Title

stata.com

version — Version control

Syntax Description Option Remarks and examples Also see

Syntax

Show version number to which command interpreter is set

```
version
```

Set command interpreter to version #

```
version # [ , born(ddMONyyyy) ]
```

Execute command under version #

```
version # [ , born(ddMONyyyy) ]: command
```

Description

In the first syntax, `version` shows the current internal version number to which the command interpreter is set.

In the second syntax, `version` sets the command interpreter to internal version number #. `version #` is used to allow old programs to run correctly under more recent versions of Stata and to ensure that new programs run correctly under future versions of Stata.

In the third syntax, `version` executes *command* under version # and then resets the version to what it was before the `version #:...` command was given.

For information about external version control, see [R] **which**.

Option

`born(`*ddMONyyyy*`)` is rarely specified and indicates that the Stata executable must be dated *ddMONyyyy* (for example, 13Jul2009) or later. StataCorp and users sometimes write programs in ado-files that require the Stata executable to be of a certain date. The `born()` option allows us or the author of an ado-file to ensure that ado-code that requires a certain updated executable is not run with an older executable.

Generally all that matters is the version number, so you would not use the `born()` option. You use `born()` in the rare case that you are exploiting a feature added to the executable after the initial release of that version of Stata. See `help whatsnew` to browse the features added to the current version of Stata since its original release.

Remarks and examples

stata.com

version ensures that programs written under an older release of Stata will continue to work under newer releases of Stata. If you do not write programs and if you use only the programs distributed by StataCorp, you can ignore version. If you do write programs, see [U] 18.11.1 Version for guidelines to follow to ensure compatibility of your programs with future releases of Stata.

❑ Technical note

When Stata is invoked, it sets its internal version number to the current version of Stata, which is 13.0 as of this writing. Typing version without arguments shows the current value of the internal version number:

```
. version
version 13.0
```

One way to make old programs work is to set the internal version number interactively to that of a previous release:

```
. version 9.0
. version
version 9.0
```

Now Stata's default interpretation of a program is the same as it was for Stata 9.0.

You cannot set the version to a number higher than the current version. For instance, because we are using Stata 13.0, we cannot set the version number to 13.7.

```
. version 13.7
this is version 13.0 of Stata; it cannot run version 13.7 programs
  (output omitted)
r(9);
```

❑

❑ Technical note

We strongly recommend that all ado-files and do-files begin with a version command. For programs (ado-files), the version command should appear immediately following the program command:

```
program myprog
        version 13.0
        (etc.)
end
```

❑

❑ Technical note

Version control for all random-number generators is specified at the time the set seed command is given, not at the time the random-number generation function such as rnormal() is used. For instance, typing

```
. (assume version is set to be 11.2 or later)
. set seed 123456789
. any_command ...
```

causes *any_command* to use the modern version of `rnormal()` even if *any_command* is an ado-file containing an explicit `version` statement setting the version to less than 11.2. This occurs because the version of `rnormal()` that is used was determined at the time the seed was set, and the seed was set under version 11.2 or later.

This works in both directions. Consider

```
. version 11.1: set seed 123456789
. any_command ...
```

In this case, *any_command* uses the older version of `rnormal()` because the seed was set under version 11.1, before `rnormal()` was updated. *any_command* uses the older version of `rnormal()` even if *any_command* itself includes an explicit `version` statement setting the version to 11.2 or later.

Thus both older and newer ado-files can use the newer or older `rnormal()`, and they can do so without modification. The only case in which you need to modify a do-file or ado-file is when it is older, it contains `set seed`, and you now want it to use the new `rnormal()`. In that case, find the `set seed` command in the do-file or ado-file,

```
version 10              // for example
...
set seed 123456789
...
```

and change it to read

```
version 10              // for example
...
version 11.2: set seed 123456789
...
```

You need to change only the one line.

Everything written above about prefixing `set seed` with a `version` is irrelevant if you are restoring the seed to a state previously obtained from `c(seed)`:

```
set seed X075bcd151f123bb5159a55e50022865700023e53
```

The string state `X075bcd151f123bb5159a55e50022865700023e53` includes the version number at the time the seed was set. Prefixing the above with `version`, whether older or newer, will do no harm but is unnecessary. The version number currently in effect for random-number generators when `set seed` was called is available in `c(version_rng)`; see [P] **creturn**. This is different from the main Stata version number. `c(version_rng)` is only changed when there is a change in Stata's random-number generator. The last change to it was in Stata 12.1, so `c(version_rng)` currently returns `12.1`.

❑

For an up-to-date summary of version changes, see `help version`.

Also see

[P] **display** — Display strings and values of scalar expressions

[R] **which** — Display location and version for an ado-file

[U] **18.11.1 Version**