Title

stata.com

classutil — Class programming utility

Description

If you have not yet read [P] **class**, please do so. `classutil` stands outside the class system and provides utilities for examining and manipulating what it contains.

`classutil drop` drops the specified top-level class instances from memory. To drop all class objects, type `discard`; see [P] **discard**.

`classutil describe` displays a description of an object.

`classutil dir` displays a list of all defined objects.

`classutil cdir` displays a directory of all classes available.

`classutil which` lists which `.class` file corresponds to the class specified.

Syntax

Drop class instances from memory

`classutil drop` *instance* [*instance* [...]]

Describe object

`classutil describe` *object* [`, recurse newok`]

List all defined objects

`classutil dir` [*pattern*] [`, all detail`]

Display directory of available classes

`classutil cdir` [*pattern*]

List .class file corresponding to classname

`classutil which` *classname* [`, all`]

where

object, *instance*, and *classname* may be specified with or without a leading period.

instance and *object* are as defined in [P] **class**: *object* is an *instance* or a *classname*.

pattern is as allowed with the `strmatch()` function: `*` means that 0 or more characters go here, and `?` means that exactly one character goes here.

Command `cutil` is a synonym for `classutil`.

Options for classutil describe

recurse specifies that classutil describe be repeated on any class instances or definitions that occur within the specified object. Consider the case where you type classutil describe .myobj, and myobj contains myobj.c0, which is a coordinate. Without the recurse option, you will be informed that myobj.c0 is a coordinate, and classutil describe will stop right there.

With the recurse option, you will be informed that myobj.c0 is a coordinate, and then classutil describe will proceed to describe .myobj.c0, just as if you had typed "classutil describe .myobj.c0". If .myobj.c0 itself includes classes or class instances, they too will be described.

newok is relevant only when describing a class, although it is allowed—and ignored—at other times. newok allows classes to be described even when no instances of the class exist.

When asked to describe a class, Stata needs to access information about that class, and Stata knows the details about a class only when one or more instances of the class exist. If there are no instances, Stata is stuck—it does not know anything other than a class of that name exists. newok specifies that, in such a circumstance, Stata may temporarily create an instance of the class by using .new. If Stata is not allowed to do this, then Stata cannot describe the class. The only reason you are being asked to specify newok is that in some complicated systems, running .new can have side effects, although in most complicated and well-written systems, that will not be the case.

Options for classutil dir

all specifies that class definitions (classes) be listed, as well as top-level instances.

detail specifies that a more detailed description of each of the top-level objects be provided. The default is simply to list the names of the objects in tabular form.

Option for classutil which

all specifies that classutil which list all files along the search path with the specified name, not just the first one (the one Stata will use).

Remarks and examples

stata.com

Remarks are presented under the following headings:

classutil drop

classutil drop may be used only with top-level instances, meaning objects other than classes having names with no dots other than the leading dot. If .mycoord is of type coordinate (or of type double), it would be allowed to drop .mycoord but not coordinate (or double). Thus each of the following would be valid, assuming that each is not a class definition:

```
. classutil drop .this
. classutil drop .mycolor
. classutil drop .this .mycolor
```

The following would be invalid, assuming that `coordinate` is a class:

```
. classutil drop coordinate
```

There is no need to drop classes because they are automatically dropped when the last instance of them is dropped.

The following would not be allowed because they are not top-level objects:

```
. classutil drop .this.that
. classutil drop .mycolor.color.rgb[1]
```

Second-, third-, and higher-level objects are dropped when the top-level objects containing them are dropped.

In all the examples above, we have shown objects identified with leading periods, as is typical. The period may, however, be omitted.

```
. classutil drop this mycolor
```

❑ Technical note

Stata's graphics are implemented using classes. If you have a graph displayed, be careful not to drop objects that are not yours. If you drop a system object, Stata will not crash, but `graph` may produce some strange error messages. If you are starting a development project, it is best to `discard` (see [P] **discard**) before starting—that will eliminate all objects and clear any graphs. This way, the only objects defined will be the objects you have created.

❑

classutil describe

`classutil describe` presents a description of the object specified. The object may be a class or an instance and may be of any depth. The following are all valid:

```
. classutil describe coordinate
. classutil describe .this
. classutil describe .color.rgb
. classutil describe .color.rgb[1]
```

The object may be specified with or without a leading period; it makes no difference.

Also see above the descriptions of the `recurse` and `newok` options. The following would also be allowed:

```
. classutil describe coordinate, newok
. classutil describe line, recurse
. classutil describe line, recurse newok
```

classutil dir

`classutil dir` lists all top-level instances currently defined. Note the emphasis on instances: class definitions (*classes*) are not listed. `classutil dir, all` will list all objects, including the class definitions.

If the `detail` option is specified, a more detailed description is presented, but it is still less detailed than that provided by `classutil describe`.

pattern, if specified, is as defined for Stata's `strmatch()` function: `*` means that 0 or more characters go here, and `?` means that exactly one character goes here. If *pattern* is specified, only top-level instances or objects matching the pattern will be listed. Examples include

```
. classutil dir
. classutil dir, detail
. classutil dir, detail all
. classutil dir c*
. classutil dir *_g, detail
```

classutil cdir

`classutil cdir` lists the available classes. Without arguments, all classes are listed. If *pattern* is specified, only classes matching the pattern are listed:

```
. classutil cdir
. classutil cdir c*
. classutil cdir coord*
. classutil cdir *_g
. classutil cdir color_?_?_*
```

pattern is as defined for Stata's `strmatch()` function: `*` means that 0 or more characters go here, and `?` means that exactly one character goes here.

`classutil cdir` obtains the list by searching for `*.class` files along the ado-path; see [P] **sysdir**.

classutil which

`classutil which` identifies the `.class` file associated with class *classname* and displays lines from the file that begin with `*!`. For example,

```
. classutil which mycolortype
C:\ado\personal\mycolortype.class
*! version 1.0.1
. classutil which badclass
file "badclass.class" not found
r(611);
```

`classutil which` searches in the standard way for the `.class` files, that is, by looking for them along the ado-path; see [P] **sysdir**.

With the `all` option, `classutil which` lists all files along the search path with the specified name, not just the first one found (the one Stata would use):

```
. classutil which mycolortype, all
C:\ado\personal\mycolortype.class
*! version 1.0.1
C:\ado\plus\m\mycolortype.class
*! version 1.0.0
```

`*!` lines have to do with versioning. `*` is one of Stata's comment markers, so `*!` lines are comment lines. `*!` is a convention that some programmers use to record version or author information. If there are no `*!` lines, then only the filename is listed.

Stored results

`classutil drop` returns nothing.

`classutil describe` returns macro `r(type)` containing `double`, `string`, *classname*, or `array` and returns `r(bitype)` containing the same, except that if `r(type)=="`*classname*`"`, `r(bitype)` contains `class` or `instance`, depending on whether the object is the definition or an instance of the class.

`classutil cdir` returns in macro `r(list)` the names of the available classes matching the pattern specified. The names will not be preceded by a period.

`classutil dir` returns in macro `r(list)` the names of the top-level instances matching the pattern specified as currently defined in memory. The names will be preceded by a period if the corresponding object is an instance and will be unadorned if the corresponding object is a class definition.

`classutil which` without the `all` option returns in `r(fn)` the name of the file found; the name is not enclosed in quotes. With the `all` option, `classutil which` returns in `r(fn)` the names of all the files found, listed one after the other and each enclosed in quotes.

Also see

[P] **class** — Class programming