the variable and selects those observations for which it is not 0, which in practice is the same subset of observations as those for which the condition `if foreign == 1` is true.

There are positive and negative sides to this flexibility. The positive side is that we can write Stata code that may appeal to readers as idiomatic in their own language and in Stata too. "Let's focus on the cars that are foreign" becomes the condition `if foreign`. Such coding works best if you follow a convention, which we strongly recommend, of naming an indicator variable for the condition coded as 1. That is precisely what the developers of Stata did at the very beginning when coding up the auto data.

The negative side is that the inclusiveness here could bite if there are nonzero values that the condition `if foreign` would catch too, even though that is not what you intend. As said, nonzero values include any numeric missing values. So you might well prefer to be safe rather than succinct and always spell out, say, `if foreign == 1`.

For more on truth and falsity in Stata, see Cox (2005, 2016). For more on indicator variables, see Cox and Schechter (2019), especially if you have been thinking "Don't you mean dummy variables?" (Yes, we do.)

3 The if command

The previous section may have strengthened your understanding of the `if` qualifier, say, by spelling out some nuances. At this point in the story, the most important detail about the `if` command is that it is emphatically not a way to do the same thing differently. Oddly, or otherwise, a misunderstanding that the two are equivalent (or at least overlap in what they do) seems to arise most often with people new to Stata who are accustomed to programming in some other language. Such programmers may guess or hope that Stata's `if` command is similar to, or an extension of, what they know already.

Whatever the explanation, constructs using `if` or some equivalent keyword have been present in many programming languages over several decades. Examples can be found in Sammet (1969), Kernighan and Plauger (1978), and Bal and Grune (1994).

We will pursue this negative theme before turning to when and why the `if` command is appropriate or useful. Otherwise, there would be no point to including it within Stata.

Any puzzlement is intensified whenever Stata allows use of the `if` command in a way that seems equivalent to use of the `if` qualifier. It then gives results that occasionally are what you want but more often just seem bizarre. As examples, consider these two statements and their results:

```
. if foreign == 1 summarize mpg
. if foreign == 0 summarize mpg
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
         mpg |         74     21.2973    5.785503         12         41
```

Stata complains about neither statement, so each is perfectly legal. But you might even wonder whether you have unearthed a bug. The first statement yields no results, whereas we already know that there are observations for which `foreign == 1`. Other way round, the second statement yields results, but if you look carefully, you will see that the results are for the entire dataset and so include both foreign and domestic cars.

The explanation is immediate given one extra piece of information. When an `if` command refers to a variable (or variables) in the dataset, Stata looks only in the first observation. It is exactly as if you wrote `if foreign[1] == 1` or `if foreign[1] == 0`. It so happens that the first statement is false and the second statement is true, as can be checked independently by looking at the data with, say, `list in 1` or `edit in 1` or `display foreign[1]`. Because the first statement was false, Stata did not execute the next command, `summarize mpg`. Because the second statement was true, Stata did execute the (same) next command. In both cases, the subset of observations specified was not part of the syntax for the next command.

We could make that plainer by writing the same syntax using curly brackets or braces:

```
if foreign[1] == 0 {
    summarize mpg
}
```

Backing up slightly: Here a so-called subscript such as `[1]` attached to a variable name indicates an observation number, so in another example `foreign[42]` would be the value of `foreign` in observation 42. We say "subscript" as an allusion to mathematical notation such as y_1 or y_{42}, but naturally writing *sub scriptum*, below the line, is not strictly possible in Stata.

A more general point to emphasize is that there is no sense in Stata in which the `if` command iterates or loops over the observations in the dataset. (Here we are assuming that there are data in memory; it is perfectly possible to use Stata with no variables in memory, and you may wish to think through what could be done depending on what else is allowed.) Positively put, the `if` command makes one and only one decision, depending on whether the condition specified is true.

The `if` command is very widely used within do-files and within programs, including within programs that define other commands.

There are many examples within Stata programs. Options are typically implemented in this way. In many commands, there are optional choices, either for extra actions or to vary some action from the default. Inside the command code, there is typically a switch for each option whereby different code is executed. The `summarize` command has options, such as `meanonly` (to do less than the default) or `detail` (to do more). That command is built in, so users may not see the internal code, but very many commands are implemented through ado-code, so much of or all the code is visible. If you are curious, you can look inside ado-code with, say,

```
. viewsource tabstat.ado
```

and you will immediately see a series of switches all using the `if` command to set up calculations according to whatever a user did (or did not) specify when issuing the `tabstat` command.

