

EUDBIMPORT

Stata package to import EUROSTAT databases

Nicola Tommasi
C.I.D.E.
nicola.tommasi@univr.it

Indice

1	Fonte dei dati	1
2	Struttura dei databases	2
3	Scaricare tutti i database in locale	3
3.1	Come usare 7z in Stata	4
4	Costruzione dei do-file per gli items	4
5	Il file eudbimport_labvar.do	7
6	Regole per rinominare le variabili	7
6.1	Le frequenze presenti nei db	8
7	Sintassi di eudbimport	10
7.1	Esempi	11
7.2	Database di grandi dimensioni	18
8	Stats	21
9	To Do	22

Abstract

eudbimport è un nuovo comando Stata che permette

- di importare in Stata i database di EUROSTAT
- di trasformarli in serie storica
- di convertire in numeriche le variabili create con codifica dei dati missing
- di fare il label delle variabili del database
- di selezionare le osservazioni importate

Esiste anche un altro comando Stata per importare i dati da fonte EUROSTAT, `sdmxuse` di Sebastien Fontenay. Nella parte degli esempi mostrerò un raffronto tra `sdmxuse` e `eudbimport`.

1. Fonte dei dati

Tutti i database compatibili con il comando `eudbimport` sono reperibili a [questo indirizzo](#).

Per scaricare un database serve conoscere il suo nome, sul sito è indicato tra parentesi quadre come si vede in figura 1 dove sono elencati i database `EI_BSCO_M`, `EI_BSCO_Q` ed `EI_BSIN_M_R2`.



Figura 1 — Come reperire il nome di un database

È possibile avere l'elenco completo dei database seguendo queste istruzioni. Cliccate sul link del nuovo sito e poi su **DOWNLOADS** (vedi figura 2). Selezionate *Data*, in *Download operations* selezionate *Download full list of items* e infine su **Apply**. Verrà prodotto e proposto per il download il file *Full_Items_List_EN.txt*. Questo file può essere usato per scaricare tutti i database del sito (sono circa 6900).

In seguito verrà mostrato un do-file che a partire da questo file esegue il download completo dei database in esso elencati. Il download è piuttosto lungo, 7-8 ore, e scarica circa 50GB di dati (vedi [Scaricare tutti i database in locale](#) di pagina 3).

Per fare il label delle variabili servono una serie di altri files presenti nella sezione *Code lists* (vedi sempre figura 2). Questi files non sono necessari se non siete interessati al label delle variabili. Se invece volete il label dovete selezionare l'item relativo alla variabile che vi interessa e fare il download. Gli item dei codes lists sono circa 600 e per scaricarli si possono selezionare solo a blocchi di 100. Anche in questo caso verrà mostrato e descritto un do-file che si occupa di caricare i files relativi e creare dei do-file con lo scopo di fare il label delle variabili ([Costruzione dei do-file per gli items](#) di pagina 4).

L'ispirazione per questo comando viene da [qui](#), dall'ottimo e ricchissimo sito di Asjad Naqvi.

The screenshot shows the Eurostat Data Browser interface. At the top, there's a navigation bar with 'ALL DATA', 'RECENTLY UPDATED', and 'DOWNLOADS' (highlighted in yellow with a red arrow). To the right are links for 'Sign in', 'English EN', and a search bar. Below the navigation bar, the 'Download operations' section is visible. It includes tabs for 'Data', 'Code lists', and 'Metadata'. The 'Data' tab is selected (indicated by a red arrow). Under this tab, there's a dropdown menu for 'Download operations' with 'Download full list of items (.txt)' selected (indicated by a red arrow). To the right is a language selector set to 'English (en)' and an 'Apply' button (indicated by a red arrow). Below this, a summary bar shows 'Data', '0 items selected / 6893 items', and a filter input field. A table follows with columns: a checkbox, 'Code', 'Last data change', and 'Last structural change'. The first row of data shows 'AACT_ALI01' with dates '15/07/2022 23:00'.

Figura 2 – Come reperire la lista dei database

2. Struttura dei databases

Tutti i database presenti sul sito hanno una struttura di base costante. Vediamone un esempio e familiarizziamo con una serie di definizioni. Questo (figura 3) è il database MIGR_IMM3CTB. Nella parte superiore si vedono quelle che d'ora in poi chiameremo *selectionvars*. In questo caso sono *Geopolitical entity*, *Time*, *Country/region of birth*, *Time frequency*, *Unit of measure*, *Age Class*, *Age definition* e *Sex*. Nella parte inferiore si vedono i dati relativi a questa configurazione delle *selectionvars*:

- sulle righe le 34 specificazioni di Geopolitical entity
- sulle colonne gli anni selezionati in Time
- il valore Total di Country/region of birth
- il valore Total di Age Class
- il valore Age reached during the year di Age definition
- il valore total di Sex
- la frequenza è annuale
- l'unità di misura è Number

Il simbolo + vicino a ciascuna *selectionvars* indica che è possibile selezionare una diversa specificazione di quella variabile. Per esempio in *Age class* sono disponibili 27 diverse configurazioni delle classi di età (Total, Less than 5 years, From 5 to 9 years ...).

Il comando `eudbimport` sostanzialmente permette di trasformare questo database in una time serie, cioè trasforma la variabile temporale che qui è rappresentata nelle colonne in una variabile e trasforma i valori di una delle *selectionvars* nelle nuove variabili di colonna attraverso un reshape. Questa variabile è definita `reshapevar` e le sue specificazioni diventeranno nuove variabili.

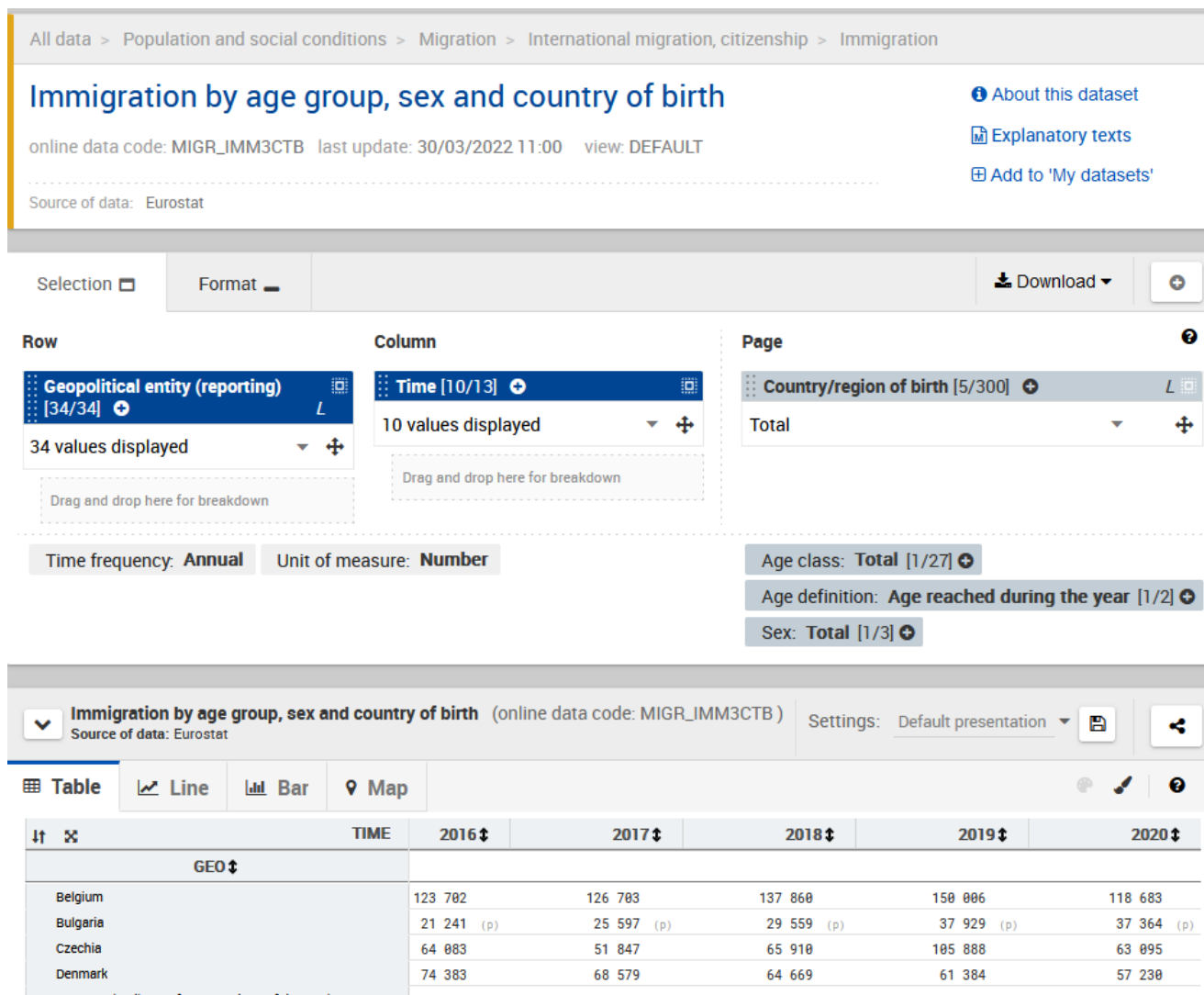


Figura 3 – La struttura dei database

3. Scaricare tutti i database in locale

Adesso mostro come scaricare in locale, cioè sul proprio Pc, tutti i database del sito EUROSTAT. Questa operazione non è necessaria perché il comando `eudbimport` prevede la possibilità di scaricare direttamente il file del database dal sito di EUROSTAT o, alternativamente, di importare il file precedentemente scaricato da una cartella del PC.

