C.I.D.E.

Centro Interdipartimentale di Documentazione Economica

Università degli Studi di Verona

Manuale di Stata

...ovvero una informale introduzione a Stata
con l'aggiunta di casi applicati

Autore:
Nicola Tommasi

3 dicembre 2018
rev. 0.20

Info

Sito web: http://www.stata.com/
Forum: https://www.statalist.org/forums/

dott. Nicola Tommasi
e-mail: nicola.tommasi@gmail.com - nicola.tommasi@univr.it
Tel.: 045 802 80 48 (p.s.: la mail è lo strumento migliore e con probabilità di successo più elevata per contattarmi).
sito: https://sites.google.com/site/nicolatommasi/

La versione più aggiornata del manuale la potete trovare qui: http://cide.univr.it/statamanual.pdf
... altrimenti cercate nella sezione download di **http://cide.univr.it**
... altrimenti cercate qui, **nel mio sito**, dove troverete anche altre cosette...
... e anche qui, su GitHub (https://github.com/NicolaTommasi8).

Figura 1: II Incontro degli Utenti di Stata, Milano, 10-11 ottobre 2005

Indice

Info iii

Indice v

Ringraziamenti vii

Lista delle modifiche ix

Introduzione xi

I Manuale 1

1 Descrizione di Stata 3
1.1 La disposizione delle finestre . . . 3
1.2 Limiti di Stata . . . 5

2 Convenzioni Tipografiche 9

3 La Filosofia del Programma 11
3.1 Schema di funzionamento . . . 12

4 Organizzare il Lavoro 15
4.1 Organizzazione per cartelle di lavoro . . . 15
4.2 Interazione diretta VS files .do . . . 19
4.3 Registrazione dell'output . . . 20
4.4 Aggiornare il programma . . . 21
4.5 Aggiungere comandi . . . 21
4.6 Fare ricerche . . . 23
4.7 Cura dei dati . . . 24
4.8 Intestazione file .do . . . 26

5 Alcuni Concetti di Base 29
5.1 L'input dei dati . . . 29

5.1.1 Caricamento dei dati in formato proprietario . . . 29
5.1.2 Caricamento dei dati in formato testo . . . 29
5.1.3 Caricamento dei dati in altri formati proprietari (StatTransfer) . . 30
5.2 Regole per denominare le variabili . . . 30
5.3 Il qualificatore *in* . . . 31
5.4 Il qualificatore *if* . . . 32
5.5 Operatori di relazione . . . 33
5.6 Operatori logici . . . 34
5.7 Caratteri jolly e sequenze . . . 34
5.8 L'espressione `by` . . . 35
5.9 Dati missing . . . 36

6 Il Caricamento dei Dati **39**
6.1 Dati in formato proprietario (.dta) . . . 39
6.2 Dati in formato testo . . . 43
6.2.1 Formato testo delimitato . . . 43
6.2.2 Formato testo non delimitato . . . 44
6.3 Altri tipi di formati . . . 47
6.4 Esportazione dei dati . . . 48
6.5 Cambiare temporaneamente dataset . . . 49

7 Gestione delle Variabili **53**
7.1 Descrizione di variabili e di valori . . . 53
7.2 Controllo delle variabili chiave . . . 62
7.3 Rinominare variabili . . . 63
7.4 Ordinare variabili . . . 65
7.5 Prendere o scartare osservazioni o variabili . . . 66
7.6 Gestire il formato delle variabili . . . 67

8 Creare Variabili **71**
8.1 Il comando `generate` . . . 71
8.1.1 Funzioni matematiche . . . 71
8.1.2 Funzioni di distribuzione di probabilità e funzioni di densità . . . 73
8.1.3 Funzioni di generazione di numeri random . . . 75
8.1.4 Funzioni stringa . . . 75
8.1.5 Funzioni di programmazione . . . 78
8.1.6 Funzioni data . . . 79
8.1.7 Funzioni per serie temporali . . . 80
8.1.8 Funzioni matriciali . . . 80
8.2 Lavorare con osservazioni indicizzate . . . 83
8.3 Estensione del comando `generate` . . . 85
8.4 Sostituire valori in una variabile . . . 88
8.5 Creare variabili dummy . . . 92

9 Analisi Quantitativa **95**
9.1 `summarize` e `tabulate` . . . 95
9.1.1 Qualcosa di più avanzato . . . 106
9.2 Analisi della correlazione . . . 109
9.3 Analisi outliers . . . 111

10 Trasformare Dataset **117**
10.1 Aggiungere osservazioni . . . 117
10.2 Aggiungere variabili . . . 118
10.3 Collassare un dataset . . . 126
10.4 `reshape` di un dataset . . . 127
10.5 Contrarre un dataset . . . 131

11 Lavorare con Date e Orari **135**
11.1 La teoria . . . 135
11.2 Visualizzazione delle date e delle ore . . . 138
11.3 Ricavare date da variabili stringa . . . 140
11.4 Visualizzazione delle ore . . . 141
11.5 Operazioni con date e ore . . . 141

12 Macros e Cicli **143**
12.1 Macros . . . 143
12.2 I cicli . . . 145

13 Catturare Informazioni dagli Output **151**

14 Esportazione dell'output **155**
14.1 arraytex e arrayxls . . . 156
14.2 fretex e frexls . . . 156
14.3 mrtabtex e mrtabxls . . . 156
14.4 tabletex e tablexls . . . 156
14.5 tabstattex e tabstatxls . . . 156
14.6 tabtex e tabxls . . . 156

15 Mappe **157**

II Casi Applicati **165**

16 Dataset di Grandi Dimensioni **167**

17 Da Stringa a Numerica **171**
17.1 Fondere variabili stringa con numeriche . . . 171
17.2 Da stringa a numerica categorica . . . 174