Another common sequence within ado-code is something like this.

```
. marksample touse
. count if `touse'
  74
. if r(N) == 0 error 2000
```

Here `marksample` has the job of creating a temporary indicator variable `` `touse' `` that is 1 when observations are to be used and 0 otherwise. (If the name `touse` looks odd to you, think "to use".) Exclusions arise for one of two reasons: whenever missing values make the use of observations impossible or whenever an `if` qualifier (there it is again) or an `in` qualifier excludes observations by implication. We then `count` the

observations to be used. The result is left in `r(N)`. If that result is 0, then there are no observations to use, which here and usually is regarded as an error. If, as it were, no news is good news, such as when we are checking for something bad but fail to find it, then the syntax would be different. We might well condition on, say, `r(N) > 0`.

There are other vital differences between the `if` qualifier and the `if` command, beyond the cosmetic (but still crucial) difference that the first follows and the second precedes associated code.

The `if` command can be associated with code following `else` to indicate what should be done if the condition specified is false. Indeed, a more or less complicated series of branching decisions may be needed depending on a menu of possible choices. Again, if you are curious, look at the results of

```
. viewsource duplicates.ado
```

which show a series of branches aimed at identifying the subcommand that a user specified after the command itself, such as `duplicates report` or `duplicates list`.

Lest you think that the `if` command is primarily of interest to Stata programmers, let's look at an example of its use in a common situation that arises in data analysis. Suppose you want to analyze some panel data, performing some specific calculations separately in each panel but only in those panels that offer a minimum sample size. Here we assume for simplicity that firms have distinct numeric identifiers. The code in your do-file might look like this:

```
generate abnormal_return = .
levelsof firm, local(firms)
foreach f of local firms {
    count if firm == `f'                                // N.B. if qualifier
    if r(N) >= 30 {                                     // N.B. if command
        regress return market_return if firm == `f'     // if qualifier
        predict resid, resid
        replace abnormal_return = resid if firm == `f' // if qualifier
        drop resid
    }
}
```

Notice that both the `if` command and the `if` qualifier are used in this code, with very different effects. The `if` qualifier applies only to the single command in which it appears, and it restricts those commands to the observations for which `firm == `f'`. The `if` command appears only once in the code, but it controls execution of the following four commands; they are executed only if the result of the preceding `count` command is at least 30. Note, in particular, that this `if` command does not examine any observations in the data in memory: it refers only to the result returned by the preceding `count` command. Note also the use of curly braces to apply the single `if` command to an entire block of commands. Those four commands are all executed, or none are, depending on the available sample size for the firm.

You may be thinking of refinements, such as counting observations with nonmissing values, because observations with missing values are of no use for any regression. You

may also know of community-contributed commands in this area, but discussing those is beyond our scope here. Even if you have access to such commands, understanding the principles in this last example is valuable in many contexts.

References

Bal, H. E., and D. Grune. 1994. *Programming Language Essentials*. Wokingham: Addison–Wesley.

Cox, N. J. 2005. FAQ: What is true or false in Stata? https://www.stata.com/support/faqs/data-management/true-and-false/.

———. 2016. Speaking Stata: Truth, falsity, indication, and negation. *Stata Journal* 16: 229–236. https://doi.org/10.1177/1536867X1601600117.

Cox, N. J., and C. B. Schechter. 2019. Speaking Stata: How best to generate indicator or dummy variables. *Stata Journal* 19: 246–259. https://doi.org/10.1177/1536867X19830921.

Kernighan, B. W., and P. J. Plauger. 1978. *The Elements of Programming Style*. New York: McGraw–Hill.

Sammet, J. E. 1969. *Programming Languages: History and Fundamentals*. Englewood Cliffs, NJ: Prentice-Hall.

Wernow, J. 2005. FAQ: I have an if or while command in my program that only seems to evaluate the first observation. What's going on? http://www.stata.com/support/faqs/programming/if-command-versus-if-qualifier/.