7+ tools to make your Stata life more pleasant

Jesse Wursten[1]

London SUGM, August 2019

[1]Faculty of Economics and Business
KU Leuven

User-Computer Interaction

Four commands that make the computer talk to you

- sendtoslack
- cdo
- stop
- beep

sendtoslack

Send a message from Stata to your smartphone

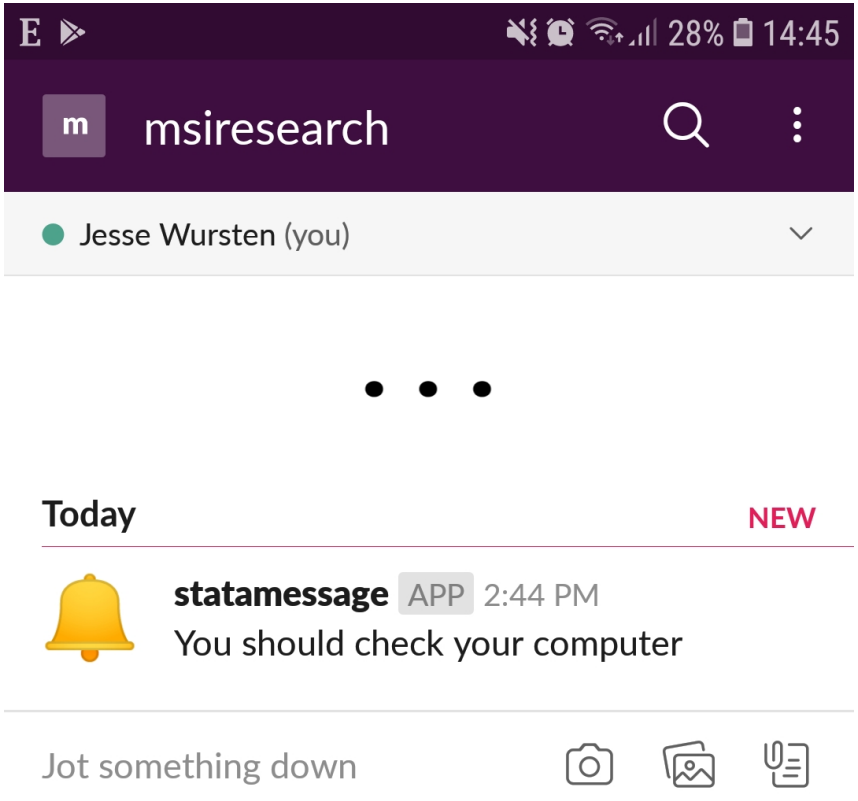
- Stata tells Powershell what to send and who to send it to
- Powershell sends that info to Slack
- Slack sends you your message
- You receive the message on your desktop or smartphone

Some notes

- Easily integrated in other programs (as you will see)
- Options can be saved in profile.do
- Available on SSC