18 Liste di Files e Directory **175**

19 Previsioni Multiple **181**
19.1 Introduzione . . . 181
19.2 La stima del modello . . . 181

20 `reshape` su molte Variabili **193**

21 Importare dati .xml dal GME **197**
21.1 Accesso ai files .zip . . . 197
21.2 Leggere dati in formato .xml . . . 200

III Appendici **205**

A `spmap`: Visualization of spatial data **207**
A.1 Syntax . . . 207
A.1.1 basemap_options . . . 207
A.1.2 polygon_suboptions . . . 208
A.1.3 line_suboptions . . . 209
A.1.4 point_suboptions . . . 209
A.1.5 diagram_suboptions . . . 210
A.1.6 arrow_suboptions . . . 211
A.1.7 label_suboptions . . . 212
A.1.8 scalebar_suboptions . . . 213
A.1.9 graph_options . . . 213
A.2 description . . . 213
A.3 Spatial data format . . . 214
A.4 Color lists . . . 219
A.5 Choropleth maps . . . 221
A.6 Options for drawing the base map . . . 222
A.7 Option polygon() suboptions . . . 225
A.8 Option line() suboptions . . . 227
A.9 Option point() suboptions . . . 228
A.10 Option diagram() suboptions . . . 230
A.11 Option arrow() suboptions . . . 232
A.12 Option label() suboptions . . . 234
A.13 Option scalebar() suboptions . . . 235
A.14 Graph options . . . 236
A.15 Acknowledgments . . . 263

B Lista pacchetti aggiuntivi **267**

IV Indici **303**

Indice Analitico **305**

Elenco delle figure **309**

Elenco delle tabelle **311**

Ringraziamenti

Molto del materiale utilizzato in questo documento proviene da esperienze personali. Prima e poi nel corso della stesura alcune persone mi hanno aiutato attraverso suggerimenti, insegnamenti e correzioni; altre hanno contribuito in altre forme. Vorrei ringraziare sinceramente ciascuno di loro. Naturalmente tutti gli errori che troverete sono miei.

Li elenco in ordine rigorosamente sparso

> Fede che mi ha fatto scoprire Stata quando ancora non sapevo accendere un PC
> Raffa con cui gli scambi di dritte hanno contribuito ad ampliare le mie conoscenze
> Piera che mi dato i primissimi rudimenti

Lista delle modifiche

rev. 0.01

- Prima stesura

rev. 0.02

- Aggiunti esempi di output per illustrare meglio i comandi
- Aggiornamenti dei nuovi comandi installati (adoupdate)
- Controllo delle variabili chiave (duplicates report)

rev. 0.03

- Aggiunti esempi di output per illustrare meglio i comandi
- Conversione del testo in LaTeX (così lo imparo)
- Creata la sezione con i casi applicati

rev. 0.04

- Indice analitico
- Mappe (comando `spmap`, ex `tmap`
- Ulteriori esempi

rev. 0.06

- Correzioni varie
- Date e ore
- Ulteriori casi applicati

rev. 0.08

- Correzioni varie
- completamento Date e ore

- caso GME

rev. 0.09

- rivista la parte del `reshape`
- rivista la parte del `merge` alla luce della nuova versione del comando
- rivista la parte della organizzazione del lavoro
- link ad altri miei materiali
- correzioni varie

rev. 0.10

- correzioni varie (thanks Fabio Ciaponi)

Introduzione

Questo è un tentativo di produrre un manuale che integri le mie esperienze nell'uso di Stata. È un work in progress in cui di volta in volta aggiungo nuovi capitoli, integrazioni o riscrivo delle parti. In un certo senso è una collezione delle mie esperienze di Stata, organizzate per assomigliare ad un manuale, con tutti i pro e i contro di una tale genesi. Non è completo come vorrei ma il tempo è un fattore limitante. Se qualcuno vuole aggiungere capitoli o pezzi non ha che da contattarmi, sicuramente troveremo il modo di inglobare i contributi che verranno proposti. Naturalmente siete pregati di segnalarmi tutti gli errori che troverete (e ce ne saranno).

Parte I

Manuale

Capitolo 1

Descrizione di Stata

Software statistico per la gestione, l'analisi e la rappresentazione grafica di dati

Piattaforme supportate

- Windows (versioni 32 e 64 bit)
- Linux (versioni 32 e 64 bit)
- Macintosh
- Unix, AIX, Solaris Sparc

Versioni (in senso crescente di capacità e potenza)

- Small Stata
- Stata/IC
- Stata/SE
- Stata/MP

La versione SE è adatta alla gestione di database di grandi dimensioni. La versione MP è ottimizzata per sfruttare le architetture multiprocessore attraverso l'esecuzione in parallelo dei comandi di elaborazione (parallelizzazione del codice). Per farsi un'idea si veda l'ottimo documento reperibile qui:
Stata/MP Performance Report
(http://www.stata.com/statamp/report.pdf)
Questa versione, magari in abbinamento con sistemi operativi a 64bit, è particolarmente indicata per situazioni in cui si devono elaborare grandi quantità di dati (dataset di svariati GB) in tempi che non siano geologici.

1.1 La disposizione delle finestre

Stata si compone di diverse finestre che si possono spostare ed ancorare a proprio piacimento (vedi Figura 1.1). In particolare:

1. **Stata Results**: finestra in cui Stata presenta l'output dei comandi impartiti

2. **Review**: registra lo storico dei comandi impartiti dalla Stata Command. Cliccando con il mouse su uno di essi, questo viene rinviato alla Stata Command
3. **Variables**: quando un dataset è caricato qui c'è l'elenco delle variabili che lo compongono
4. **Stata Command**: finestra in cui si scrivono i comandi che Stata deve eseguire

A partire dalla versione 8 è possibile eseguire i comandi anche tramite la barra delle funzioni dove sotto 'Data', 'Graphics' e 'Statistics' sono raggruppati i comandi maggiormente usati. Dato che ho imparato ad usare Stata alla vecchia maniera (ovvero da riga di comando) non tratterò questa possibilità. Però risulta molto utile quando si devono fare i grafici; prima costruisco il grafico tramite 'Graphics' e poi copio il comando generato nel file .do.

