The Stata Journal (*yyyy*) *vv*, Number *ii*, pp. 1–11

Project Management with Working Directories

Scott Long
Indiana University
Bloomington, IN/USA
jslong@indiana.edu

Draft 2017-08-09

Note: `search changewd` to install commands; this will be changed to `search changewd`.

Abstract. REVISE: Changing working directories is cumbersome in Stata, yet having a different working directory for each project is critical for an efficient workflow. This article describes the `cdsave`, `cdlist`, `cdinfo`, and `cddelete` commands that create and manage commands for quickly moving among working directories.

Keywords: st0001, `cdsave`, `cdlist`, `cdinfo`, `cddelete`, `cd`, working directory, managing projects

Unless you work on only one project, having separate working directories for each project is critical for an effective workflow. Sadly, Stata's support for working directories is awkward, inefficient, and inconsistent across operating systems. To change working directories, you can use `cd` *directory-name* or point and use *Change working directory...* in the File menu.[1] Differences across operating systems in how working directories are controlled adds complexity to what should be a simple task. Stata for Mac and Unix open in the working directory from your last session. Stata for Windows opens in the working directory set during installation or controlled by `cd` in `profile.do`. In Window, the `cd` command without a path reports the current working directory, while in other operating systems this changes your working directory to the base level of your current drive.

The `workflow` package is organized around the idea of a project. A project is a distinct activity such as a writing a paper, taking or teaching a specific class, writing a book, or developing a Stata command. Most simply, the `workflow` commands makes it easier to change working directories as your work moves from one project to another. For many, this will be the only feature of the commands of interest. For those doing more complex work, other features allow you to easily set up an environment for each project where globals are defined that specifying a path where datasets are located, associating an URL with your project, and automatically running commands when changing projects. The `workflow` package includes four commands:

`cdsave` creates ado files that changes the working directory and creates globals with information about each project. These commands are named `cd`*project* and are

1. Changing working directories in Stata is like having a powerful and elegant sports car that can only be started by opening the trunk to insert the key.

referred to as the `cd*` commands. To make these commands available regardless of your current working directory, they ado files are saved in your PERSONAL directory.

`cdlists` lists the the commands created by `cdsave` along with the name of the working directorythat is set by the or a note describing the project.

`cdinfo` lists the globals created by the last `workflow` command that was run.

`cddrop` drops commands that are no longer needed.

Non-programmers can use these commands without knowing anything about writing ado files. Programmers can easily customize the commands that are created to add other features.

The `workflow` package was inspired by the `fastcd` package by Winters (2002), Baum's ideas on creating globals for setting up the environment for a research project (Baum 2016, 78-79), and Long's ado files for changing the working directories (Long 2009, 111-112,117-118).

1 Using the project commands

This section begins by illustrating how the `workflow` package allows you to easily changing working directories. Next, I show how to modify your `profile.do` to control the working directory in which Stata opens and to then quickly changing to the directory for the project you want to work on. Next, I explain how to create robust and portable do-files that use globals to specify the location of datasets. And finally, I show how commands can be automatically executed when you change projects to further customize your project environment.

1.1 Quickly changing the working directory

A simple example using two projects explains the most basic use of `cd`*project* commands to quickly move among working directories. Suppose that project A uses working directory `d:/projecta/work` while project B uses the directory `d:/projectb/work`. To create the command `cdprojecta` to change the working directory, you begin by changing to the working directory you want for your project ant then run `cdsave`:

```
. cd d:/projecta/work
d:/projecta/work

. cdsave projecta, replace
command cdprojecta saved in your personal directory
```

I do the same thing for project B:

```
. cd d:/projectb/work
d:/projectb/work
```

```
. cdsave projectb, replace
command cdprojectb saved in your personal directory
```

To move between directories:

```
. cdprojecta
d:/projecta/work

. cdprojectb
d:/projectb/work
```

To list the `cd`*project* commands that I have defined,

```
. cdlist

      cdcda - d:/active/cda2017/work/
     cddesk - c:/users/jslong/desktop/
 cdprojecta - d:/projecta/work/
 cdprojectb - d:/projectb/work/
    cdstart - d:/statastart/
  cdwficpsr - d:/active/wficpsr2017/work/
 cdworkflow - d:/active/workflowwd/work/
```

I can change to the working directory for any project by running the command. Or, I can use the mouse to click on the name of the command I want to run. This feature is not available on some Stata installations. If it works on your system, the names should appear in blue.

