7+ tools to make your Stata life more pleasant

Jesse Wursten
London SUGM, August 2019

Faculty of Economics and Business
KU Leuven
User-Computer Interaction

Four commands that make the computer talk to you

- sendtoslack
- cdo
- stop
- beep
Send a message from Stata to your smartphone

- Stata tells Powershell what to send and who to send it to
- Powershell sends that info to Slack
- Slack sends you your message
- You receive the message on your desktop or smartphone

Some notes

- Easily integrated in other programs (as you will see)
- Options can be saved in profile.do
- Available on SSC
. sendtoslack, m(You should check your computer) url(default)
Sending message to default url: https://hooks.slack.com/services/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
Message sent: You should check your computer

Today

statamessage  APP  2:44 PM
You should check your computer

Jot something down
cdo

-do- things with some extra options

- Send message to phone on success
- Send message to phone on failure
- Execute local copy
- Run program on dofile end
- Make log of this run

Some notes

- Example of sendtoslack being integrated
- cdo can add timestamps to the logfile names
- Available on SSC
cdo

Command including optional saved options:  cdo niceDofile.do, arguments() url(default) program() log()
This corresponds to following do-command:  do niceDofile.do
Sending message to default url:  https://hooks.slack.com/services/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
Message sent:  Starting dofile niceDofile.do

. sleep 1000

. 

end of do-file

Execution: success
Sending message to default url:  https://hooks.slack.com/services/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
Message sent:  Successfully completed dofile niceDofile.do . Execution took 0 hours, 0 minutes and 1 seconds.
Starting dofile niceDofile.do

Successfully completed dofile niceDofile.do. Execution took 0 hours, 0 minutes and 1 seconds.
cdo

Command including optional saved options: cdo faildoFile.do, arguments() url(default) program() log()
This corresponds to following do-command: do faildoFile.do
Sending message to default url: https://hooks.slack.com/services/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
Message sent: Starting dofile faildoFile.do

.error 600
r(600);
Starting dofile faildoFile.do

Error 600 has occurred in dofile faildoFile.do
stop

stops a dofile

- Can send a message to smartphone
- Closes log files (if you want to)
- That’s more or less it

Some notes

- It feels wrong to write errors into my code
- Available on SSC
. stop
    Process on Jesse PC has finished.

    Sending message to default url: https://hooks.slack.com/services/T7895UN5N/B7SK8J26P/bhP4HcGoz651kdmjks5i2Q0
    Message sent: Process on Jesse PC has finished.
    Effective command executed: stop, sts(default) message(Process on Jesse PC has finished.)

--Break--
r(1);
beep

Produces a beep

- Start your dofile
- Then add the beep command in the command window

Some notes

- Requires sound
- The beep is moderately annoying
- Built-in
beep

```
. sleep 10000
```

Command

beep
Lazy Programming

Five commands that help the lazy programmer

- batcher
- timeit
- smallfileversioning
- censusapi
- makecode
Task parallelisation in Stata

- Give batcher a set of arguments (numlist)
- Batcher opens a new Stata instance
- ... runs dofile with first argument
- ... opens new Stata, run dofile with second argument, etc
- ... reports back when instances finish/crash

Some notes

- Integrated with sendtoslack
- Requires minor adaptation to dofile
- Available on SSC
global thread = `1'       // Not required, but conceptually clearer

* Some examples
** Simple display of iteration number
if "$thread" == "1" di "I was asked to display 1"
if "$thread" == "2" di "I was asked to display two"

.batcher $examplePath, i(1/2) tempfolder(C:/StataWD)
Starting c:\ado\personal\b\exampleDofile.do
  iteration 1
  iteration 2
. batcher $examplePath, i(1/2) tempfolder(C:/StataWD)
Starting c:\ado\personal/b/exampleDofile.do
  iteration 1
  iteration 2
Starting tracking in 60 seconds. Refreshing every 30 seconds.
  Finished:  1  2  OK

Batch job has finished.
timeit

Single-line timer

• Prefix timeit #:
• Use timer list to obtain results

Some notes

• You can also give the timers names
• Also saves incremental timers
• Available on SSC
. timeit 1 crucial: sleep 1000

. timeit 1 lesscrucial: sleep 2000

. timer list
  1: 3.00 / 2 = 1.4980

. return list

scalars:
  r(t1) = 2.996
  r(nt1) = 2
  r(lesscrucial) = 2.996
  r(delta_t1) = 1.991
  r(critical) = 1.005
smallfileversioning (sfv)

Saves old versions of dofiles automatically

- You tell sfv which folders to watch and how often
- sfv checks if any of the files in there have changed
- ... if so, it copies those files to a different folder
- ... and names them according to the change date/time

Some notes

- Very easy to set up
- Less effort than github or manual duplication
- Not limited to dofiles
Accessing the US Census API

- Hides most of the ‘API’-part behind the scenes
- Splits the URL into options to make code more readable
- Allows for P0100001-P0100022 varlist style
- Splits Census calls of too many variables