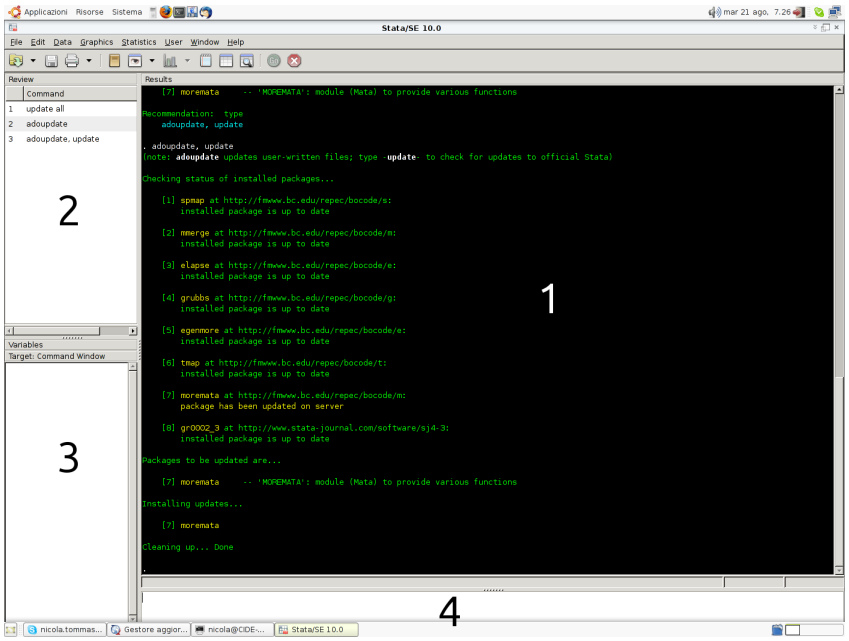


Figura 1.1: Le finestre di Stata

Come già accennato i riquadri che compongono la schermata del programma si possono spostare. Quella presentata in figura 1.1 è la disposizione che personalmente ritengo più efficiente . . . ma naturalmente dipende dai gusti.
Per salvare la disposizione: 'Prefs -> Save Windowing Preferences'

Trucco: Il riquadro 'Variables' prevede 32 caratteri per il nome delle variabili. Se a causa di questo spazio riservato al nome delle variabili, il label non è visibile si può intervenire per restringerlo:

```
set varlabelpos #
```

con 8 <= # <= 32, dove #è il numero di caratteri riservati alla visualizzazione del nome delle variabili. Quelle con nome più lungo di #verranno abbreviate e comparirà il simbolo ∼ nel nome a segnalare che quello visualizzato non è il vero nome ma la sua abbreviazione.

1.2 Limiti di Stata

Con il comando `chelp limits` possiamo vedere le potenzialità e le limitazioni della versione di Stata che stiamo utilizzando:

```
. chelp limits

help limits
-------------------------------------------------------------------------------

Maximum size limits

                                                                  Stata/MP and
                                          Small       Stata/IC       Stata/SE
-------------------------------------------------------------------------------
   # of observations  (1)           about 1,000  2,147,483,647  2,147,483,647
   # of variables                        99          2,047         32,767
   width of a dataset                   200         24,564        393,192

   value of matsize                      40            800         11,000

   # characters in a command          8,697        165,216      1,081,527
   # options for a command              70             70             70

   # of elements in a numlist         1,600          1,600          1,600

   # of unique time-series operators in
       a command                       100            100            100
   # seasonal suboperators per time-series
       operator                          8              8              8

   # of dyadic operators in an expression    66            800            800
   # of numeric literals in an expression    50            300            300
   # of string literals in an expression    256            512            512
   length of string in string expression    244            244            244
   # of sum functions in an expression       5              5              5

   # of characters in a macro          8,681        165,200      1,081,511

   # of nested do-files                  64             64             64

   # of lines in a program            3,500          3,500          3,500
   # of characters in a program     135,600        135,600        135,600

   length of a variable name             32             32             32
   length of ado-command name            32             32             32
   length of a global macro name         32             32             32
   length of a local macro name          31             31             31

   length of a string variable          244            244            244

   adjust
       # of variables in by() option      7              7              7

   anova
```

```
    # of variables in one anova term       8             8             8
    # of terms in the repeated() option    4             4             4

char
    length of one characteristic       8,681        67,784        67,784

constraint
    # of constraints                  1,999         1,999         1,999

encode and decode
    # of unique values                 1000         65,536        65,536

_estimates hold
    # of stored estimation results      300           300           300

estimates store
    # of stored estimation results      300           300           300

grmeanby
    # of unique values in varlist      _N/2          _N/2          _N/2

graph twoway
    # of variables in a plot            100           100           100
    # of styles in an option´s stylelist  20            20            20

impute
    # of variables in varlist            31            31            31

infile
    record length without dictionary    none          none          none
    record length with a dictionary  524,275       524,275       524,275

infix
    record length with a dictionary  524,275       524,275       524,275

label
    length of dataset label              80            80            80
    length of variable label             80            80            80
    length of value label string     32,000        32,000        32,000
    length of name of value label        32            32            32
    # of codings within one
        value label                   1,000        65,536        65,536

label language
    # of different languages           100           100           100

manova
    # of variables in single manova term   8             8             8

matrix  (2)
    dimension of single matrix       40 x 40      800 x 800  11,000x11,000

maximize options
    iterate() maximum                16,000        16,000        16,000

mlogit
    # of outcomes                        20            50            50

net (also see usersite)
    # of description lines in .pkg file  100           100           100

nlogit and nlogittree
    # of levels in model                  8             8             8

notes
```

```
        length of one note                 8,681         67,784         67,784
        # of notes attached to _dta        9,999          9,999          9,999
        # of notes attached to each
            variable                       9,999          9,999          9,999

    numlist
        # of elements in the numeric list  1,600          1,600          1,600

    ologit and oprobit
        # of outcomes                        20             50             50

    reg3, sureg, and other system estimators
        # of equations                       40            800         11,000

    set adosize
        memory ado-files may consume        500K           500K           500k

    set scrollbufsize
        memory for Results window buffer    500K           500K           500k

    stcox
        # of variables in strata() option     5              5              5

    stcurve
        # of curves plotted on the same graph 10             10             10

    table and tabdisp
        # of by variables                     4              4              4
        # of margins, i.e., sum of rows,
          columns, supercolumns, and
          by groups                       3,000          3,000          3,000

    tabulate  (3)
        # of rows in one-way table          500          3,000         12,000
        # of rows & cols in two-way table 160x20         300x20       1,200x80

    tabulate, summarize (see tabsum)
        # of cells (rows X cols)            375            375            375

    xt estimation commands (e.g., xtgee,
        xtgls, xtpoisson, xtprobit, xtreg
        with mle option, and xtpcse when
        neither option hetonly nor option
        independent are specified)

        # of time periods within panel       40            800         11,000
        # of integration points accepted     40            800         11,000
          by intpoints(#)                   195            195            195
-------------------------------------------------------------------------------

Notes

    (1) 2,147,483,647 is a theoretical maximum; memory availability will
        certainly impose a smaller maximum.

    (2) In Mata, matrix is limited by the amount of memory on your computer.

    (3) For Stata/IC for the Macintosh, limits are 2,000 for the number of rows
        for a one-way table and 180 for number of rows for a two-way table.
```