Per prima cosa dobbiamo scaricare il file `Full_Items_List_EN.txt` con l'elenco completo dei databases (circa 7000)¹. È stato descritto prima come procurarsi questo file, vedi la sezione **Fonte dei dati** di pagina 1.

```
import delimited "Full_Items_List_EN.txt", clear varnames(1)

qui count
forvalues i=1/`r(N)' {
    local urltsv = datadownloadurltsv in `i'
    local filename = code in `i'
    di "Download file `filename' - `i' of `r(N)'"
    copy "`urltsv'" `filename'.tsv, replace

    if "`c(os)'" == "Unix" shell 7zz a -t7z `filename'.7z `filename'.tsv -mx9 -bb1
    else shell "$E7z" a -t7z `filename'.7z `filename'.tsv -mx9 -bb1
}
```

1. 6893 il 10 settembre 2022

```
    erase `filename'.tsv
}
```

Il codice è abbastanza semplice. Con `import delimited` importiamo in Stata il file `txt`, poi inizia un ciclo che recupera una serie di informazioni da ciascuna riga del file importato. Nella `local urltsv` viene registrato l'url per accedere al database e nella `local filename` il suo nome. Poi il comando `copy` provvede a scaricare `filename` in locale con estensione `.tsv`. Le ultime 2 righe si occupano di comprimere `filename.tsv`.

Dato che alcuni database sono abbastanza grandi (`EF_M_FARMANG.tsv` ad esempio pesa circa 2.3GiB), è possibile zippare i singoli file `tsv` scaricati anche come opzione del comando. Per adesso ho implementato solo `7z` come metodo di compressione ma a breve aggiungerò anche il classico `zip`.

3.1. Come usare 7z in Stata

Il sito del programma è <https://www.7-zip.org/> dove trovate la versione per i diversi sistemi operativi. Installate quella adatta al vostro.

- in Windows:
 - installare il programma `7z`
 - nella cartella di installazione di Stata creare il file `profile.do` o se esiste già aggiungere questa riga:
`global E7z "C:\Program Files\7-Zip\7z.exe"`
Il percorso riportato è quello di installazione del programma, modificarlo di conseguenza se lo avete installato in una directory diversa.
- in Linux:
 - a seconda della vostra distribuzione installate il pacchetto `7zip-full`
 - poi non serve fare più nulla, in linux non serve intervenire nel file `profile.do`
- in MacOS:
 - installare il programma
 - booo!

4. Costruzione dei do-file per gli items

Nel capitolo [Fonte dei dati](#) di pagina 1 è stato descritto come scaricare dal sito di EUROSTAT i file relativi ai *Codes list*. Ora vediamo come sono strutturati. Sono dei files di testo con estensione `.tsv` e il nome è `ESTAT_<CODELIST>.tsv`² dove `CODELIST` è il nome della variabile a cui fanno riferimento i code list. Per esempio `ESTAT_ACCIDENT_en.tsv` riporta i codici relativi alla variabile *accident* (Accident). Questo è il suo contenuto:

TOTAL	Total
SRS	Serious accidents
SRS_F	Serious accidents - women
SRS_M	Serious accidents - men
FATAL	Fatal accidents
COLLIS	Collisions of trains, including collisions with obstacles within the clearance gauge
COLLIS_X_LVL_CROS	Collisions (excluding level-crossing accidents)
DERAIL	Deraillments of trains
DGD	Accidents involving transport of dangerous goods

2. `_en` finale è per la versione inglese del file, le altre possibili sono `_fr` e `_de`.

NDGD	Accidents not involving transport of dangerous goods
DGD_RL	Accidents in which dangerous goods are released
DGD_NRL	Accidents in which dangerous goods are not released
LVL_CROS	Level crossing accidents
RSTK_MOT	Accidents to persons caused by rolling stock in motion
RD_TRF	Road traffic
HOM_SCH	Accident at home / school / leisure
HOM_LEIS	Home and leisure
HOM	Home
LEIS	Leisure
RSTK_FIRE	Fires in rolling stock
FAT_NT	Fatalities in injury accidents on national territory (all operators)
ACC_NT	Injury accidents on national territory (all operators)
FAT_NC	Fatalities in injury accidents where a national company was involved (worldwide)
ACC_NC	Injury accidents where a national company was involved (worldwide)
OTH	Others
UNK	Unknown

Lo possiamo vedere come un database con due variabili. La prima variabile contiene i valori che può assumere la variabile *accident*, la seconda le relative descrizioni. Le due variabili sono separati da tabulazione. Ma perchè questi files sono utili? Perchè se *accident* sarà la variabile scelta per il *reshape* (opzione *reshapevar()*), le sue specifiche saranno le variabili create con il *reshape* e le descrizioni potranno essere usate per fare il label delle variabili. Questa operazione viene svolta dal file *db_items.do*, che viene di seguito riportato e commentato:

```
clear all
set more off
cls
```

```
capture mkdir items
capture mkdir dic
```

```
copy "https://ec.europa.eu/eurostat/api/dissemination/files/inventory?type=codelist&lang=en" "items/inventory_code_list.txt"
```

```
***? import & download items ***
**Code lists -> Download inventory -> save in /items with name inventory_code_list.txt
import delimited "items/inventory_code_list.txt", clear
```

```
qui count
forvalues j=1/`r(N)' {
    local link = specifictsvdownloadurl in `j'
    local name = code in `j'
    local name = "ESTAT_`name'.en.tsv"
    di "`name', `j' di `r(N)'"
    copy "`link'" "items/`name'", replace
}
```

```
/*****
/****! questa parte non c'è +. Si chiamava così ma si riferiva ai dati PRODCOM !****/
```

```
***? import & download items for comext ***
**Comext code lists -> Download inventory -> save inventory_code_comext.txt
import delimited "items/inventory_code_comext.txt", clear
qui count
forvalues j=1/`r(N)' {
    local code = code in `j'
    local link = "https://ec.europa.eu/eurostat/api/comext/dissemination/sdmx/2.1/codelist/ESTAT/`code'/latest?compress=1"
    local name = code in `j'
    local name = "ESTAT_`name'.en.tsv"
    di "`name', `j' di `r(N)'"
    copy "`link'" "items/`name'", text replace
}
*****/
```

```
****! QUESTI POI VANNO COPIATI IN GitHub/eudbimport/dic/
local itemslist : dir "items" files "*.tsv", respectcase
```

```

local nitems : word count `itemslst'
di `nitems'

foreach f of local itemslst {
    local item : subinstr local f "ESTAT_" ""
    local item : subinstr local item "_en.tsv" ""
    di "`item'"
    local item = lower("`item'")

    qui import delimited "items/`f'", clear encoding(UTF-8) stringcols(_all) delimiter(tab) varnames(nonames)

    **alcuni errori da correggere:
    if "`f'"=="ESTAT_INDIC_IN_en.tsv" replace v2=subinstr(v2,`"innovation"', "innovation",1)
    if "`f'"=="ESTAT_NET_SEG_en.tsv" replace v2=subinstr(v2,`"', "'",.)

    qui describe
    **assert r(k)==2
    assert r(k)>=2
    duplicates report v1
    **assert r(unique_value)==r(N)
    if r(unique_value)!=r(N) {
        duplicates tag v1, gen(dups)
        list v1 v2 if dups>=1
        drop dups
        duplicates drop v1, force
    }

    *! regola per avere nomi compatibili
    di "`f'"
    if "`f'" == "ESTAT_ICD10_en.tsv" {
        replace v1="C54__C55" if v1=="C54-C55"
        replace v1="F00__F03" if v1=="F00-F03"
        replace v1="G40__G41" if v1=="G40-G41"
    }
    if "`f'" == "ESTAT_LCSTRUCT_en.tsv" replace v1="D12__D4_MD5" if v1=="D12-D4_MD5"
    if "`f'" == "ESTAT_NACE_R1_en.tsv" {
        replace v1="C__E" if v1=="C-E"
        replace v1="L__Q" if v1=="L-Q"
    }
    if inlist("`f'", "ESTAT_NACE_R2_en.tsv", "ESTAT_NACE_R2D_en.tsv") {
        replace v1="B06__B09" if v1=="B06-B09"
        replace v1="O__U" if v1=="O-U"
    }
    if "`f'" == "ESTAT_UNIT_en.tsv" replace v1="MIO__EUR__NSA" if v1=="MIO-EUR-NSA"

    **forse è un errore la presenza di _2000W01 dato che sono date
    if "`f'" == "ESTAT_TIME_en.tsv" drop if v1=="_2000W01"

    replace v1 = ustrtoname(v1,1)

    duplicates examples v1
    duplicates drop v1, force
    duplicates report v1
    assert r(unique_value)==r(N)

    if "`item'"=="farmtype" {
        replace v2=subinstr(v2," (calculated with Standard Output)", "",1)
        replace v2=subinstr(v2," (calculated with Standard Gross Margin)", "",1)
    }
    if "`item'"=="bop_item" {
        replace v2 = subinstr(v2,"(Rural development support)", "",1)
        replace v2 = subinstr(v2,"; currency of denomination Euro", "",1)
        replace v2 = subinstr(v2,"; currency of denomination US dollar", "",1)
        replace v2 = subinstr(v2,"; currency of denomination Japanese yen", "",1)
        replace v2 = subinstr(v2,"; all currencies of denomination except Euro, US dollar and Japanese yen", "",1)
        replace v2 = subinstr(v2,"; all currencies of denomination except EUR and USD", "",1)
    }

    if "`item'"=="indic_ef" {replace v2 = subinstr(v2,"hold:", "",1)
        replace v2 = subinstr(v2,"ha:", "",1)
        replace v2 = subinstr(v2,"pers:", "",1)
        replace v2 = subinstr(v2,"Euro:", "",1)
        replace v2 = subinstr(v2,"head:", "",1)
        replace v2 = subinstr(v2,"places:", "",1)
        replace v2 = subinstr(v2,"LSU:", "",1)
        replace v2 = subinstr(v2,"Nb:", "",1)
    }
}

