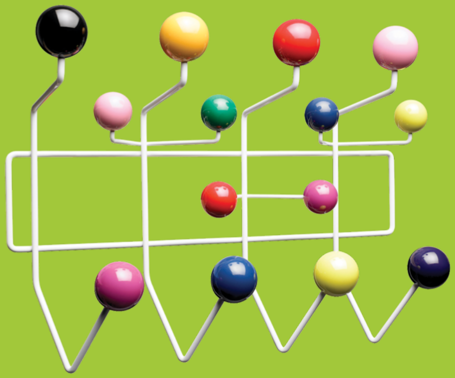
KU LEUVEN HIVA

ONDERZOEKSINSTITUUT VOOR ARBEID EN SAMENLEVING

Reindex & Indielabel
Two tools for data management

Sem Vandekerckhove
Stata User Group Meeting
18 September 2018
Brussels

Sem.Vandekerckhove@kuleuven.be



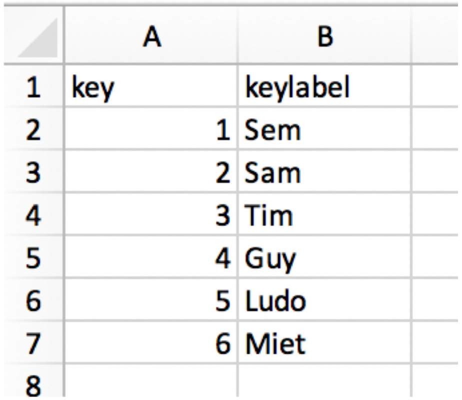
Design Charles & Ray Eames - Hang it all © Vitra

- Who hates Excel?
- Who uses Excel?

- Stata labels aren’t very handy
  - Label define / list / save
  - Collapse: lost
  - Mlab() option: decode first
  - Shorten labels: issue of –valuesof- vs –labelsof-
  - Translate or adjust labels

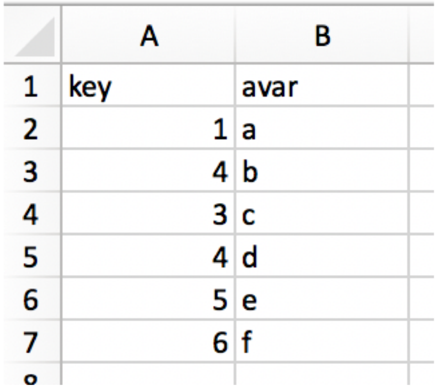
Indielabel = indepent labels

- Excel file (or .dta file) with two columns
  - Varname
  - Varnamelabel



	A	B
1	key	keylabel
2	1	Sem
3	2	Sam
4	3	Tim
5	4	Guy
6	5	Ludo
7	6	Miet
8		

- Data file with varname



	A	B
1	key	avar
2	1	a
3	4	b
4	3	c
5	4	d
6	5	e
7	6	f

- User command (soon on ssc):

```
indielabel key using usingfile.xlsx
```

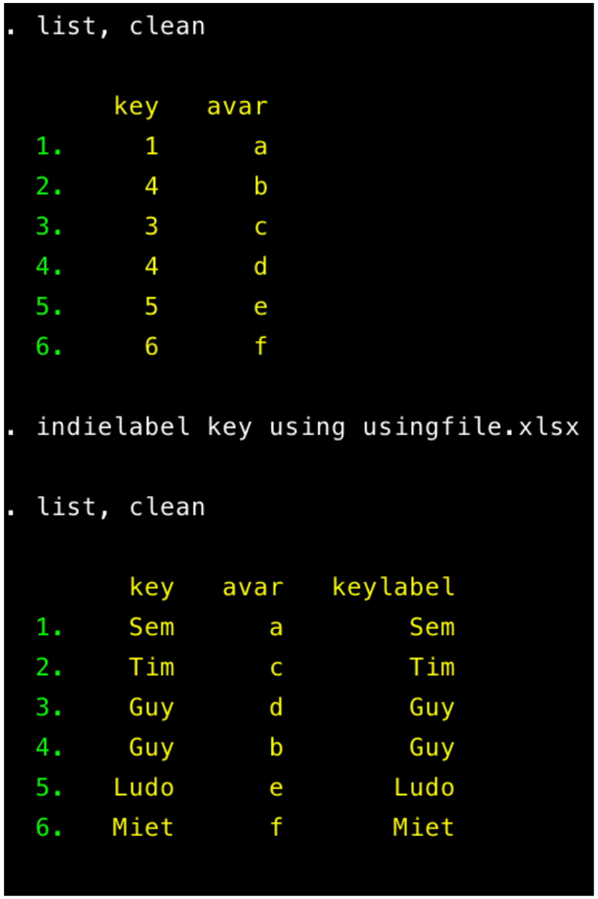
How does it work?