Per sapere quale versione del programma stiamo usando:

```
. about

Stata/SE 10.0 for Windows
Born 25 Jul 2007
```

```
Copyright (C) 1985-2007

Total physical memory:     2096624 KB
Available physical memory: 1447220 KB

Single-user Stata for Windows perpetual license:
       Serial number:  81910515957
         Licensed to:  C.I.D.E.
                       Univeristy of Verona
```

Capitolo 2

Convenzioni Tipografiche

Per quanto possibile si cercherà di seguire le seguenti convenzioni tipografiche in accordo con i manuali stessi di Stata. Quando verranno spiegati i comandi, essi saranno rappresentati in questo modo:

`command` [*varlist*] [`=`*exp*] [*if*] [*in*] [*weight*] [`,` *options*]

dove tutto ciò che è racchiuso tra [ ] rappresenta parti opzionali del comando e quindi non indispensabili per la sua esecuzione.
Quindi ad esempio:

- se il comando presenta *varname* significa che il nome di una variabile è necessario
- se il comando presenta [*varname*] significa che il nome di una variabile non è necessario
- se il comando presenta *varlist* significa che una lista di variabili è necessaria
- se il comando presenta [*varlist*] significa che una lista di variabili non è necessaria

Tra parentesi { } saranno indicati liste di parametri tra i quali è indispensabile scegliere. Per esempio in

`tsset` [*panelvar*] *timevar* [`, format(%`*fmt*`) {daily | weekly | monthly | quarterly | halfyearly | yearly | generic }]

la parte `{daily | weekly` ... `generic }` indica una lista di opzioni tra le quali scegliere.

Taluni comandi, se non viene specificata una variabile o una lista di variabili, si applicano a tutte le variabili del dataset.

Spesso e volentieri le *options* sono molto numerose, per cui mi limiterò a trattare quelle che secondo me sono più importanti.

Porzioni di files .do o output di Stata saranno indicati con il seguente layout:

```
. use auto
(1978 Automobile Data)

. summ

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
        make |         0
       price |        74    6165.257    2949.496       3291      15906
         mpg |        74     21.2973    5.785503         12         41
       rep78 |        69    3.405797    .9899323          1          5
    headroom |        74    2.993243    .8459948        1.5          5
-------------+--------------------------------------------------------
       trunk |        74    13.75676    4.277404          5         23
      weight |        74    3019.459    777.1936       1760       4840
      length |        74    187.9324    22.26634        142        233
        turn |        74    39.64865    4.399354         31         51
displacement |        74    197.2973    91.83722         79        425
-------------+--------------------------------------------------------
  gear_ratio |        74    3.014865    .4562871       2.19       3.89
     foreign |        74    .2972973    .4601885          0          1
```

Capitolo 3

La Filosofia del Programma

Stata è progettato per gestire efficacemente grandi quantità di dati, perciò tiene tutti i dati nella memoria RAM (vedi opzione `set mem`)

Stata considera il trattamento dei dati come un esperimento scientifico, perciò assicura:

a. la riproducibilità tramite l'uso dei files .do
b. la misurabilità tramite l'uso dei files .log o .smcl

Stata si compone di una serie di comandi che sono:

- **compilati** nell'eseguibile del programma
- presenti in forma di file di testo con estensione **.ado**
- **scritti da terzi** con la possibilità di renderli disponibili all'interno del programma
- **definiti dall'utente** e inseriti direttamente all'interno di files .do