Some notes

- Only tested on Decennial Census and QWI
- Available on SSC
censusapi

```r
. censusapi, dataset("https://api.census.gov/data/1990/sf1") variables("AREALAND P0110001-P0110031 P0060001-P0060005") predicate("for=place:*&in=state:*") dest > ination("test.txt")

Importing variables through Census API.
Using stored key (eb082e585b03c87d8c27d6c4955b8b7a01ed2607)
Parsing variable list
AREALAND
P0110001,P0110002,P0110003,P0110004,P0110005,P0110006,P0110007,P0110008,P0110009,P0110010,P0110011,P0110012,P0110013,P0110014,P0110015,P0110016,P0110017,P0
>110018,P0110019,P0110020,P0110021,P0110022,P0110023,P0110024,P0110025,P0110026,P0110027,P0110028,P0110029,P0110030,P0110031
P0060001,P0060002,P0060003,P0060004,P0060005
Opening file
```

```
C:\\WINDOWS\\SYSTEM32\\cmd.exe

% Total  % Received % Xfered Average Speed   Time    Time     Time  Current
100 98547  0 98547  0  0 19709 0  --:--:-- 0:00:05 --:--:-- 19772
```
<table>
<thead>
<tr>
<th>arealand</th>
<th>p01100001</th>
<th>p01100002</th>
<th>p01100003</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40308</td>
<td>28</td>
<td>97</td>
</tr>
<tr>
<td>2</td>
<td>7948</td>
<td>43</td>
<td>89</td>
</tr>
<tr>
<td>3</td>
<td>7688</td>
<td>7</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>1437</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>48812</td>
<td>235</td>
<td>583</td>
</tr>
<tr>
<td>6</td>
<td>66073</td>
<td>158</td>
<td>418</td>
</tr>
<tr>
<td>7</td>
<td>100616</td>
<td>177</td>
<td>403</td>
</tr>
<tr>
<td>8</td>
<td>11646</td>
<td>35</td>
<td>93</td>
</tr>
</tbody>
</table>
Create structured code blocks

- Can only explain this in picture

Some notes

- Meant to be customised
- Considering move to Powershell
- Not publicly available (email me)
makecode

```bash
makecode reg_firmLevel 2
if "$reg_firmLevel" == "1" {
    * Start log
    cap log close reg_firmLevel
    log using "$path_logs/2_reg_firmLevel.log", replace text name(reg_firmLevel)

    * Close log
    log close reg_firmLevel
}
```

```
global reg_firmLevel "1"
if "$reg_firmLevel" == "1" {
    * Start log
    cap log close reg_firmLevel
    log using "$path_logs/2_reg_firmLevel.log", replace text name(reg_firmLevel)

    * Close log
    log close reg_firmLevel
}
```
makecode

global parsePatentData "0"
  global ppd_gaetan "0"
  global ppd_fixShapefiles "0"
  global ppd_eipo "0"
  global ppd_splitEpo "0"
  global ppd_geolocateEpo "0"
  global ppd_addSpatialIdentifiersToEpo "1"
  global ppd_identifyAddress "0"
  global ppd_citedPatents "0"
  global ppd_citedPatentsAddresses "0"
  global parseCensusUSData "0"
    global pcd_version "20190130_unique"
  global parseCensusUK "0"
    global pcuk_oaCrosswalk "0"
    global pcuk_wzCrosswalk "0"
    global pcuk_mergeInWz "0"
    global pcuk_wzStats "0"
    global pcuk_interpolate "1"
  global combineData "1"
  global cd_patentDataSource "0"
    global cd_pds_focAddressFile "0"
      // FOCAL: Get file with NUTS codes for all focal addresses
    global cd_pds_focSameLocFile "0"
      // FOCAL: Identify same-NUTS teams
    global cd_pds_focRandomLocFile "0"
      // FOCAL: Randomly select a location (three)
    global cd_pds_focAllInventors "0"
      // FOCAL: All teams with location information
    global cd_pds_citRandomSameLocFile "0"
      // CITED: Random inventor & Same Location info
    global cd_pds_citAllInventors "1"
      // CITED: All teams with location information
    global cd_pds_focValidAppl "1"
      // FOCAL: Validated applicant address
    global cd_pds_citFocLink "0"
      // XWALK: Focal-cited
    global cd_pds_citInfo "0"
      // CITED: Application info
    global cd_pds_focInfo "1"
      // FOCAL: Application info
  global cd_matchvars "0"
    global cd_mv_sameNuts_sameLoc "0"
      // Same NUTS focal, same LOC cited
    global cd_mv_sameNuts_randomCited "0"
      // Same NUTS focal, random LOC cited
    global cd_mv_randomFocal_randomCited "0"
      // Random focal, random cited
```java
* Code

if "$parsePatentData" == "1" {

if "$parseCensusUSData" == "1" {

if "$parseCensusUK" == "1" {

if "$combineData" == "1" {

if "$assignCountries" == "1" {

if "$regressionPrep" == "1" {

if "$regressionPrep_old" == "1" {
```
Bonus tricks

Old tools sometimes work best

• Windows copy path
• Label data/variable/values
Any Questions or Suggestions?

I highly recommend the Metropolis Theme for Beamer (modified version used in this presentation), it can be found at https://github.com/matze/mtheme