sendtoslack

```
. 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
```



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

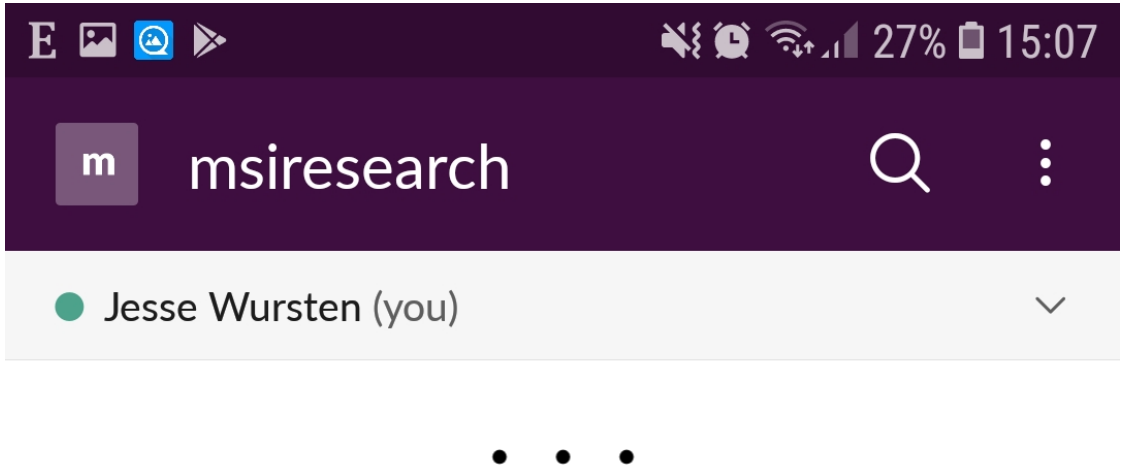
```
cdo
   Command including optional saved options:     cdo niceDofile.do, arguments() url(default) program() log()
   This corresponds to following do-command:     do niceDofile.do
   Sending message to default url: https://hooks.slack.com/services/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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

cdo

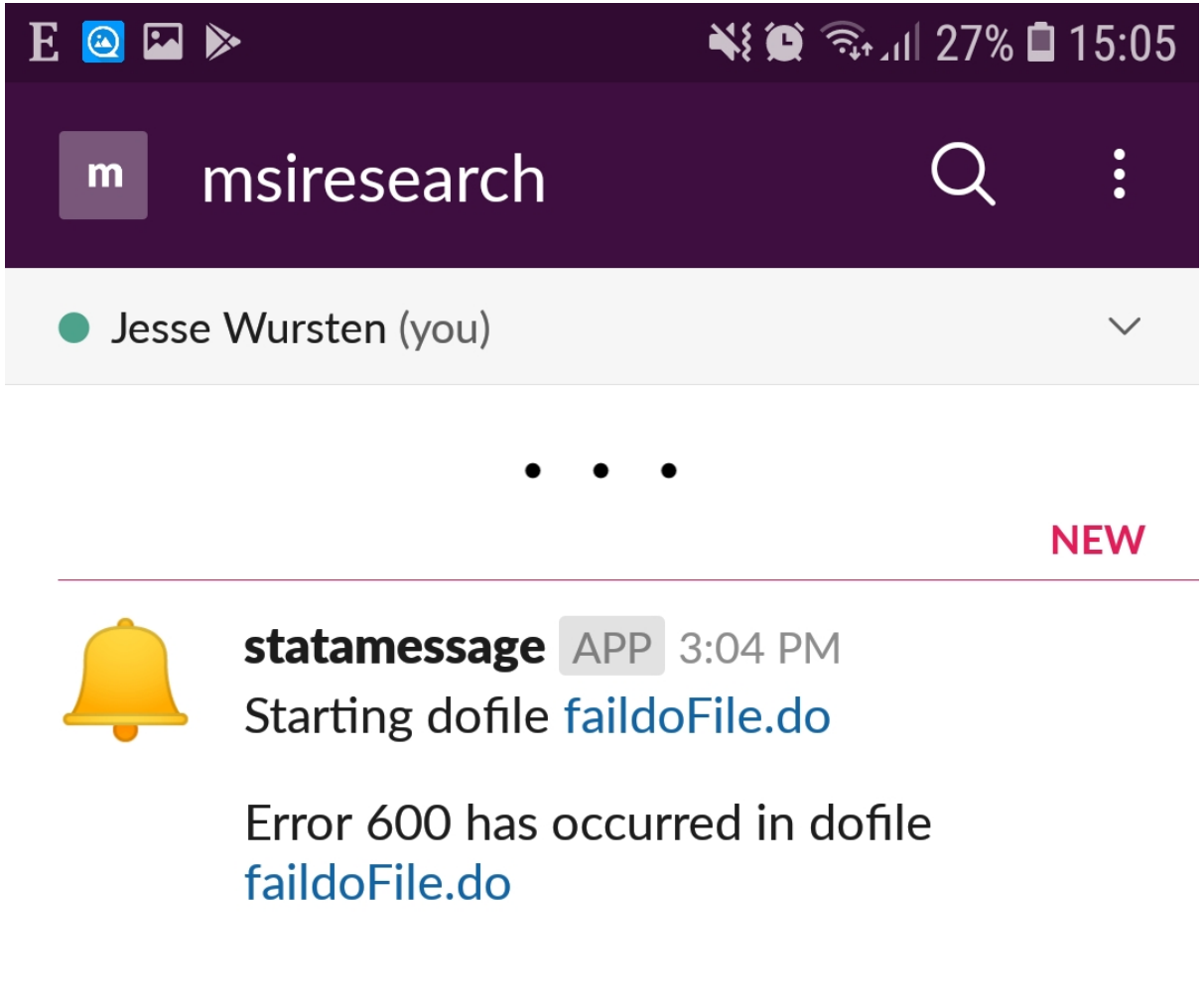

27% 15:07
m
msiresearch
Jesse Wursten (you)
NEW
Starting dofile niceDofile.do
Successfully completed dofile niceDofile.do . Execution took 0 hours, 0 minutes and 1 seconds.

cdo