Per vedere dove sono salvati i comandi scritti nei files .ado basta dare il comando.

```
. sysdir
   STATA:  C:\eureka\Stata10\

 UPDATES:  C:\eureka\Stata10\ado\updates\
    BASE:  C:\eureka\Stata10\ado\base\
    SITE:  C:\eureka\Stata10\ado\site\
    PLUS:  c:\ado\stbplus\
PERSONAL:  c:\ado\personal\
OLDPLACE:  c:\ado\
```

I comandi scritti da terzi solitamente si installano nella directory indicata in PLUS

Stata si usa essenzialmente da riga di comando

Gli input e gli output vengono dati in forma testuale

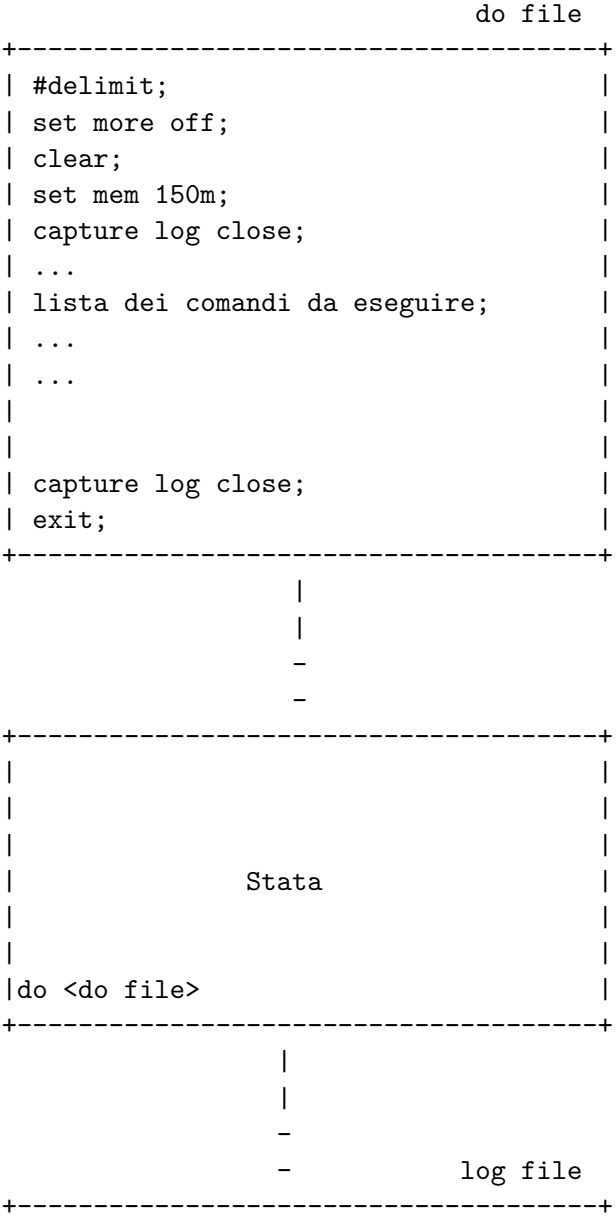
Di seguito si farà rifermento a variabili e osservazioni e in particolare

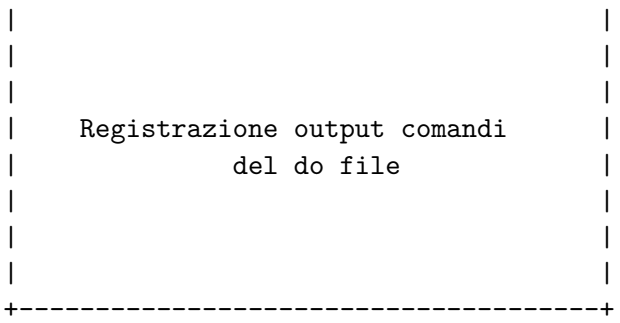
ciò che in Excel viene chiamato -colonna corrisponde a variabile in Stata
-riga corrisponde a osservazione in Stata

ciò che in informatica viene chiamato -campo corrisponde a variabile in Stata
-record corrisponde a osservazione in Stata

3.1 Schema di funzionamento

Questo è lo schema di funzionamento del programma. Capitelo bene e sarete più efficienti e produttivi nel vostro lavoro.





Il `do file` è un semplice file di testo nel quale viene scritta la sequenza dei comandi che devono essere eseguiti dal programma. Questo file viene passato a Stata per l'esecuzione tramite il comando `do <do_file>` da impartire dalla finestra Command. Se ci sono degli errori l'esecuzione si blocca in corrispondenza dell'errore stesso, altrimenti Stata esegue il do file e registra gli output dei comandi nel log file.

Capitolo 4

Organizzare il Lavoro

Dato che il metodo migliore di passare i comandi a Stata è la riga di comando, conviene dotarsi di un buon editor di testo. Quello integrato nel programma non è sufficientemente potente (si possono creare al massimo file di 32k), per cui consiglio di dotarsi uno dei seguenti editor gratuiti:
Notepad++ -> http://notepad-plus.sourceforge.net/it/site.htm
NoteTab Light -> http://www.notetab.com/
PSPad -> http://www.pspad.com/
RJ TextEd -> http://www.rj-texted.se
Tra i quatto indicati io preferisco l'ultimo. Sul sito spiegano anche come integrare il controllo e l'evidenziazione della sintassi dei comandi di Stata.
Utilizzando editor esterni si perde la possibilità di far girare porzioni di codice; c'è però un tentativo di integrare gli editor esterni; vedi a tal proposito:
http://fmwww.bc.edu/repec/bocode/t/textEditors.html