- It checks usingfile
  - Default `dta`
  - If `xlsx`: `import excel`
  - If anything else: error

- It checks if `varnamelabel` exists in `usingfile`
  - To do: allow given label names

- It makes a **labelfile** to
  - Merge with the `usingfile` (default `nomerge` – large files)
  - Build en run a do-file (`label save`) (default `label`)

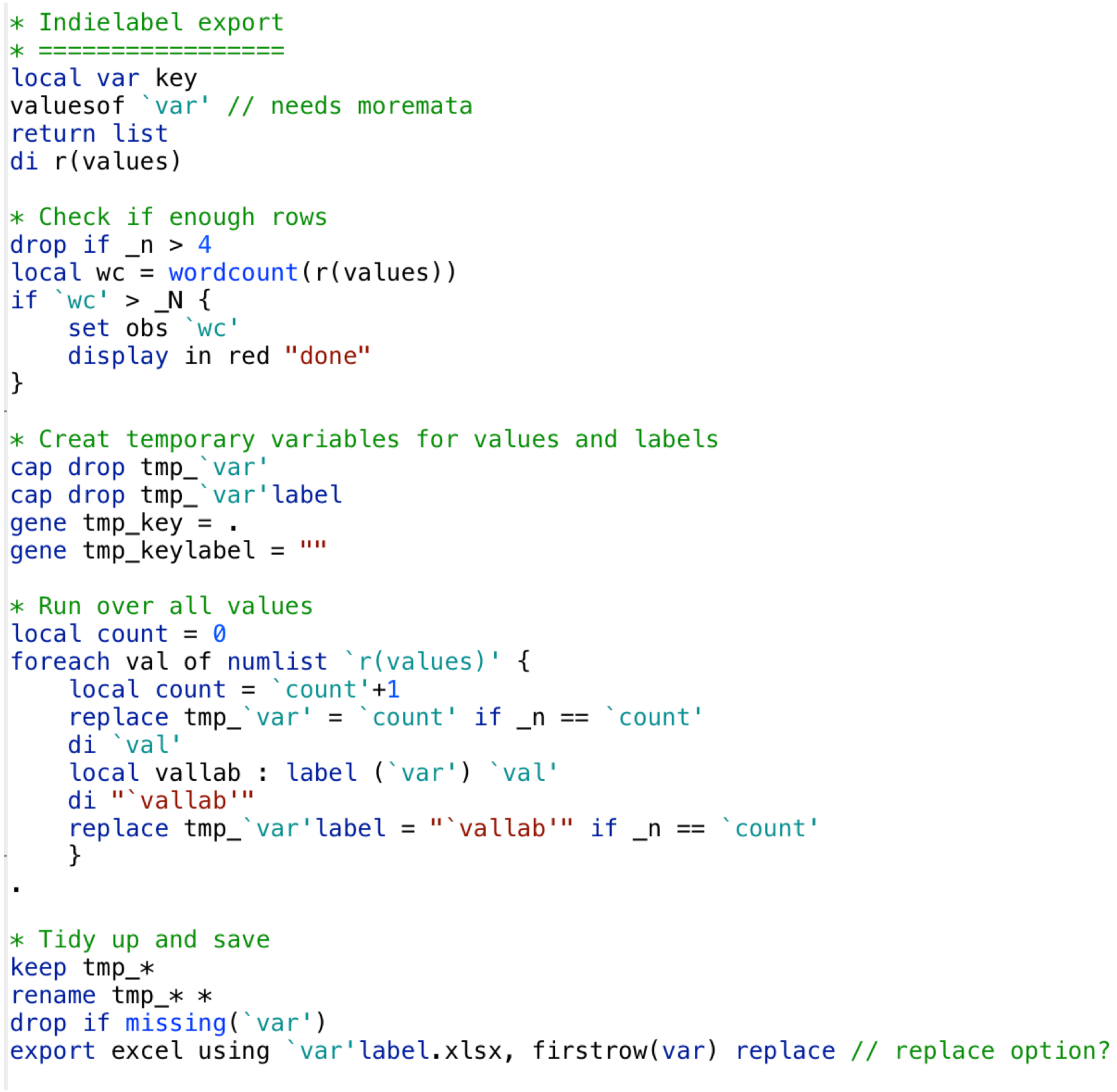
Indielabel: result



```
. list, clean

       key   avar
  1.     1      a
  2.     4      b
  3.     3      c
  4.     4      d
  5.     5      e
  6.     6      f

. indielabel key using usingfile.xlsx

. list, clean

        key   avar   keylabel
  1.    Sem      a        Sem
  2.    Tim      c        Tim
  3.    Guy      d        Guy
  4.    Guy      b        Guy
  5.   Ludo      e       Ludo
  6.   Miet      f       Miet
```

```
import excel using masterfile.xlsx,
firstrow clear
list, clean
indielabel key using usingfile.xlsx
list, clean
```