```
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);
```

cdo


27% 15:05
m
msiresearch
Jesse Wursten (you)
NEW
statamessage APP 3:04 PM
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

```
. stop
   Process on Jesse PC has finished.

   Sending message to default url: https://hooks.slack.com/services/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
   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

batcher

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

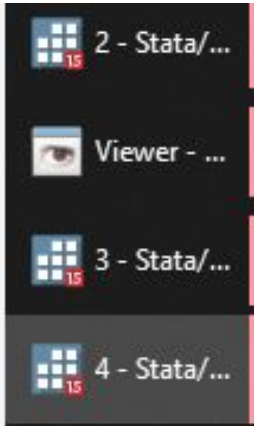
batcher

```
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




```
. 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

```
. 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(crucial) =  1.005
```

smallfileversioning (sfv)

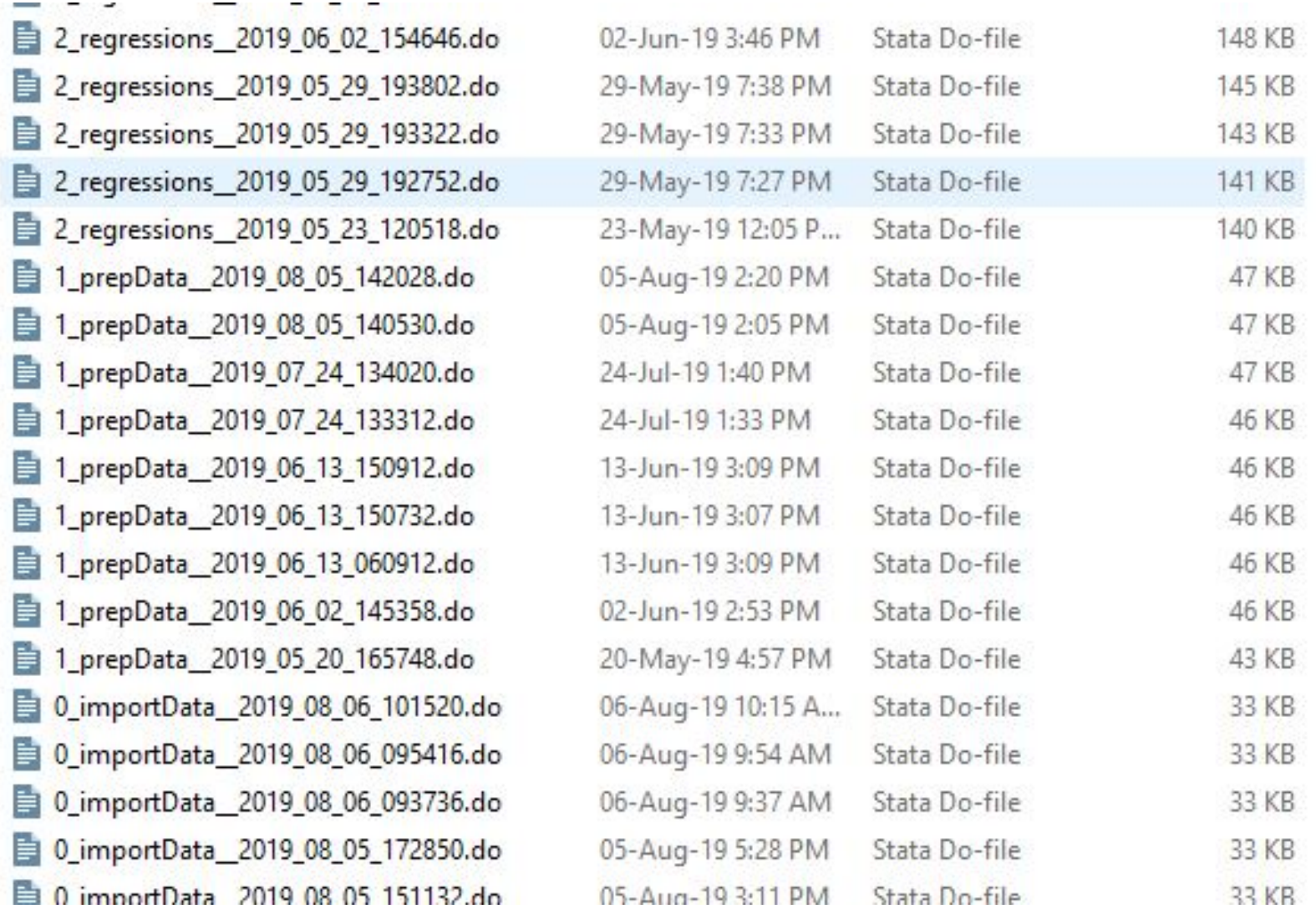
Saves old versions of dofiles automatically

- 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

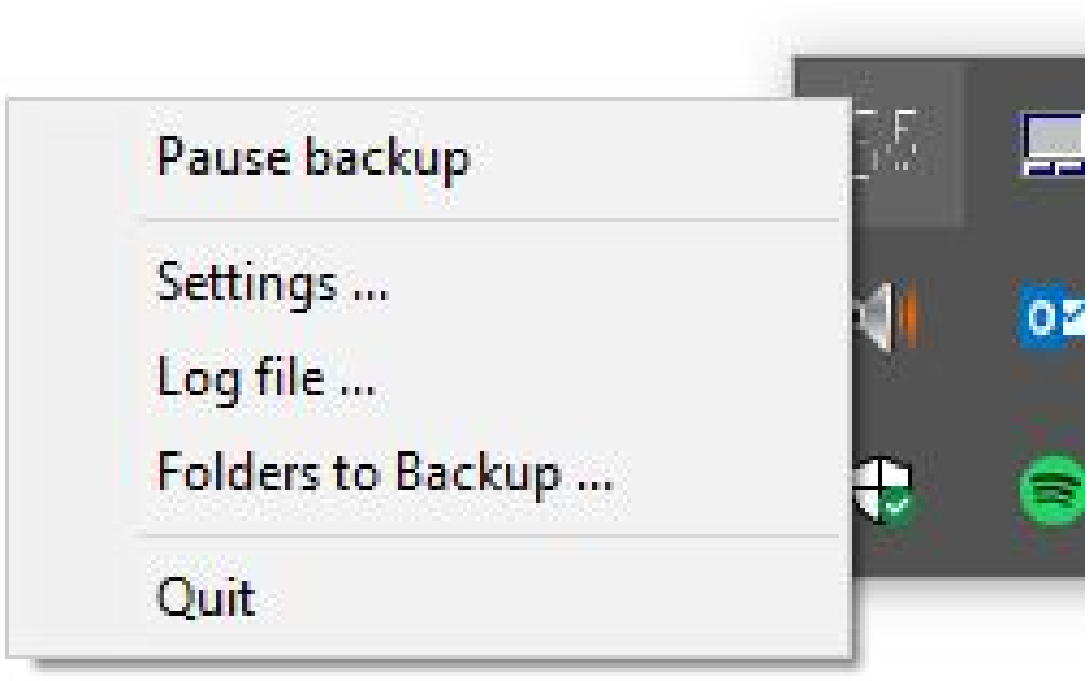
- Very easy to set up
- Less effort than github or manual duplication
- Not limited to dofiles
- Available on Github: `http://bit.ly/sfv_sugm`