4.1 Organizzazione per cartelle di lavoro

La maniera più semplice ed efficiente di usare Stata è quella di organizzare il proprio lavoro in directory e poi far lavorare il programma sempre all'interno di questa directory. Se si usano i percorsi relativi la posizione di tale directory di lavoro sarà ininfluente e sarà possibile far girare i propri programmi anche su altri computer senza dover cambiare i percorsi.
In basso a sinistra, Stata mostra la directory dove attualmente sta' puntando. In alternativa è possibile visualizzarla tramite il comando:

```
pwd
```

in questo esempio Stata punta alla cartella C:\projects\CorsoStata\esempi e se impartite il comando di esecuzione di un file .do o di caricamento di un dataset senza specificare il percorso, questo verrà ricercato in questa cartella:

```
. pwd
C:\projects\CorsoStata\esempi
```

Il principio guida che ispira ogni elaborazione di dati dovrebbe essere la replicazione con il minimo sforzo di tutti i passaggi di preparazione dei dati, di analisi e di ottenimento dei risultati riportati nella stesura finale dei lavori. Servono quindi delle regole e degli schemi di lavoro standard che dovrebbero incentivare il senso di responsabilità e di rendicontazione del proprio lavoro. Da un po' di tempo cerco di adottare questa organizzazione per tutti i progetti che direttamente o indirettamente seguo[1]. Innanzitutto si crea una cartella con il nome del progetto e questa al suo interno conterrà almeno le seguenti cartelle e file:

1. una cartella data e la suo interno altre due cartelle
    - la cartella raw_data
    - la cartella out_dta
2. una cartella metadata
3. una cartella docs
4. una cartella graphs
5. un file di testo readme.txt
6. un file master.do
7. un file import.do
8. un file cleaning.do
9. un file results.do

Ed ecco qual è la funzione di ciascuna cartella e di ciascun file:

- cartella data: Contiene al suo interno la cartella raw_data in cui verranno messi tutti i file di partenza dei dati nel loro formato originale. Se questi file sono in un formato direttamente caricabile da Stata (.dta, .xls, .xlsx, .csv, tutti i dati in formato testo delimitato e non delimitato) si lasciano come sono altrimenti, oltre al file in formato originale, bisogna inserire anche la sua conversione in un formato importabile da Stata. La cartella out_dta invece ha la funzione di raccogliere tutti i file in formato .dta prodotti durante l'elaborazione dei dati. Conterrà anche tutti i file di dati che eventualmente dovessero essere prodotti in un formato diverso dal .dta.

[1] L'ispirazione viene da qui: Ball Richard and Medeiros Norm. 2011. Teaching Students to Document Their Empirical Research.

- cartella metadata: Contiene tutti i file che servono da documentazione per i dati contenuti nella cartella row_data. Solitamente sono file di documentazione forniti assieme ai dati come la definizione delle variabili, gli schemi di codifica dei valori o i metodi di campionamento e di raccolta dei dati. Se il file è unico deve avere lo stesso nome del relativo raw file, altrimenti si crea una cartella sempre con il nome del relativo raw file e al suo interno si inseriscono tutti i file di documentazione.
- cartella docs: Contiene tutta la documentazione relativa al progetto come paper di riferimento o istruzioni su come organizzare l'elaborazione dei dati
- cartella graphs: È una cartella opzionale, nel senso che se non vengono prodotti grafici che debbano essere salvati si può evitare di crearla. Se invece si producono grafici e questi devono essere salvati come file (.gph, .eps, .eps ...) questa è la cartella che li conterrà. Se i grafici sono molti, si possono organizzare in sottocartelle della principale.
- file readme.txt: Questo file contiene una panoramica di tutto il materiale che è stato assemblato nella cartella del progetto. In particolare dovrà contenere:
  - la lista di tutti i file contenuti nella cartella data con descrizione della loro posizione all'interno della cartella , del loro contenuto e del loro formato
  - la fonte dei dati ed eventualmente le istruzioni per riottenere gli stessi dati
  - la lista in ordine di esecuzione di tutti i .do files contenuti nella cartella e una breve descrizione della loro funzione
  - una referenza (e-mail, numero di telefono...) per contattare l'autore del lavoro in caso di necessità di ulteriori informazioni
- file master.do: Contiene la sequenza di lancio dei do-file. I tre do-file indicati sono caldamente consigliati ma non obbligatori, ovvero se le operazioni da compiere non sono molte il do-file potrebbe essere unico e quindi diventerebbe inutile anche il master.do. D'altra parte, se il progetto di ricerca fosse particolarmente complesso, è consigliabile aumentare il numero di do-file in aggiunta a quelli consigliati
- file import.do: Lo scopo di import.do è di importare i dati da ciascun file della cartella row_data in formato .dta e salvarli nella cartella out_dta. Se tutti i dati fossero già in formato Stata questo do-file non deve essere creato. Ciascun .dta file creato avrà lo stesso nome del corrispondente file in row_data. Per esempio se in row_data c'è il file unaid.txt, il corrispondente file in out_dta si chiamerà unaid.dta.
- file cleaning.do: Lo scopo di cleaning.do è di processare i dati al fine di arrivare al dataset finale da usare per le analisi. Quindi carica i dataset presenti in out_dta e applica procedure di pulizia dei dati, di merge e di append, quindi salva il file così prodotto sempre in out_dta con il prefisso clean_ o final_. É consigliabile in questa fase anche condurre delle analisi di esplorazione dei dati e di sperimentazione per verificare preventivamente le analisi che si intendono effettuare in results.do.

- file results.do: Contiene i comandi per generare per generare le variabili necessarie, per generare tabelle e figure, regressioni e tutti i risultati pubblicati nel report finale della ricerca.