As the number of `cd`*project* commands increase, scanning the directories become awkward. A more efficient approach is to use the note option when creating a command:

```
. cdsave projecta, note(Project A 2017) replace
command cdprojecta saved in your personal directory
```

Having done this for all of the commands except for `cdprojectb`,

```
. cdlist

      cdcda - CDA class
     cddesk - Desktop
 cdprojecta - Project A 2017
 cdprojectb - d:/projectb/work/
    cdstart - Default starting directory
  cdwficpsr - ICPSR WF class
 cdworkflow - Workflow cd commands
```

Even when notes are set for projects, I sometimes find it useful to list the working directories set by each command. This can be done either by running `cdlist, dir` which replaces notes with the working directory.

1.2 Controlling the starting working directory

You can change the working directory in which Stata starts, but there is not a simple way to do this. One solution is simply run `cdlist` as soon as Stata starts and click

on the project where you want to start. This section shows you how to change your `profile.do` to make this even simpler. It requires some effort the first time you do it, but thereafter it is automatic.

For Windows, the starting working directory is set when you install Stata. *Getting Started with Stata for Windows* explains how to change the properties sheet to change this. The explanation is confusing and has never worked for me. In Mac, the initial working directory is the working directory you were using when you last quit Stata. In Unix, the only method described in *Getting Started with Stata for Unix* is using the `cd` command and typing the directory you want each time Stata opens.

All three operating systems will run `profile.do` located the STATA directory; run `sysdir` to find the location. You can add a `cd` to `profile.do` or you can use the the commands in the `workflow` package. The first thing to do use `cdsave` to create the command `cdstart` to change to the working directory you want to start in. Then, add the command `cdstart` to `profile.do`. The `cdstart` command will run each time Stata starts and will set the working directory. If you change your mind on where you want Stata to start, you can create a new cdstart command with `cdsave ..., ... replace`.

If you work on many projects at the same time, you are likely to change directories immediately after you start Stata. To make this easier, you can add `cdlist` to your `profile.do`. For example,

```
cdstart
cdlist
```

If you want to prevent an error in case the `workflow` commands have not been installed on the computer you are using, you can use the more robust commands:

```
noisily di _new ". cdstart"
capture noisily cdstart
capture noisily cdlist
```

1.3 Robust specification of data paths

Stata's `use` command loads a dataset from the working directory unless a path has been specified. This path can point to a directory on your computer or an URL. While I prefer to keep the datasets related to a project in that project's working directory, in some cases this is not possible. For example, I might need to access data over the web or in a collaboration multiple people might need common access to datasets on a shared directory. Or, you might simple prefer to have your data stored someplace other than the working directory. In order for your do-files to run on other computers, you cannot hard code a directory location within your do-files. That is, you cannot do something like use `d:/data/binlfp4, clear` since your do-file will not run on a computer that does not have exactly the same directory structure, including drive names. Recently, the director of our statistical consulting center showed me a message a client received from an editor asking him/her to resubmit the script files for an article after removing the hard coding of directories. Globals can be used so that your do-files can load data from

locations other than your working directory, but that also allow do-files to be portable across computers. To do this, I create a global with the path where data is saved. For example,

```
global S_cddata "d:/datasets/"
```

This global is *not* created in my do-file for reasons explained below. My do-file uses data with the command:

```
use "${S_cddata}binlfp4", clear
```

which is interpreted as:

```
use "d:/datasets/binlfp4", clear
```

If the global was not defined, the command would be interpreted as:

```
use "binlfp4", clear
```

Or, if I am working on a computer with a different file structure, my do-file will still run if I first create the global. For example,

```
global S_cddata "c:/users/jslong/documents/datasets/"
```

My globals defining the path always end with `/`. If I did not do this, I would load my dataset with:

```
use "${S_cddata}/binlfp4", clear
```

Indeed, I could even remove the parentheses:

```
use "$S_cddata/binlfp4", clear
```

While this looks simpler, an error occurs if the global has not been defined since the command is interpreted as

```
use "/binlfp4", clear
```

which causes an error.

When creating `cd` *project* commands there are three ways to add information about data locations to your project environment. The `dataset(`*path*`)` options specifies a path where datasets are saved. For example,

```
cdsave census, data(k:/txcdc/census2000/datasets)
```

This path is saved in the global `S_cddata`. A web address can added with the `url(`*address*`)` option. For example,

```
cdsave spost, url(http://www.indiana.edu/~jslsoc/stata/spex_data/)
```

Since you might want more than two data locations, the `user(`*string*`)` can be used to creates the global `S_cduser` that could hold another path.

You can add both a data path, an URL, and a user path at the same time. For example,

```
cdsave paths, note(Example of data paths) data(d:/data) ///
    url(http://www.indiana.edu/~jslsoc/stata/spex_data) ///
    user(m:/bssr/datadepository/) replace
```

The cdsave command will automatically add an ending `/` to the path or URL, but you must add it yourself with the `user()` option since this option could be used to hold other information where you do not want an ending slash.