```



```

    replace v2 = substr(v2,"AWU:", "",1)
    replace v2 = substr(v2,"(Rural development support)", "",1)
}

gen labelvar = "cap label var " + v1 + `"' + v2 + `"'

**variable è una reserved word, quindi si rinomina in VARIABLE
if "`item'"=="variable" local item VARIABLE
outfile labelvar using "dic/labvar_`item'.do", replace noquote
}

exit

```

Una volta scaricati tutti i files relativi ai code list nella cartella items, si crea una lista di questi files da passare ad un ciclo. Nel ciclo, dal nome del file si isola il nome della variabile a cui fa riferimento (`local item`), si importa il contenuto del file (`import delimited`) e si verifica che non ci siano duplicati. Questo controllo è importante altrimenti poi ci sarebbero due o più variabili con lo stesso nome. Le specifiche assunte dalle diverse variabili non sempre sono compatibili con le regole relative ai nomi delle variabili in Stata. Il grosso di queste incompatibilità viene risolto dalla funzione `ustrtoname()` che opera tre tipi di modifiche

- aggiunge `_` (underscore) se una specifica inizia con un numero. Si vedano ad esempio le specifiche di `item_newa`.
- sostituisce il carattere `-` con `_` (underscore) se una specifica contiene tale carattere. Si vedano ad esempio le specifiche di `icd10`.
- tronca la specifica al trentaduesimo carattere nel caso fosse più lunga

Rimangono pochi casi da risolvere manualmente, dovuti principalmente al fatto che le modifiche prodotte da `ustrtoname()` generano dei duplicati. Ad esempio in `icd10` esistono sia la specifica `C54-C55` che `C54_C55`. `ustrtoname()` converte la prima in `C54_C55` creando un duplicato. In questi casi si interviene aggiungendo un secondo `_` in modo che `C54-C55` diventi `C54__C55`. Anche il comando `eudbimport` esegue le stesse operazioni.

A questo punto si crea la variabile `labelvar`, ovvero una stringa che contiene il comando `label` variabile relativo alla variabile. Poi con `outfile` questa variabile viene esportata in un do-file (`labvar_<nomevar>.do`). Questo per esempio è il risultato relativo alla variabile `acomsizes`:

```

cap label var CODE "Label - English"
cap label var TOTAL "Total"
cap label var LT10 "Less than 10 bedplaces"
cap label var GE10 "10 bedplaces or more"

```

Tutti i files `labvar_<nomevar>.do` sono disponibili sul mio account GitHub e chiamati direttamente dal comando `eudbimport`.

5. Il file `eudbimport_labvar.do`

Contiene il label di tutte le variabili presenti nei vari database. Non è strettamente necessario, il label può essere fatto anche manualmente dopo l'esecuzione del comando. Se manca qualche label, può essere aggiunto editando il do-file e aggiungendo una riga di comando con lo schema `capture label var <varname> "description"`.

6. Regole per rinominare le variabili

La variabile scelta nell'opzione `reshapevar()` può non essere compatibile con le regole che Stata impone per i nomi delle variabili. I motivi sono essenzialmente due, più un terzo che si verifica solo una volta:

- nomi che iniziano con un numero
- nomi che contengono il carattere ”-”
- nomi che corrispondono a reserved word di Stata³

Per i primi due casi viene usata la funzione `ustrtoname(s,1)` che converte tutti i caratteri non ammessi in Stata con l’_, che aggiunge _ se un nome inizia con un carattere numerico e che tronca il nome a 32 caratteri.

L’utilizzo di `ustrtoname(s,1)` è stato modificato nei seguenti casi per evitare casi di nomi duplicati in seguito alla modifica apportata dalla funzione

- nell’item ICD10
 - C54-C55 convertito in C54__C55
 - F00-F03 convertito in F00__F03
 - G40-G41 convertito in G40__G41
- nell’item LCSTRUCT
 - D12-D4_MD5 convertito in D12__D4_MD5
- nell’item NACE_R1
 - C-E convertito in C__E
 - L-Q convertito in L__Q
- nell’item NACE_R2
 - B06-B09 convertito in B06__B09
 - O-U convertito in O__U
- nell’item NA_ITEM
 - D2_D5_D91_D61_M_D611V_D612_M_M_D613V_D614_M_D995 convertito in D2_D5_D91_-D61_M_D611V_D612_M_M_D
 - D2_D5_D91_D61_M_D612_M_D614_M_D995 convertito in D2_D5_D91_D61_M_D612_M_-D614_M_D9
- nell’item UNIT
 - MIO-EUR-NSA convertito in MIO__EUR__NSA

6.1. Le frequenze presenti nei db

I dati presenti nei vari database hanno frequenze temporali diverse. Quasi sempre la frequenza è unica, in alcuni casi sono presenti contemporaneamente più frequenze, per esempio dati mensili e trimestrali. `eudbimport` converte la frequenza presente nel database nel corrispettivo formato numerico di Stata. Questo però non è possibile se ci sono frequenze multiple e quindi, in questo caso, le frequenze rimangono come variabili stringa. La tabella 1 mostra, per le diverse frequenze, il formato di come appaiono nei dati di EUROSTAT e di come vengono modificati per renderli compatibili con Stata. La variabile temporale creata si chiama `data`.