Utili in questo contesto sono i comandi:

`mkdir` *directoryname*

per creare delle cartelle; in *directoryname* va indicato il percorso e il nome della directory da creare. Se in tale percorso ci fossero degli spazi bianchi, è necessario racchiudere il tutto tra virgolette.
Per esempio per creare la cartella `pippo` all'interno dell'attuale cartella di lavoro:

```
mkdir pippo
```

Per creare la cartella `pippo` nella cartella superiore all'attuale cartella di lavoro

```
mkdir ..\pippo
```

o

```
mkdir../pippo
```

Per create la cartella `pippo` nella cartella `pluto` contenuta nell'attuale cartella di lavoro

```
mkdir pluto/pippo
```

Per create la cartella `pippo` attraverso un percorso assoluto (sistema caldamente sconsigliato!!)

```
mkdir c:/projects/pippo
```

Per spostarsi tra le cartelle[2]

`cd [''][drive:][path]['']`

Per vedere la lista di file e cartelle relativi alla posizione corrente o per vedere il contenuto di altre cartelle, si usa il comando `dir`

```
dir pippo
dir ..\pippo
dir pluto\pippo
```

Per cancellare files

`erase ['']filename.ext ['']`

Attenzione che bisogna specificare anche l'estensione del file da cancellare

Nota1: Stata è in grado di eseguire anche comandi DOS, purchè siano preceduti dal simbolo '!'. Per esempio

```
!del *.txt
```

[2] '`cd ..`' serve per salire di un livello nella struttura delle directory, `cd ../..` di due e così via.

cancella tutti i files con estensione .txt nella cartella corrente.
Nota2: già detto, ma meglio ribadirlo; se nel percorso, il nome di un file o di una directory hanno degli spazi bianchi, l'intero percorso deve essere racchiuso tra virgolette.
Nota3: Stata è case sensitive per i comandi e per i nomi delle variabili (ma anche per gli scalar e per le macro), ma non per i nomi dei files e dei percorsi[3]

4.2 Interazione diretta VS files .do

Stata accetta i comandi in due modi:

a. Interazione diretta tramite l'inserimento dei comandi nella finestra 'Stata Command' o ricorrendo a 'Statistics' nella barra delle funzioni.
b. Attraverso dei files di semplice testo con estensione .do che contengono la serie di comandi da passare al programma per l'esecuzione.

Personalmente caldeggio l'adozione del secondo sistema perché consente di ottenere 2 importantissimi requisiti:

I. Si documentano tutti i passaggi che vengono fatti nella elaborazione dei dati
II. Si ha la riproducibilità dei risultati.

Per i files .do sono possibili due soluzioni per delimitare la fine di un comando. Di default Stata esegue un comando quando trova un invio a capo. Oppure si può scegliere il carattere `;` come delimitatore di fine comando. Data l'impostazione di default, per utilizzare il `;` bisogna dare il comando

```
#delimit ;
```

per ritornare alla situazione di default si usa il comando

```
#delimit cr
```

È inoltre possibile inserire commenti usando il carattere `*` se si vuole fare un commento su una sola riga, con `/*` all'inizio e `*/` alla fine per commenti disposti su più righe.
Se state lavorando con il delimitatore `cr` è possibile suddividere un comando su più righe usando `///`.
Se state lavorando con il delimitatore `;`, esso va messo anche alla fine di ciascuna riga commentata con `*`. Se invece state usando `/*` e `*/` va messo solo dopo `*/`.
Segue un esempio di quanto appena detto

```
/**** #delimit cr   ****/
gen int y = real(substr(date,1,2))
gen int m = real(substr(date,3,2))
gen int d = real(substr(date,5,2))
summ y m d
```

[3]Ciò vale per i SO Windows, non per i sistemi Unix/Linux. Per i Mac e per gli altri sistemi, semplicemente non lo so'.

```
recode y (90=1990) (91=1991) (92=1992) (93=1993) ///
(94=1994) (95=1995) (96=1996) (97=1997) (98=1998) ///
(99=1999) (00=2000) (01=2001) (02=2002) ///
(03=2003) (04=2004) /*serve per usare la funzione mdy*/
gen new_data = mdy(m,d,y)
format new_data %d

#delimit;
gen int y = real(substr(date,1,2));
gen int m = real(substr(date,3,2));
gen int d = real(substr(date,5,2));
summ y m d;

*Commento: le tre righe seguenti hanno l´invio a capo;
recode y (90=1990) (91=1991) (92=1992) (93=1993)
         (94=1994) (95=1995) (96=1996) (97=1997)
         (98=1998) (99=1999) (00=2000) (01=2001)
         (02=2002) (03=2003) (04=2004) /*serve per usare la funzione mdy*/;
gen new_data = mdy(m,d,y);

/****************************************
questo è un commento su + righe
bla bla bla
bla bla bla
****************************************/;

format new_data %d;
#delimit cr
```

È possibile dare l'invio a capo senza esecuzione del comando anche in modo cr se si ha l'accortezza di usare i caratteri `/*` alla fine della riga e `*/` all'inizio della successiva come mostrato nell'esempio seguente

```
use mydata, clear
regress lnwage educ complete age age2 /*
    */ exp exp2 tenure tenure2 /*
    */ reg1-reg3 female
predict e, resid
summarize e, detail
```

Attenzione: il comando #`delimit` non può essere usato nell'interazione diretta e quindi non si possono inserire comandi nella finestra 'Command' terminando il comando con `;`

4.3 Registrazione dell'output

Stata registra gli output dell'esecuzione dei comandi in due tipi di file:

- file .smcl (tipo di default nel programma)
- file .log

I files .smcl sono in formato proprietario di Stata e "abbelliscono" l'output con formattazioni di vario tipo (colori, grassetto, corsivo...), ma possono essere visualizzati solo con l'apposito editor integrato nel programma[4].

[4] Attraverso 'File -> Log -> View' o apposita icona.