To do: indielabel key, export



```
* Indielabel export
* ================
local var key
valuesof `var' // needs moremata
return list
di r(values)

* Check if enough rows
drop if _n > 4
local wc = wordcount(r(values))
if `wc' > _N {
    set obs `wc'
    display in red "done"
}

* Creat temporary variables for values and labels
cap drop tmp_`var'
cap drop tmp_`var'label
gene tmp_key = .
gene tmp_keylabel = ""

* Run over all values
local count = 0
foreach val of numlist `r(values)' {
    local count = `count'+1
    replace tmp_`var' = `count' if _n == `count'
    di `val'
    local vallab : label (`var') `val'
    di "`vallab'"
    replace tmp_`var'label = "`vallab'" if _n == `count'
    }
.

* Tidy up and save
keep tmp_*
rename tmp_* *
drop if missing(`var')
export excel using `var'label.xlsx, firstrow(var) replace // replace option?
```

Reindex

- GDP growth, multiple countries
  - Levels: country size differs
  - Index: base year all equal
  - Percent: loses cumulation

- We would like to switch, fast and easy

Reindex

Syntax

```
reindex varname, generate(name) [i(name)] j(name) from(string) to(string)
[fromscale(integer 1)] [toscale(integer 1)] [tobase(integer -999)]
```

varname is the name of the variable you want to manipulate

Options

generate()	mandatory option, give a name for the new variable
i()	set the panel id variable if there are panels. You can use just one variable and it needs to be numeric. Pretty much like -xtset-, if there is no panel id, you can leave it out.
j()	mandatory: time variable. Intervals must always be the same
from()	mandatory: choose between levels, percent, factor, and index
to()	mandatory: choose between percent, factor, and index
fromscale()	in case your variable was multiplied by some factor (say 100 for percentages)
toscale()	same for the target variable
tobase()	set the base time period (mandatory for to(index))

Reindex: how it works

- It checks if time increments are always the same

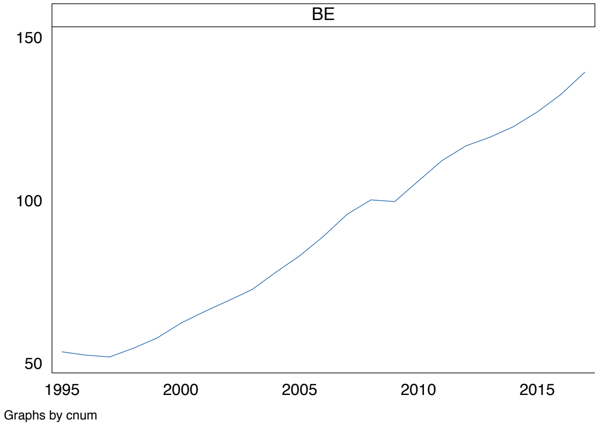
- Percent to factor: `replace `t' = `t'+1`
- Levels to factor: `replace `ta' = `t'/`t'[_n-1]`

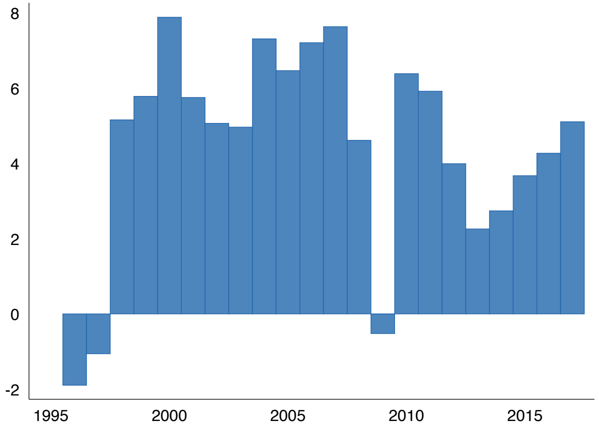
- Factor to index: use logs to sum over rows then exponentiate to multiply
- Index to factor: like levels

- Of course you can’t go back from factor/index to levels

Reindex: working example

```
* Gdp
* ===
sort geo year
list geo year gdp_ri if old == 1, clean

* BE index
cap drop gdp_ri
reindex gdp2010, generate(gdp_ri) i(cnum) j(year) ///
    from(levels) to(index) tobase(2008) toscale(100)

xtset cnum year
xtline gdp_ri if geo == "BE", scheme(burd)  xtitle("") ytitle("")
graph export gdp_ri_be.pdf, replace

* BE percent
cap drop gdp_rp
reindex gdp2010, generate(gdp_rp) i(cnum) j(year) ///
    from(levels) to(percent) toscale(100)

twoway bar gdp_rp year if geo == "BE", scheme(burd)  xtitle("") ytitle("")
graph export gdp_rp_be.pdf, replace
```





To do: reindex

- Use `xtset` instead of i() and j()
- How to deal with unbalanced data?