Nel caso di database a frequenze multiple, le frequenze vengono modificate come segue:

3. Fino ad ora ho trovato un solo caso che riguarda il nome `variable` e che è stato convertito in `VARIABLE`

Tabella 1 – Formati delle date

Tipo	Formato in EUROSTAT	Etichetta	Stata compliant	Stata format
Dati giornalieri	####-##-##	D	YYYYMMDD	%td
Dati settimanali	####-W#	W	YYYYwW	%tw
Dati mensili	####-##	M	YYYYmMM	%tm
Dati trimestrali	####-Q#	Q	YYYYqQ	%tq
Dati semestrali	####-S#	S	YYYYhH	%th
Dati annuali	####	A	YYYY	%ty

Dati giornalieri: Y####M##D####

Dati settimanali: Y####W##

Dati mensili: Y####M##

Dati trimestrali: Y####Q#

Dati semestrali: Y####H#

Dati annuali: Y####

Quindi possiamo avere i seguenti casi:

1. Database con frequenza unica. La frequenza viene convertita in formato numerico Stata compliant
2. Database con frequenze diverse, per esempio con dati mensili, trimestrali e annuali. In questo caso ci sono 2 possibili scenari:
 - (a) tramite l'opzione `select()` si seleziona una sola frequenza e si ricade nel caso 1.
 - (b) si tengono le frequenze diverse. In questo caso la variabile `freq` rimane stringa e nel database finale rimangono dati con frequenze diverse.

In alcuni database la variabile con la frequenza presenta dei valori non compatibili con la frequenza stessa. Per esempio nel database `ENV_AIR_ESD` la frequenza è annuale ma è presente anche `TARGET` nell'elenco degli anni. In questi casi (per adesso solo per dati annuali), il valore viene convertito in 3000 e associato con una label che riproduce il valore non compatibile.

```
eudbimport ENV_AIR_ESD, rawdata("data/raw_data/") outdata("data/out_data/") ///
  reshapevar(unit) download
I'm downloading the file...
I'm importing data...
```

```
Database: ENV_AIR_ESD
Selection's variables: freq unit geo
Time Period: A
Reshape variable: unit
I'm reshaping long...
I'm reshaping wide...
I'm destringing variables...
```

```
. freq date
```

```
date -- Time
```

		Freq.	Percent	Valid	Cum.
Valid	2005	30	5.91	5.91	5.91
	2006	30	5.91	5.91	11.81
	2007	30	5.91	5.91	17.72
	2008	30	5.91	5.91	23.62
	2009	30	5.91	5.91	29.53
	2010	30	5.91	5.91	35.43
	2011	30	5.91	5.91	41.34
	2012	30	5.91	5.91	47.24
	2013	30	5.91	5.91	53.15

2014		30	5.91	5.91	59.06
2015		30	5.91	5.91	64.96
2016		30	5.91	5.91	70.87
2017		30	5.91	5.91	76.77
2018		30	5.91	5.91	82.68
2019		30	5.91	5.91	88.58
2020		29	5.71	5.71	94.29
3000 TARGET		29	5.71	5.71	100.00
Total		508	100.00	100.00	

7. Sintassi di eudbimport

Per installare il comando, dalla command bar di Stata digitate:

```
net install eudbimport, from(https://raw.githubusercontent.com/NicolaTommasi8/eudbimport/master)
```

Ora il comando dovrebbe essere disponibile e pronto all'uso.

La sintassi del comando è la seguente:

```
eudbimport DBNAME, reshapevar(varname) [rawdata(string) outdata(string) download
select(string) timeselect(string) nosave erase strrec
dollar(string) compressed nodestring /*undocumented*/
debug /*undocumented*/ ]
```

dove:

DBNAME è il nome del database come riportato sul sito EUROSTAT (può essere indicato sia in maiuscolo che in minuscolo).

reshapevar() è la variabile usata nel reshape e le cui specifiche diventano le nuove variabili.

rawdata() è il percorso dove verrà scaricato il file del database se si usa l'opzione download o dove trovare il file DBNAME se scaricato manualmente. Il percorso va specificato tra virgolette e con / finale. Se non viene specificato il comando cercherà il file DBNAME nella directory di lavoro corrente.

outdata() è il percorso dove verrà salvato il file del database. Il percorso va specificato tra virgolette e con / finale. Se non viene specificato il comando salverà il file DBNAME nella directory di lavoro corrente.

download specifica che DBNAME deve essere scaricato dal sito di EUROSTAT.

select() specifica un sottoinsieme di osservazioni di DBNAME che devono essere importate. Si possono usare tutti i comandi di Stata per selezionare osservazioni (keep, drop...).

timeselect() specifica l'intervallo temporale da importare.

nosave specifica che il database importato non venga salvato.

erase specifica che il file DBNAME.tsv venga eliminato.

strrec converte in numeriche alcune variabili stringa. La scelta di queste variabili si basa sulla opportunità di conversione in numeriche in base alla tipologia della variabile e alla frequenza con cui è presente nei diversi dataset (*work in progress*).

dollar() opzione necessaria per importare i databases definiti come Type=EXTRACTION. Sono databases con il simbolo del dollaro (\$) nel nome come ad esempio BD_9N_R2\$DV_346. In questi casi si indica la parte prima del simbolo \$ come nome del database (BD_9N_R2) e la parte con il dollaro nell'opzione dollar() preceduta dal carattere \, in questa maniera dollar(\\$DV_346)

compressed scarica il dataset in formato compresso (.gz). Per l'estrazione del file dall'archivio compresso si ricorre ad un piccolo script in Python⁴ che quindi deve essere installato e configurato sul computer in uso. Questa opzione è particolarmente utile per i databases di grosse dimensioni come per esempio EF_M_FARMAG che pesa circa 8.2GB.

info visualizza informazioni relative al database indicato per il download. Queste comprendono, oltre al nome, la descrizione in inglese, francese e tedesco, la data dell'ultimo aggiornamento e dell'ultima modifica, il periodo coperto, il numero di valori presenti, il link per accedere ai metadati in formato html o sdmx, il link per il download del database in formato tsv o sdmx.

eudbimport esegue la conversione in numeriche delle variabili originate dalla variabile indicata in `reshapevar()`. In accordo con quanto riportato sul sito di EUROSTAT, i valori missing sono così classificati:

- .a not available
- .b break in time series
- .c confidential
- .d definition differs
- .e estimated
- .f forecast
- .i imputed
- .n not significant
- .p provisional
- .s Eurostat estimate
- .u low reliability
- .z not applicable

Sono possibili anche delle combinazioni dei casi precedenti, per esempio be (break in time series, estimated). In questi casi, il codice del caso missing è quello della prima lettera (.c nell'esempio precedente)⁵.

In alcuni casi il destrring non funziona perché nella stessa variabile c'è una commistione di dati numerici e non. Per esempio nel database **INN_C1012** (Number of innovating enterprises supported by government, by size class) se la variabile scelta come `reshapevar` è *unit*, in alcune variabili, oltre ai dati numerici sono presenti le stringhe "low", "med_low", "med_high" e "high". In questi casi non viene fatto il destrring e la variabile viene lasciata come stringa. Si lascia all'utilizzatore la scelta di come trattare questi casi.

eudbimport necessita dei comandi del pacchetto **gtools** (`greshape long`, `greshape wide` e `glevelsof`), del comando **missings** e del comando **fre**. Tutti i comandi aggiuntivi necessari vengono installati automaticamente da eudbimport.

7.1. Esempi

Partiamo dall'importazione del database NAMA_10_GDP, GDP and main components (output, expenditure and income) che potete visualizzare [qui](#).

```
eudbimport NAMA_10_GDP, download outdata("data/out_data/") reshapevar(na_item)
I'm downloading the file...
I'm importing data...
```

```
Database: NAMA_10_GDP
```

4. Lo script usa i pacchetti `gzip` e `shutil` di Python.

5. [Riferimento per le flags dei dati](#)

```

Selection's variables: freq unit na_item geo
Time Period: A
Reshape variable: na_item
I'm reshaping long...
I'm reshaping wide...
I'm destringing variables...

