

Guida al comando eudbimport

Nicola Tommasi*

13 SETTEMBRE 2022

*Centro Interdipartimentale di Documentazione Economica (C.I.D.E.)
email: nicola.tommasi@univr.it

Indice

1	Fonte dei dati	1
2	Struttura dei databases	2
3	Scaricare tutti i database in local	3
3.1	Come usare 7zip in Stata	3
4	Costruzione dei do-file per gli items	4
5	Il file eudbimport_labvar.do	6
6	Regole per rinominare le variabili	6
6.1	Le frequenze presenti nei db	7
7	Sintassi di eudbimport	8
7.1	Esempi	9

1. Fonte dei dati

Tutti i database compatibili con il comando **eudbimport** sono reperibili a [questo indirizzo](#) che però a breve sarà sostituito da [questo](#).

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. 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); il download è piuttosto lungo, 7-8 ore, e scarica circa 50GB di dati (QUESTO DATO è DA VERIFICARE).

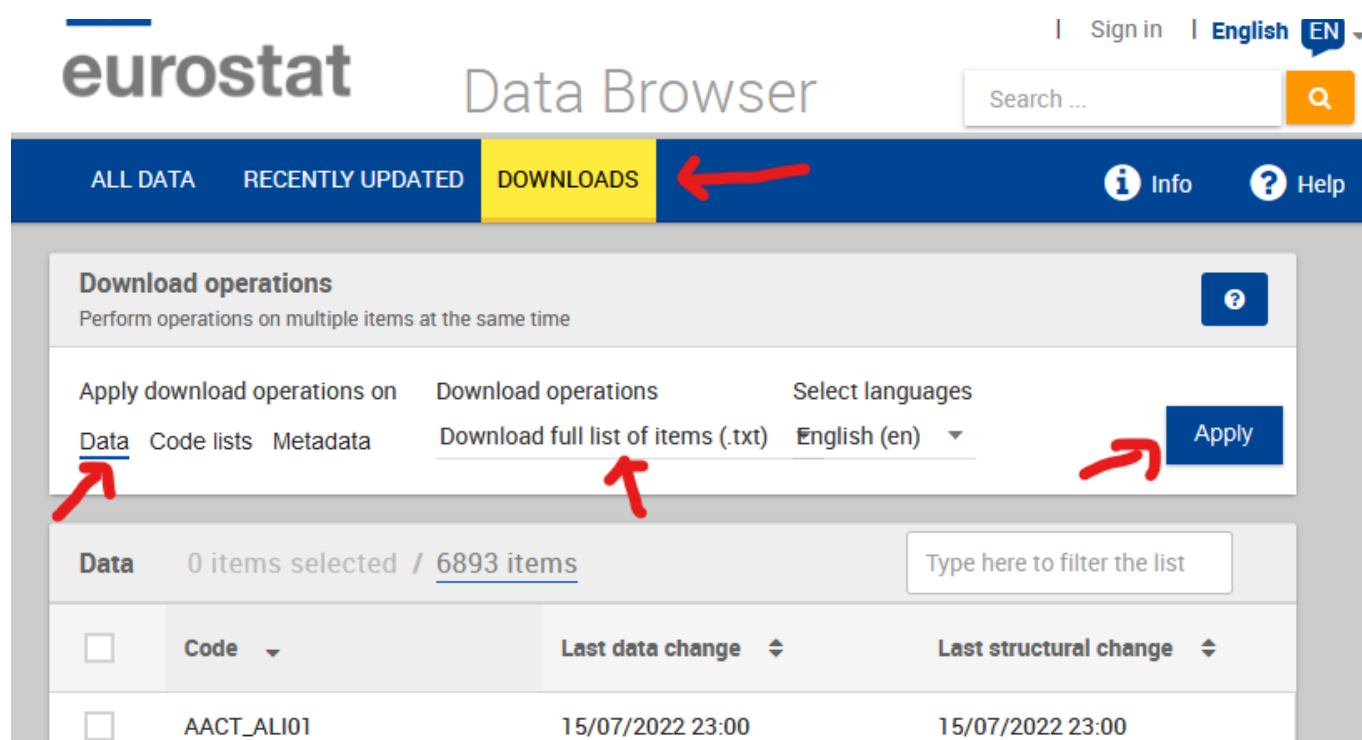


Figura 2 – Come reperire la lista dei database

In seguito verrà mostrato un do-file che a partire da questo file esegue il download completo dei database in esso elencati.

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.