After the data locations are added, a single do-file could load data from multiple locations:

```
use "${S_cddata}binlfp4", clear
  (output omitted)
use ${S_cdurl}nomocc4, clear
  (output omitted)
use "${S_cduser}ordwarm4", clear
  (output omitted)
use "couart4", clear
  (output omitted)
```

The quotes are needed in case your path includes a space.

You might also want different data directories for source data and derived datasets:

```
cdsave paths, note(Example of data paths) ///
    data(d:/data/source) ///
    user(d:/data/derived) ///
    url(http://www.indiana.edu/~jslsoc/stata/spex_data) replace
```

then:

```
use "${S_cddata}binlfp4", clear
  (output omitted)
save "${S_cduser}binlfp5", replace
  (output omitted)
```

The `cdinfo` command list the globals that are currently active. For example,

```
. cdinfo
  S_cdcmd:    cdpaths
  S_cdnote:   Example of data paths
  S_cdwd:     d:/active/project/work/
  S_cdauto:
  S_cddata:   d:/data/source
  S_cdurl:    http://www.indiana.edu/~jslsoc/stata/spex_data/
  S_cduser:   d:/data/derived/
```

Notice that all data locations are saved using `/` which works in Win, Mac, and Unix. If you enter a path using `\` it is converted to `/`. The list of globals can also be obtained by adding the `details` option to either `cdsave` or `cdlist`.

1.4 Further customization of your research environment

The `autorun(`*command*`)` option in `cdsave` lets you add a command that is run at the end of the `cd`*project* command. Since it is the last command run, it can use any of the globals created by the command. The command can be an internal Stata command. For example,

```
cd d:/datasets
cdsave autodir, note(Automatically list datasets) ///
    data(d:/datasets) auto(dir $S_data*.dta) replace
```

When I run `cdautodir`, the available datasets are listed:

```
. cdautodir
d:/datasets
  51.1k   9/08/16  8:48  binlfp4.dta
  30.6k   4/24/14  6:02  couart4.dta
  18.6k  11/13/13  9:21  couexposure4.dta
  69.4k  11/20/13 15:31  gssclass4.dta
  75.9k   3/02/14 11:19  gsskidvalue4.dta
  39.0k   3/12/14 13:26  partyid4.dta
  32.0k   3/02/14 11:03  science4.dta
 935.4k   3/04/14 13:16  svyhrs4.dta
  12.3k   4/01/14  7:14  travel4.dta
  10.7k   4/01/14  7:14  travel4case.dta
 153.6k   8/11/14 15:08  wlsrank4.dta
  70.3k   3/02/14 10:59  wlsrnk4.dta
```

Or, you could automatically load a dataset:

```
cdsave autouse, data(d:/datasets) auto(use $S_databinlfp4, clear)
```

Then,

```
. cdautouse
d:/datasets
(binlfp4.dta | Mroz data on labor force participation of women | 2014-10-20)
```

Or you could automatically run a do-file associated with your project that could be arbitrarily complex:

```
cdsave projectc, autorun(do projectc-setup.do)
```

2 Programming features used by the workflow package

The `workflow` package depends on several features of Stata to allow the commands to run regardless of your current working directory and for the globals defining your project environment to persist after dropping macros.

Dynamically creating ado files

Non-programmers are not likely to learn how to write ado files in order to simplify changing working directories and programmers are unlikely to want to take the time to write these commands. Accordingly, `cdsave` use `file write` to create the ado files with commands for changing projects. Unless you want to customize what these commands do, there is no reason to change them.

The PERSONAL directory

`cdsave` saves the ado files in the PERSONAL directory so that the commands will run regardless of your current working directory. If PERSONAL is not defined or the directory specified in PERSONAL does not exist, `cdsave` exists with an error. To learn more, run `sysdir` to determine where your PERSONAL is located. If no directory is listed or it points to a directory that does not exist, you can set the directory with `sysdir set`; enter `help sysdir` for details.

Use of system macros

The globals that are created by `cd`*project* commands to define your project environment need to persist as long as you are working on that project. To prevent these globals from being remove if you run `macro drop all`, the global names begin with `S_cd`. Macros that being with `S_` can only be deleted by using `macro drop S_`*name* or `macro drop S_*`. Critical system macros will not be dropped. The following globals are created by each `cd*` command:

`S_cdauto`	User specified command executed by `cd`*project*.
`S_cdcmd`	Name of the current `cd`*project* command.
`S_cddata`	Path for using datasets.
`S_cdnote`	Note associated with `cd`*project*.
`S_cdurl`	Web address for using datasets.
`S_cduser`	Any string.
`S_cdwd`	Working directory set by `cd`*project*.