```

summ

Variable	Obs	Mean	Std. dev.	Min	Max
freq	0				
unit	0				
geo	0				
date	33,009	2006.497	9.710072	1975	2021
B11	4,837	45418.51	180214	-912332	3043369
B111	4,781	18673.14	130834.4	-1361881	1183899
B112	4,781	27554.3	128647.7	-105596.3	2579556
B1G	32,096	777400.2	2600605	-53.6	4.69e+07
B1G0	32,787	853907.9	2918289	-52.1	5.53e+07
B2A3G	6,551	438581.5	1334048	0	2.45e+07
D1	6,551	496025.6	1481134	0	2.22e+07
D11	6,350	401562.4	1196030	0	1.95e+07
D12	6,242	108352.6	312267.7	0	3116823
D2	3,734	86425.84	447051.6	4.8	9768440
D21	24,290	54305.72	340910.1	-43.5	8653925
D21X31	31,893	96711.92	366199	-43.5	8317458
D2X3	6,551	124072.2	409677.3	0	8523261
D3	3,734	12500.89	66591.66	0	1269050
D31	24,194	46114.57	6666347	-80.7	1.04e+09
P3	32,532	650966.2	2196996	-49.3	3.79e+07
P31_S13	30,005	104290.2	352255.7	-50.7	5218086
P31_S14	31,032	437485.3	1528685	-49.6	2.55e+07
P31_S14_S15	32,547	473729.7	1590562	-49.6	2.66e+07
P31_S15	30,663	11734.05	44968.61	-68.9	1122674
P32_S13	30,005	69735.11	268345.3	-44.7	6094656
P3_P5	30,597	879162.4	2931217	-49.6	5.48e+07
P3_P6	30,577	1275103	4600263	-46.7	9.98e+07
P3_S13	32,580	177546.1	608007	-47.8	1.13e+07
P41	29,947	564113.2	1948056	-49.8	3.18e+07
P51G	32,551	185148.6	643754.5	-67.7	1.50e+07
P52	5,189	7012.944	51668.61	-583396	1866392
P52_P53	8,276	8789.521	46173.74	-556516	1921255
P53	4,642	786.0776	4422.157	-16464.2	96626
P5G	32,149	193008	679006.2	-94.6	1.69e+07
P6	32,578	377031.5	1722949	-47.6	4.49e+07
P61	31,956	287547.1	1360826	-55.9	3.70e+07
P62	31,956	95248.29	386816.5	-59.7	8850062
P7	32,578	358581.2	1637718	-39.91	4.45e+07
P71	31,956	279715.8	1345908	-33.01	3.84e+07
P72	31,956	84000.17	319504	-60.4	6457932
YA0	3,782	-139.0197	2177.819	-38881.5	40610.5
YA1	4,178	19.69667	569.3599	-2400	22109.2
YA2	2,259	26.995	666.5498	-1057.1	22109.2

Se voglio importare solo le variabili che iniziano con B11 posso usare l'opzione `select()`. Attenzione alla logica usata in questo passaggio. La `reshapevar` è `na_item` e quindi saranno le specifiche di questa variabile che diventeranno i nomi delle nuove variabili del database importato.

```

eudbimport NAMA_10_GDP, reshapevar(na_item) select(keep if strmatch(na_item,"B11*"))
I'm importing data...

```

```

Database: NAMA_10_GDP
Selection's variables: freq unit na_item geo
Time Period: A
Reshape variable: na_item
I'm reshaping long...

```

I'm reshaping wide...
I'm destringing variables...

. summ

Variable	Obs	Mean	Std. dev.	Min	Max
freq	0				
unit	0				
geo	0				
date	4,837	2006.634	9.615158	1975	2021
B11	4,837	45418.51	180214	-912332	3043369
B111	4,781	18673.14	130834.4	-1361881	1183899
B112	4,781	27554.3	128647.7	-105596.3	2579556

Ora aggiungo come ulteriore condizione di selezionare solo CP_MEUR tra i valori della variabile *unit*

```
eudbimport NAMA_10_GDP, reshapevar(na_item) select(keep if strmatch(na_item,"B11*") & unit=="CP_MEUR")
I'm importing data...
```

Database: NAMA_10_GDP
Selection's variables: freq unit na_item geo
Time Period: A
Reshape variable: na_item
I'm reshaping long...
I'm reshaping wide...
I'm destringing variables...

. summ

Variable	Obs	Mean	Std. dev.	Min	Max
freq	0				
unit	0				
geo	0				
date	1,240	2006.494	9.640844	1975	2021
B11	1,240	36744.87	98376.68	-62146	559387.8
B111	1,214	24533.9	78313.61	-162093	436725.9
B112	1,214	13237.09	38398.91	-105596.3	320052.7

. fre unit

unit -- Unit of measure

		Freq.	Percent	Valid	Cum.
Valid	CP_MEUR	1240	100.00	100.00	100.00

Questo è un esempio di importazione di un database di tipo EXTRACTION (BD_9AC_L_FORM_R2\$DV_343)

```
. eudbimport BD_9AC_L_FORM_R2, rawdata("data/raw_data/") outdata("data/out_data/") ///
nosave erase download dollar(\$DV_343)
I'm downloading the file...
I'm importing data...
```

Database: BD_9AC_L_FORM_R2\$DV_343
Selection's variables: freq indic_sb leg_form nace_r2 geo
Time Period: A
Reshape variable: indic_sb
I'm reshaping long...
I'm reshaping wide...
I'm destringing variables...

Vediamo adesso degli esempi per ovviare ad alcuni problemi con certi database. Il database **AVIA_GOEXCC** (International extra-EU freight and mail air transport by reporting country and partner world regions and countries) è piuttosto grande (132MB) e multifrequenza, ovvero contiene dati riferiti a unità temporali diverse (dati mensili, trimestrali e annuali).

Usato così, il comando genera *date* come variabile stringa e ci mette un po' di tempo a completare la procedura di importazione

```
. eudbimport AVIA_GOEXCC, rawdata("data/raw_data/") outdata("data/out_data/") ///
> reshapevar(tra_meas)
I'm importing data...

Database: AVIA_GOEXCC
Selection's variables: freq unit tra_meas partner geo
Time Period: A M Q
Reshape variable: tra_meas
I'm reshaping long...
I'm reshaping wide...
I'm destringing variables...
```

```
fre freq
```

```
freq -- Time frequency
```

		Freq.	Percent	Valid	Cum.
Valid	A	86813	9.35	9.35	9.35
	M	601389	64.76	64.76	74.11
	Q	240432	25.89	25.89	100.00
	Total	928634	100.00	100.00	

```
. codebook date
```

```
-----
date                                                    Time
-----
```

```
      Type: String (str8)
```

```
Unique values: 425
```

```
Missing "": 0/928,634
```

```
Examples: "Y2006Q4"
          "Y2010M12"
          "Y2014M05"
          "Y2018"
```

Se seleziono una sola frequenza, la variabile *date* diventa numerica... e ci mette molto meno tempo

```
. eudbimport AVIA_GOEXCC, rawdata("data/raw_data/") outdata("data/out_data/") select(keep if freq=="A") ///
> nodestring reshapevar(tra_meas)
I'm importing data...

Database: AVIA_GOEXCC
Selection's variables: freq unit tra_meas partner geo
Time Period: A
Reshape variable: tra_meas
I'm reshaping long...
I'm reshaping wide...
```

```
. fre freq
```

```
freq -- Time frequency
```

		Freq.	Percent	Valid	Cum.
Valid	A	216949	100.00	100.00	100.00

```
. codebook date
```

```
-----
date                                                    Time
-----
```

```
      Type: Numeric yearly date (int)
```

```
Range: [1993,2021]
Or equivalently: [1993,2021]
Unique values: 29
```

```
Units: 1
Units: years
Missing : 0/216,949
```


Mean:	2007 = 2007				
Std. dev.:	8.36662				
Percentiles:	10%	25%	50%	75%	90%
	1995	2000	2007	2014	2019
	1995	2000	2007	2014	2019

Ci sono altri 2 problemi legati al numero di variabili: il primo riguarda il numero di variabili presenti nel file .tsv da importare, il secondo riguarda il numero di variabili che entrano nel reshape long e nel reshape var.