sfv



2_regressions__2019_06_02_154646.do	02-Jun-19 3:46 PM	Stata Do-file	148 KB
2_regressions__2019_05_29_193802.do	29-May-19 7:38 PM	Stata Do-file	145 KB
2_regressions__2019_05_29_193322.do	29-May-19 7:33 PM	Stata Do-file	143 KB
2_regressions__2019_05_29_192752.do	29-May-19 7:27 PM	Stata Do-file	141 KB
2_regressions__2019_05_23_120518.do	23-May-19 12:05 P...	Stata Do-file	140 KB
1_prepData__2019_08_05_142028.do	05-Aug-19 2:20 PM	Stata Do-file	47 KB
1_prepData__2019_08_05_140530.do	05-Aug-19 2:05 PM	Stata Do-file	47 KB
1_prepData__2019_07_24_134020.do	24-Jul-19 1:40 PM	Stata Do-file	47 KB
1_prepData__2019_07_24_133312.do	24-Jul-19 1:33 PM	Stata Do-file	46 KB
1_prepData__2019_06_13_150912.do	13-Jun-19 3:09 PM	Stata Do-file	46 KB
1_prepData__2019_06_13_150732.do	13-Jun-19 3:07 PM	Stata Do-file	46 KB
1_prepData__2019_06_13_060912.do	13-Jun-19 3:09 PM	Stata Do-file	46 KB
1_prepData__2019_06_02_145358.do	02-Jun-19 2:53 PM	Stata Do-file	46 KB
1_prepData__2019_05_20_165748.do	20-May-19 4:57 PM	Stata Do-file	43 KB
0_importData__2019_08_06_101520.do	06-Aug-19 10:15 A...	Stata Do-file	33 KB
0_importData__2019_08_06_095416.do	06-Aug-19 9:54 AM	Stata Do-file	33 KB
0_importData__2019_08_06_093736.do	06-Aug-19 9:37 AM	Stata Do-file	33 KB
0_importData__2019_08_05_172850.do	05-Aug-19 5:28 PM	Stata Do-file	33 KB
0_importData__2019_08_05_151132.do	05-Aug-19 3:11 PM	Stata Do-file	33 KB

sfv



censusapi

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

```
.  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 (eb082e589b03c87d8c27d6c4955b8b7a01ed2607)
  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 % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 98547    0 98547    0     0  19709      0 --:--:--  0:00:05 --:--:-- 19772
```

censusapi

	arealand	p0110001	p0110002	p0110003
1	40308	28	97	
2	7948	43	89	1
3	7688	7	17	
4	1437	6	10	
5	48812	235	583	5
6	66073	158	418	4
7	100616	177	403	4
8	11646	35	93	

makecode

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

```
. 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_epo "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
        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 informati
        global cd_pds_citRandomSameLocFile "0"        // CITED: Random inventor & Same Location i
        global cd_pds_citAllInventors "1"             // CITED: All teams with location informati
        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
```

makecode

```
* 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