3 cdsave: creating cd commands

The `cdsave` command creates a `cd`*project*`.ado` file that is saved in the PERSONAL directory. When no options are specified, the only thing that the `cd`*project* command does is change the working directory to the working directory that was active at the time `cdsave` was run. The syntax is:

`cdsave cd`*project* [`, note(`*string*`) datapath(`*data-path*`) url(`*url*`)`
`autorun(`*command*`) details user(`*string*`) replace`]

where *project* is the mnemonic for your project. Commands created by `cdsave` must begin with the letters `cd`, but you can enter the name without the `cd` prefix. For example, both `cdsave myproject` and `cdsave cdmyproject` can be used.

Options

`note(`*string*`)` creates a note to describe the current project. If no note is specified, the working directory being set is used as the note. Notes are shown by `cdlist` to help you remember what each `cd` command does.

`datapath(`*data-path*`)` is a path where datasets are located. This path is stored in the global `S_cddata`. You can load a dataset from this path with `use ${S_cddata}`*filename*

`url(`*web-address*`)` is a web address for datasets. The URL is stored in the global `S_cdurl`. You can load datasets from this location with `use ${S_cdurl}`*filename*.

`user(`*string*`)` can by anything. For example, you could use it as a second data directory or a second URL. The string is stored in the global `S_cduser`. If it contained a path, you could load datasets using `use ${S_cduser}`*filename*

`autorun(`*command*`)` is a command that will be run by `cd`*project*. This can be an official Stata command, the name of an ado file you have written, or a do-file that is to be run, such as `auto(do projectstartup.do)`.

`details` lists the globals that will be created when `cd`*project* is run.

`replace` overwrite `cd`*project*`.ado` if it exists.

Globals created by `cd`*project* commands are not removed by `clear all` or `macro drop _all`. To remove them you must use `macro drop` *name*, such as `macro drop S_cddata` or `macro drop S_cd*`.

4 The cdlist command

`cdlist` lists the names of ado-files in PERSONAL begin with `cd` along with the note for each project. If you click on the name of a command, which is shown in blue, the command is executed. The syntax is:

`cdlist` [`, dir`*directories* `d`*details*]

If the working directory for a command is not found, a warning is given.

Options

`details` displays the `S_cd` globals created by each command.

`directories` displays the directory being set even if a note is defined.

Example The names of each command are listed in blue in the Results window. You can click on them to run the command. If the `note()` option was not used in creating command, such as `cdbbb` below, the working directory for the project is listed.

```
. cdlist

      cdaaa - project A
      cdbbb - d:/bbb/work/
     cddesk - Desktop
  cdproject - My project
    cdspost - SPost development
    cdstart - Default working directory
 cdworkflow - Commands for managing projects
```

5 The cdinfo commands

This command simply lists the `S_cd*` macros associated with last `cd`*project* command.

`cdinfo`

There are no options.

Example

```
. cdproject
d:\project\work

. cdinfo
  S_cdcmd:    cdproject
  S_cdnote:   My project
  S_cdwd:     d:/project/work/
  S_cdauto:
  S_cddata:   d:/project/work/
  S_cdurl:    http://www.indiana.edu/~jslsoc/stata/spex_data/
  S_cduser:
```

6 The cddrop command

`cddrop` drops, deletes, or restores `cd`*project*`.ado` files in PERSONAL.

`cddrop cd`*project* | `_all` [`, delete restore`]

`cd`*project* is the name of the command whose ado file in PERSONAL is modified. By

default, `cddrop cd`*project* renames `cd`*project*`.ado` to `cd`*project*`.dropped`. After the command is renamed, it will no longer run. With the `delete` option the ado file is deleted. A file that has been renamed can be restored with the `restore` option. For example, `cddrop cdexample, restore` renames `cdexample.dropped` to `cdexample.ado`. The abbreviation `_all` will drop or restore all `cd` commands.

Options

`delete` erase the ado file for the command.

`restore` renames `cd`*project*`.dropped` to `cd`*project*`.ado`.

7 Conclusions

I hope that readers find these commands useful for developing an efficient workflow that supports producing reproducible results. As the power of Stata increases, I hope they will add features that make my commands obsolete!

About the authors

Scott Long is ...

8 References

Baum, C. F. 2016. *An introduction to Stata programming*, vol. 2. 2nd ed. Stata Press College Station.

Long, J. S. 2009. *The Workflow of Data Analysis Using Stata*. College Station, TX: Stata Press.

Winters, N. 2002. fastcd: module to automate changing directories. http://www.nd.edu/~rwilliam/stata/gologit2.pdf..