Il primo è facilmente risolvibile con il comando set maxvar #. Vediamo l'esempio del database **ERT_BIL_EUR_D** (Euro/ECU exchange rates - daily data) che ha più di 12000 colonne:

```
. eudbimport ERT_BIL_EUR_D, rawdata("data/raw_data/") outdata("data/out_data/") reshapevar(statinfo)
I'm importing data...
There are more columns than allowed by this Stata. See help maxvar for more information.
r(900);
```

che si risolve così:

```
clear all

. set maxvar 15000

. eudbimport ERT_BIL_EUR_D, rawdata("data/raw_data/") outdata("data/out_data/") reshapevar(statinfo)
I'm importing data...

Database: ERT_BIL_EUR_D
Selection's variables: freq statinfo unit currency
Time Period: D
Reshape variable: statinfo
I'm reshaping long...
characteristic contents too long
The maximum value of the contents is 67,784.
r(1004);
```

ed ecco il secondo problema. In Stata ogni variabile è associata ad un insieme di caratteristiche indicate come varname[charname], dove varname è il nome della variabile e charname sono una stringa di testo come le caratteristiche della variabile. Nei limits di Stata possiamo leggere che:

char	length of one characteristic (bytes)	67,784	67,784
------	---	--------	--------

Se l'elenco del numero di variabili che entrano nel reshape supera la lunghezza dei 67,784 bytes, Stata termina l'esecuzione del comando con l'errore r(1004). L'unica soluzione è restringere il numero di variabili usate nel reshape. Se l'errore avviene durante il reshape long dobbiamo limitare il periodo temporale usando l'opzione timeselect(). Qui specifico di prendere tutte le date relative all'anno 1975

```
eudbimport ERT_BIL_EUR_D, rawdata("data/raw_data/") outdata("data/out_data/") reshapevar(statinfo) ///
timeselect(1975)
I'm importing data...

Database: ERT_BIL_EUR_D
Selection's variables: freq statinfo unit currency
Time Period: D
Reshape variable: statinfo
I'm reshaping long...
I'm reshaping wide...
I'm destringing variables...
```

```
. fre date
```

```
date -- Time
```

		Freq.	Percent	Valid	Cum.
Valid	02jan1975	6	0.33	0.33	0.33
	03jan1975	6	0.33	0.33	0.67
	06jan1975	6	0.33	0.33	1.00
	07jan1975	6	0.33	0.33	1.33
	08jan1975	6	0.33	0.33	1.67
	09jan1975	6	0.33	0.33	2.00
	10jan1975	6	0.33	0.33	2.33
	13jan1975	6	0.33	0.33	2.67
	14jan1975	6	0.33	0.33	3.00
	15jan1975	6	0.33	0.33	3.34
	16jan1975	6	0.33	0.33	3.67
	17jan1975	6	0.33	0.33	4.00
	20jan1975	6	0.33	0.33	4.34
	21jan1975	6	0.33	0.33	4.67
	22jan1975	6	0.33	0.33	5.00
	23jan1975	6	0.33	0.33	5.34
	24jan1975	6	0.33	0.33	5.67
	27jan1975	6	0.33	0.33	6.00
	28jan1975	6	0.33	0.33	6.34
	29jan1975	6	0.33	0.33	6.67
	:	:	:	:	:
	02dec1975	8	0.44	0.44	91.61
	03dec1975	8	0.44	0.44	92.05
	04dec1975	8	0.44	0.44	92.50
	05dec1975	8	0.44	0.44	92.94
	08dec1975	8	0.44	0.44	93.39
	09dec1975	8	0.44	0.44	93.83
	10dec1975	8	0.44	0.44	94.27
	11dec1975	8	0.44	0.44	94.72
	12dec1975	8	0.44	0.44	95.16
	15dec1975	8	0.44	0.44	95.61
	16dec1975	8	0.44	0.44	96.05
	17dec1975	8	0.44	0.44	96.50
	18dec1975	8	0.44	0.44	96.94
	19dec1975	8	0.44	0.44	97.39
	22dec1975	8	0.44	0.44	97.83
	23dec1975	8	0.44	0.44	98.28
	24dec1975	7	0.39	0.39	98.67
	29dec1975	8	0.44	0.44	99.11
	30dec1975	8	0.44	0.44	99.56
	31dec1975	8	0.44	0.44	100.00
Total		1799	100.00	100.00	

oppure così:

```
eudbimport ERT_BIL_EUR_D, rawdata("data/raw_data/") outdata("data/out_data/") reshapevar(statinfo) ///
timeselect(19750102-19800102)
I'm importing data...
```

```
Database: ERT_BIL_EUR_D
Selection's variables: freq statinfo unit currency
Time Period: D
Reshape variable: statinfo
I'm reshaping long...
I'm reshaping wide...
I'm destringing variables...
```

```
. fre date
```

```
date -- Time
```

		Freq.	Percent	Valid	Cum.
Valid	02jan1975	6	0.06	0.06	0.06
	03jan1975	6	0.06	0.06	0.12
	06jan1975	6	0.06	0.06	0.18

07jan1975	6	0.06	0.06	0.24
08jan1975	6	0.06	0.06	0.31
09jan1975	6	0.06	0.06	0.37
10jan1975	6	0.06	0.06	0.43
13jan1975	6	0.06	0.06	0.49
14jan1975	6	0.06	0.06	0.55
15jan1975	6	0.06	0.06	0.61
16jan1975	6	0.06	0.06	0.67
17jan1975	6	0.06	0.06	0.73
20jan1975	6	0.06	0.06	0.79
21jan1975	6	0.06	0.06	0.85
22jan1975	6	0.06	0.06	0.92
23jan1975	6	0.06	0.06	0.98
24jan1975	6	0.06	0.06	1.04
27jan1975	6	0.06	0.06	1.10
28jan1975	6	0.06	0.06	1.16
29jan1975	6	0.06	0.06	1.22
:	:	:	:	:
30nov1979	8	0.08	0.08	98.45
03dec1979	8	0.08	0.08	98.54
04dec1979	8	0.08	0.08	98.62
05dec1979	8	0.08	0.08	98.70
06dec1979	8	0.08	0.08	98.78
07dec1979	8	0.08	0.08	98.86
10dec1979	8	0.08	0.08	98.94
11dec1979	8	0.08	0.08	99.02
12dec1979	8	0.08	0.08	99.11
13dec1979	8	0.08	0.08	99.19
14dec1979	8	0.08	0.08	99.27
17dec1979	8	0.08	0.08	99.35
18dec1979	8	0.08	0.08	99.43
19dec1979	8	0.08	0.08	99.51
20dec1979	8	0.08	0.08	99.59
21dec1979	8	0.08	0.08	99.67
27dec1979	8	0.08	0.08	99.76
28dec1979	8	0.08	0.08	99.84
31dec1979	8	0.08	0.08	99.92
02jan1980	8	0.08	0.08	100.00
Total	9835	100.00	100.00	

Bisogna fare attenzione a due cose: le date vanno indicate come da colonna "Stata compliant" nella tabella 1 e devono essere date esistenti. Per esempio, se nel caso precedente indichiamo 19750101 come prima data, verrà restituito un errore perché quella data non esiste nel database.

Abbiamo visto come non sia possibile, a causa di alcune limitazioni di Stata, importare l'intero database ERT_BIL_EUR_D. Possiamo aggirare il problema importandolo a pezzi per poi riassemblare il tutto. I dati sono giornalieri, partono dal 1974 e arrivano fino al 2022. Con un ciclo è possibile importare anno per anno e alla fine unire il tutto in un unico database:

```
forvalues YY=1974/2022 {
  if `YY'==1974 eudbimport ERT_BIL_EUR_D, download nosave reshapevar(statinfo) timeselect(`YY')
  else eudbimport ERT_BIL_EUR_D, nosave reshapevar(statinfo) timeselect(`YY')

  if `YY'==1974 frame copy default ERT_BIL_EUR_D, replace
  else {
    frame copy default temp
    frame change ERT_BIL_EUR_D
    frameappend temp, drop
    desc, short
    frame change default
  }
}

frame change ERT_BIL_EUR_D
```

Alla fine otteniamo il database con tutte le date presenti in ERT_BIL_EUR_D

```
. summ date, format
```

Variable	Obs	Mean	Std. dev.	Min	Max
date	316,586	09oct2005	4182.735	01jul1974	14sep2022

7.2. Database di grandi dimensioni

Ci sono databases di grandi dimensioni come EF_M_FARMAG che pesa circa 8.2GB⁶. In questi casi il download del file va comunque eseguito (magari usando l'opzione compressed), però nella fase di importazione possiamo usare l'opzione select() per importare solo una parte dei dati. Di seguito mostro come importare l'intero dataset e come importare un sottoinsieme di dati.

```
eudbimport EF_M_FARMANG, rawdata("data/raw_data/") outdata("data/out_data/") debug ///
  reshapevar(farmtype) compressed
I'm importing data...
```

```
Database: EF_M_FARMANG
Selection's variables: freq sex age crops farmtype agrarea so_eur unit geo
Time Period: A
Reshape variable: farmtype
I'm reshaping long...
I'm reshaping wide...
I'm destringing variables...
```

Contains data from data/out_data/EF_M_FARMANG.dta

```
Observations: 21,269,576
Variables: 32 28 Oct 2022 18:27
```