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

2. Struttura dei databases

Tutti i database presenti sul sito hanno una struttura di base costante. Vediamone un esempio e famigliarizziamo 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 splitvars. In questo caso sono Geopolitical entity, Time,

Immigration by age group, sex and country of birth

online data code: MIGR_IMM3CTB last update: 30/03/2022 11:00 view: DEFAULT

Source of data: Eurostat

Selection ☐ Format ☐ Download

Row: Geopolitical entity (reporting) [34/34] 34 values displayed

Column: Time [10/13] 10 values displayed

Page: Country/region of birth [5/300] Total

Time frequency: Annual Unit of measure: Number

Age class: Total [1/27]

Age definition: Age reached during the year [1/2]

Sex: Total [1/3]

Immigration by age group, sex and country of birth (online data code: MIGR_IMM3CTB) Settings: Default presentation

Table Line Bar Map

	TIME	2016	2017	2018	2019	2020
GEO						
Belgium		123 782	126 783	137 860	150 006	118 683
Bulgaria		21 241 (p)	25 597 (p)	29 559 (p)	37 929 (p)	37 364 (p)
Czechia		64 083	51 847	65 910	105 888	63 095
Denmark		74 383	68 579	64 669	61 384	57 230

Figura 3 – La struttura dei database

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 splitvars:

- 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 splitvars 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 splitvars nelle nuove variabili di colonna attraverso un reshape. Questa variabile è definita reshapevar e le sue specificazioni diventeranno nuove variabili.

3. Scaricare tutti i database in local

Adesso mostro come scaricare in locale tutti i database del sito EUROSTAT. Questa operazione non è necessaria perchè come sarà mostrato in seguito 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.

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

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

Il codice è abbastanza semplice. Con `import delimited` importiamo in Stata il file txt, poi inizia un ciclo prendere certe 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. Per adesso ho implementato solo 7zip come metodo di compressione ma a breve aggiungerò anche il classico zip.

3.1. Come usare 7zip 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.

1. 6893 il 10 settembre 2022

Se il vostro sistema operativo è Windows:

Installare il programma 7zip

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 (si chiama così ??). Poi non serve fare più nulla, in linux non serve intervenire nel file profile.do

In MacOS:

boooo!!!

4. Costruzione dei do-file per gli items

Nel capitolo 1 di pagina 1 è stato descritto come scaricare dal sito di EUROSTAT i file relativi ai Codes list. Ora vediamo come sono strutturati questi files. Sono dei files di testo con estensione .tsv e il nome è ESTAT_<CODELIST>_en.tsv² 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_LVLXCROS	Collisions (excluding level-crossing accidents)
DERAIL	Deraillments of trains
DGD	Accidents involving transport of dangerous goods
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
LVLXCROS	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:

2. _en finale è per la versione inglese del file, le altre possibili sono fr e de.

```

clear all
set more off

capture mkdir items
capture mkdir dic
**cd items

local itemslist : dir "items" files "*.tsv", respectcase
local nitems : word count `itemslist'
di `nitems'

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

    import delimited "items/`f'", clear encoding(UTF-8) stringcols(_all) delimiter(tab)
    **devono essere lette solo 2 variabili
    qui describe
    assert r(k)==2
    duplicates report v1
    assert r(unique_value)==r(N)
    *! 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 "`f'" == "ESTAT_NACE_R2_en.tsv" {
        replace v1="B06__B09" if v1=="B06-B09"
        replace v1="O__U" if v1=="O-U"
    }
    }

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

    if "`f'" == "ESTAT_UNIT_en.tsv" replace v1="MIO__EUR__NSA" if v1=="MIO-EUR-NSA"

    replace v1 = ustrtoname(v1,1)

    duplicates report v1
    assert r(unique_value)==r(N)

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

    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 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 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 `acomsiz`:

```
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 db. Non è strettamente necessario, il label può essere fatto anche manualmente dopo aver importato il db. 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 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

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

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

- 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

Questa è da verificare:

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

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 seguente 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:

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.

7. Sintassi di eudbimport

Per installare il comando dalla command bar di Stata digitate:

```
net from https://raw.githubusercontent.com/NicolaTommasi8/eudbimport/master/
```

Otterrete questo output

```
-----
https://raw.githubusercontent.com/NicolaTommasi8/eudbimport/master/
(no title)
-----
```

```
PACKAGES you could -net describe-:
      eudbimport      Package to import EUROSTAT dababases
-----
```

quindi cliccate su eudbimport e alla pagina successiva su (click here to install).

(ancora da verificare che funzioni, ma dovrebbe essere OK)

```
eudbimport DBNAME, reshapevar(varname) [rawdata(string) outdata(string)
      download select(string) timeselect(string) nosave
      compress(string) decompress(string) /*undocumented*/
      nodestring /*undocumented*/
      debug /*undocumented*/ ]
```

dove:

`DBNAME` è il nome del database come riportato sul sito EUROSTAT e va indicato in maiuscolo.

`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.

`eudbimport` esegue il `destring` delle variabili originate dalla variabile indicata in `reshapevar()`. In alcuni casi il `destring` non funziona perchè 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".

`eudbimport` necessita dei comandi del pacchetto **gtools** (`greshape long`, `greshape wide` e `glevelsof`) e del comando **missings**.

7.1. Esempi

Partiamo dall'importazione del database NAMA_10_GDP, GDP and main components (output, expenditure and income) che potete visualizza [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
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
B1GQ	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 su `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

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 data 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]                          Units: 1  
Or equivalently: [1993,2021]                  Units: years  
Unique values: 29                             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 limiti 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 `timeselct()`. 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) timeselct(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.

splitvars			date	reshapevar			
splitvar 1	splitvar 2	splitvar3	timevar	item 1	item 2	item 3	item n