Variable name	Storage type	Display format	Value label	Variable label
freq	str1	%9s		Time frequency
sex	str3	%9s		Sex
age	str6	%9s		Age class
crops	str6	%9s		Crops
agrarea	str8	%9s		Agricultural area
so_eur	str9	%9s		Standard output in Euros
unit	str3	%9s		Unit of measure
geo	str9	%9s		Geopolitical entity (reporting)
date	int	%ty		Time
FT15_S0	float	%9.0g		Specialist cereals, oilseed and protein crops
FT16_S0	float	%9.0g		General field cropping
FT21_S0	float	%9.0g		Specialist horticulture indoor
FT22_S0	float	%9.0g		Specialist horticulture outdoor
FT23_S0	float	%9.0g		Other horticulture
FT35_S0	float	%9.0g		Specialist vineyards
FT36_S0	float	%9.0g		Specialist fruit and citrus fruit
FT37_S0	float	%9.0g		Specialist olives
FT38_S0	float	%9.0g		Various permanent crops combined
FT45_S0	float	%9.0g		Specialist dairying
FT46_S0	float	%9.0g		Specialist cattle-rearing and fattening
FT47_S0	float	%9.0g		Cattle-dairying, rearing and fattening combined
FT48_S0	float	%9.0g		Sheep, goats and other grazing livestock
FT51_S0	float	%9.0g		Specialist pigs
FT52_S0	float	%9.0g		Specialist poultry
FT53_S0	float	%9.0g		Various granivores combined
FT61_S0	float	%9.0g		Mixed cropping
FT73_S0	float	%9.0g		Mixed livestock, mainly grazing livestock
FT74_S0	float	%9.0g		Mixed livestock, mainly granivores
FT83_S0	float	%9.0g		Field crops-grazing livestock combined
FT84_S0	float	%9.0g		Various crops and livestock combined
FT90_S0	float	%9.0g		Non-classified farms
TOTAL	float	%9.0g		Total

Sorted by:

Variable	Obs	Mean	Std. dev.	Min	Max
freq	0				
sex	0				
age	0				

6. Anche fare il solo download del file sono diversi minuti.

crops		0			
agrarea		0			

so_eur		0			
unit		0			
geo		0			
date		21,269,576	2018.146	1.994632	2016 2020
FT15_S0		6,800,307	4555482	1.53e+08	0 3.93e+10

FT16_S0		9,397,242	3102172	1.16e+08	0 3.44e+10
FT21_S0		2,950,312	4161611	1.05e+08	0 1.81e+10
FT22_S0		3,123,069	1693003	4.11e+07	0 7.91e+09
FT23_S0		2,843,864	2331582	5.26e+07	-140 9.89e+09
FT35_S0		3,733,136	5300117	1.20e+08	0 2.40e+10

FT36_S0		4,822,471	2354651	5.90e+07	0 1.55e+10
FT37_S0		1,867,197	3888299	6.87e+07	0 1.37e+10
FT38_S0		3,029,256	1368960	2.80e+07	0 6.40e+09
FT45_S0		5,651,929	9452743	2.82e+08	0 5.96e+10
FT46_S0		6,832,685	2224367	6.82e+07	0 1.80e+10

FT47_S0		3,094,734	1634551	3.65e+07	0 6.29e+09
FT48_S0		7,181,087	1580967	4.63e+07	0 1.40e+10
FT51_S0		3,335,708	8357932	2.21e+08	0 3.96e+10
FT52_S0		3,125,627	5672632	1.44e+08	0 2.67e+10
FT53_S0		1,410,948	927759	1.54e+07	0 1.81e+09

FT61_S0		5,174,438	2023373	5.18e+07	0 1.26e+10
FT73_S0		2,849,471	1526068	3.13e+07	0 4.98e+09
FT74_S0		1,965,758	1947322	3.86e+07	0 4.49e+09
FT83_S0		4,430,422	3107667	9.06e+07	0 1.68e+10
FT84_S0		5,499,319	2053186	5.46e+07	0 1.23e+10

FT90_S0		1,343,277	387.1649	7899.079	0 1321180
TOTAL		18,937,776	1.63e+07	8.13e+08	-110 3.69e+11

Elapsed time was 3 hours, 14 minutes, 18.15 seconds.

Il tempo necessario per importare l'intero dataset è di circa 3 ore e 15 minuti e il file finale pesa circa 3GB. Nell'esempio che segue, tra le Geopolitical entity importo solo i dati a livello nazionale e seleziono solo "T" relativamente alla variabile sex usando l'opzione select()

```
eudbimport EF_M_FARMANG, rawdata("data/raw_data/") outdata("data/out_data/") debug ///
reshapevar(farmtype) select(keep if strlen(geo)==2 & sex=="T") compressed
I'm importing data...
```

```
Database: EF_M_FARMANG
Selection's variables: freq sex age crops farmtype agrarea so_eur unit geo
Time Period: A
Reshape variable: farmtype
I'm reshaping long...
I'm reshaping wide...
I'm destringing variables...
```

```
Contains data from data/out_data/EF_M_FARMANG.dta
Observations: 1,217,592
Variables: 32 28 Oct 2022 19:17
```

Variable name	Storage type	Display format	Value label	Variable label
freq	str1	%9s		Time frequency
sex	str1	%9s		Sex
age	str6	%9s		Age class
crops	str6	%9s		Crops
agrarea	str8	%9s		Agricultural area
so_eur	str9	%9s		Standard output in Euros
unit	str3	%9s		Unit of measure
geo	str2	%9s		Geopolitical entity (reporting)
date	int	%ty		Time
FT15_S0	float	%9.0g		Specialist cereals, oilseed and protein crops
FT16_S0	float	%9.0g		General field cropping
FT21_S0	float	%9.0g		Specialist horticulture indoor
FT22_S0	float	%9.0g		Specialist horticulture outdoor
FT23_S0	float	%9.0g		Other horticulture
FT35_S0	float	%9.0g		Specialist vineyards

FT36_S0	float	%9.0g	Specialist fruit and citrus fruit
FT37_S0	float	%9.0g	Specialist olives
FT38_S0	float	%9.0g	Various permanent crops combined
FT45_S0	float	%9.0g	Specialist dairying
FT46_S0	float	%9.0g	Specialist cattle-rearing and fattening
FT47_S0	float	%9.0g	Cattle-dairying, rearing and fattening combined
FT48_S0	float	%9.0g	Sheep, goats and other grazing livestock
FT51_S0	float	%9.0g	Specialist pigs
FT52_S0	float	%9.0g	Specialist poultry
FT53_S0	float	%9.0g	Various granivores combined
FT61_S0	float	%9.0g	Mixed cropping
FT73_S0	float	%9.0g	Mixed livestock, mainly grazing livestock
FT74_S0	float	%9.0g	Mixed livestock, mainly granivores
FT83_S0	float	%9.0g	Field crops-grazing livestock combined
FT84_S0	float	%9.0g	Various crops and livestock combined
FT90_S0	float	%9.0g	Non-classified farms
TOTAL	float	%9.0g	Total

Sorted by:

Variable	Obs	Mean	Std. dev.	Min	Max
freq	0				
sex	0				
age	0				
crops	0				
agrarea	0				
so_eur	0				
unit	0				
geo	0				
date	1,217,592	2017.924	1.998569	2016	2020
FT15_S0	487,179	1.08e+07	2.26e+08	0	3.93e+10
FT16_S0	679,352	7310360	1.71e+08	0	3.44e+10
FT21_S0	269,045	7892867	1.37e+08	0	1.81e+10
FT22_S0	309,750	3017124	5.12e+07	0	7.91e+09
FT23_S0	280,442	4212683	6.65e+07	-140	9.89e+09
FT35_S0	305,338	1.10e+07	1.80e+08	0	2.40e+10
FT36_S0	409,589	4765704	8.18e+07	0	1.55e+10
FT37_S0	151,701	8054803	1.10e+08	0	1.37e+10
FT38_S0	285,579	2529938	3.84e+07	0	6.40e+09
FT45_S0	512,168	1.77e+07	3.64e+08	0	5.96e+10
FT46_S0	604,660	4309134	9.08e+07	0	1.80e+10
FT47_S0	345,511	2551722	4.37e+07	0	6.29e+09
FT48_S0	617,349	3168642	6.38e+07	0	1.40e+10
FT51_S0	348,470	1.36e+07	2.63e+08	0	3.96e+10
FT52_S0	328,008	9402786	1.75e+08	0	2.67e+10
FT53_S0	173,455	1307532	1.72e+07	0	1.81e+09
FT61_S0	468,478	3881369	6.85e+07	0	1.26e+10
FT73_S0	327,998	2317257	3.64e+07	0	4.98e+09
FT74_S0	236,970	2823967	4.36e+07	0	4.49e+09
FT83_S0	425,928	5524535	1.15e+08	0	1.68e+10
FT84_S0	508,571	3828495	7.09e+07	0	1.23e+10
FT90_S0	111,592	829.4179	12072.33	0	1321180
TOTAL	1,148,085	4.54e+07	1.29e+09	-100	3.69e+11

Elapsed time was 49 minutes, 20.66 seconds.

Il tempo necessario per l'importazione si riduce a circa 49 minuti e dimensione finale è di circa 158MB... not so bad!

Vediamo adesso un confronto con il comando `sdmxuse`. Dal dataset `SBS_R_NUTS06_R2` (notare che in questo caso il nome del database va indicato in minuscolo) voglio scaricare tutte le frequenze disponibili (ho lasciato vuoto la prima dimension), i NACE con codice C e G, gli indicatori V11210, V13320 e tutte le ubicazioni geografiche (ho lasciato vuoto l'ultima dimension)

```
. sdmxuse data ESTAT, dataset(sbs_r_nuts06_r2) cclear dimensions(.C+G.V11210+V13320.)
2128 serie(s) imported
```

```
. summ
```

Variable	Obs	Mean	Std. dev.	Min	Max
nace_r2	0				
indic_sb	0				
geo	0				
freq	0				
time	0				
value	20,224	20394.3	62357.07	0	1433467

Per ottenere lo stesso risultato con eudbimport devo impostare così il comando:

```
. eudbimport SBS_R_NUTS06_R2, download select(keep if inlist(nace_r2,"C","G") ///
& inlist(indic_sb,"V11210","V13320"))
I'm downloading the file...
I'm importing data...
```

```
Database: SBS_R_NUTS06_R2
Selection's variables: freq nace_r2 indic_sb geo
Time Period: A
Reshape variable: indic_sb
I'm reshaping long...
I'm reshaping wide...
I'm destringing variables...
```

```
. summ
```

Variable	Obs	Mean	Std. dev.	Min	Max
freq	0				
nace_r2	0				
geo	0				
date	10,451	2013.915	3.66922	2008	2020
V11210	10,043	35508.99	84476.28	0	1433467
V13320	10,181	5484.491	15403.65	0	355472.1

8. Stats

Il comando è stato testato su tutti i databases presenti sul sito di EUROSTAT e tramite l'opzione debug sono state raccolte delle informazioni sui tempi di esecuzione. Vengono misurate le durate delle seguenti fasi ⁷:

1. Download: tempo necessario per scaricare il file .tsv dal sito⁸
2. Import: tempo necessario per importare il file .tsv in Stata tramite il comando `import delimited`
3. Reshape Long: tempo necessario per il reshape in formato long dei dati tramite il comando `reshape long`
4. Reshape Wide: tempo necessario per il reshape in formato wide dei dati tramite il comando `reshape wide`
5. Destring: tempo necessario per il destring delle variabili in formato numerico
6. Spurio: tempo impiegato per compiere diverse operazioni tra le diverse fasi (tempo totale - durata delle fasi precedenti)
7. Totale: tempo totale per l'esecuzione del comando (è la somma dei tempi delle fasi precedenti)

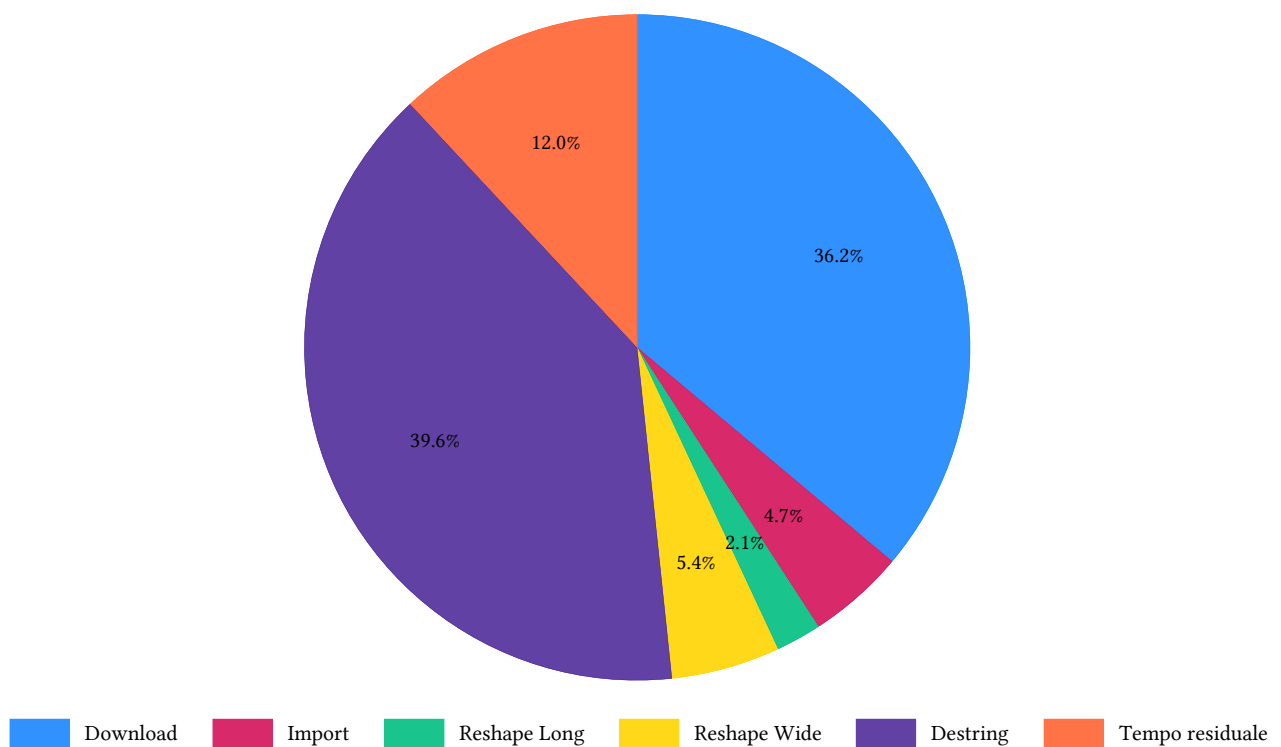
La fase più time consuming è quella del destring delle variabili generate dal reshape, mentre il download e l'import sono essenzialmente legate alla dimensione totale del database. Il pacchetto `gtools` consente di ridurre notevolmente i tempi di esecuzione dei due reshape.

7. L'opzione `info` (se specificata) non viene conteggiata nel tempo totale

8. Senza opzione `compressed`

Fasi	Media	Dev. Std.	Minimo	Massimo
Download	10.032	66.114	0.107	2611.736
Import	1.291	17.454	0.009	1074.660
Reshape Long	0.593	2.645	0.012	77.873
Reshape Wide	1.498	6.915	0.017	194.115
Destring	10.965	58.703	0.000	2118.190
Tempo residuale	3.315	17.409	0.062	601.336
Totale	27.693	127.847	0.267	3307.520

Dati riferiti a circa 7100 databases, dati in secondi



9. To Do

- English version

Riferimenti bibliografici

- [1] Ben Jann, 2007. "FRE: Stata module to display one-way frequency table", Statistical Software Components S456835, Boston College Department of Economics, revised 31 May 2021. <https://ideas.repec.org/c/boc/bocode/s456835.html>
- [2] Mauricio Caceres Bravo, 2018. "GTOOLS: Stata module to provide a fast implementation of common group commands", Statistical Software Components S458514, Boston College Department of Economics, revised 03 Apr 2019. <https://ideas.repec.org/c/boc/bocode/s458514.html>
- [3] Nicholas J. Cox, 2015. "MISSINGS: Stata module to manage missing values", Statistical Software Components S458071, Boston College Department of Economics, revised 11 May 2017. <https://ideas.repec.org/c/boc/bocode/s458071.html>
- [4] Sébastien Fontenay, 2016. "SDMXUSE: Stata module to import data from statistical agencies using the SDMX standard", Statistical Software Components S458231, Boston College Department of Economics, revised 30 Sep 2018.
- [5] Asjad Naqvi, 2020. <https://medium.com/the-stata-guide/automating-eurostat-in-stata-part-1-a047941b2b4f>
- [6] <https://ec.europa.eu/eurostat/data/database>
- [7] https://ec.europa.eu/eurostat/databrowser/explore/all/all_themes?lang=en&display=list&sort=category