I files .log sono dei semplici file di testo senza nessun tipo di formattazione e possono essere visualizzati con qualsiasi editor di testo.
Si può scegliere il tipo di log attraverso il comando

`set logtype text|smcl [, permanently]`

Si indica al programma di iniziare la registrazione tramite il comando

`log using` *filename* `[, append replace [text|smcl] name(`*logname*`)]`

La registrazione può essere sospesa tramite:

`log off [logname]`

ripresa con

`log on [logname]`

e infine chiusa con

`log close [logname]`

A partire dalla versione 10 è possibile aprire più files di log contemporaneamente.

4.4 Aggiornare il programma

Il corpo principale del programma di aggiorna tramite il comando

`update all`

```
. update all

-------------------------------------------------
> update ado
(contacting http://www.stata.com)
ado-files already up to date

-------------------------------------------------
> update executable
(contacting http://www.stata.com)
executable already up to date
```

in questo modo verranno prima aggiornati i files .ado di base del programma e poi l'eseguibile .exe. In quest'ultimo caso verrà richiesto il riavvio del programma.
Se non si possiede una connessione ad internet, sul sito di Stata è possibile scaricare gli archivi compressi degli aggiornamenti da installare all'indirizzo
http://www.stata.com/support/updates/
Sul sito vengono fornite tutte le istruzioni per portare a termine questa procedura

4.5 Aggiungere comandi

Come accennato in precedenza è possibile aggiungere nuovi comandi scritti da terze parti. Per fare ciò è necessario conoscere il nome del nuovo comando e dare il comando

`ssc install` *pkgname* `[, all replace]`

```
. ssc inst bitobit
checking bitobit consistency and verifying not already installed...
installing into c:\ado\plus\...
installation complete.
```

Di recente ad `ssc` è stata aggiunta la possibilità di vedere i comandi aggiuntivi (packages) più scaricati negli ultimi tre mesi:

`ssc whatshot [, n(#)]`

dove # specifica il numero di packages da visualizzare (`n(10)` è il valore di default). Specificando `n(.)` verrà visualizzato l'intero elenco.

```
. ssc whatshot, n(12)

Top 12 packages at SSC

       Oct2007
  Rank   # hits    Package       Author(s)
  ------------------------------------------------------------------------
     1   1214.0    outreg        John Luke Gallup
     2    911.1    estout        Ben Jann
     3    847.6    xtabond2      David Roodman
     4    830.8    outreg2       Roy Wada
     5    788.6    ivreg2        Mark E Schaffer, Christopher F Baum,
                                   Steven Stillman
     6    667.8    psmatch2      Edwin Leuven, Barbara Sianesi
     7    508.2    gllamm        Sophia Rabe-Hesketh
     8    320.3    xtivreg2      Mark E Schaffer
     9    315.3    overid        Christopher F Baum, Mark E Schaffer,
                                   Steven Stillman, Vince Wiggins
    10    266.0    tabout        Ian Watson
    11    251.0    ranktest      Mark E Schaffer, Frank Kleibergen
    12    246.4    metan         Mike Bradburn, Ross Harris, Jonathan
                                   Sterne, Doug Altman, Roger Harbord,
                                   Thomas Steichen, Jon Deeks
  ------------------------------------------------------------------------
  (Click on package name for description)
```

Siete curiosi di vedere tutti i pacchetti disponibili? Andate in Appendice B (pag. 267). Esiste anche la possibilità di installare i nuovi comandi attraverso la funzione di ricerca. In questo caso vengono fornite direttamente le indicazioni da seguire[5].
Non è raro (anzi) che questi nuovi comandi vengano corretti per dei bugs, oppure migliorati con l'aggiunta di nuove funzioni. Per controllare gli update di tutti i nuovi comandi installati si usa il comando

`adoupdate [pkglist] [, options]`

```
. adoupdate, update
(note: adoupdate updates user-written files;
       type -update- to check for updates to official Stata)

Checking status of installed packages...

    [1] mmerge at http://fmwww.bc.edu/repec/bocode/m:
        installed package is up to date

    [2] sg12 at http://www.stata.com/stb/stb10:
        installed package is up to date
```

[5] In pratica la procedura vi dirà cosa cliccare per procedere automaticamente all'installazione.

```
   (output omitted)

    [96] sjlatex at http://www.stata-journal.com/production:
         installed package is up to date

    [97] hotdeck at http://fmwww.bc.edu/repec/bocode/h:
         installed package is up to date

Packages to be updated are...

   [90] examples     -- ´EXAMPLES´: module to show examples from on-line help files

Installing updates...

   [90] examples

Cleaning up... Done
```

il quale si occupa del controllo delle nuove versioni e quindi della loro installazione.

4.6 Fare ricerche

Stata dispone di 2 comandi per cercare informazioni e di un comando per ottenere l'help dei comandi
Per ottenere l'help basta digitare :

`help` [*command_or_topic_name*] [, `options`]

Per fare ricerche si possono usare indifferentemente:

`search` *word* [*word* ...] [, *search_options*]

oppure

`findit` *word* [*word* ...]

Personalmente preferisco il secondo. Entrambi i comandi effettuano una ricerca sui comandi e sulla documentazione locale e su tutte le risorse di Stata disponibili in rete. Un esempio (`findit` fornisce lo stesso risultato):

```
. search maps, all

Keyword search

        Keywords:  maps
          Search:  (1) Official help files, FAQs, Examples, SJs, and STBs
                   (2) Web resources from Stata and from other users

Search of official help files, FAQs, Examples, SJs, and STBs



Web resources from Stata and other users

(contacting http://www.stata.com)

9 packages found (Stata Journal and STB listed first)
-----------------------------------------------------
```

```
labutil from http://fmwww.bc.edu/RePEc/bocode/l
    ´LABUTIL´: modules for managing value and variable labels / labcopy copies
    value labels, or swaps them around. labdel deletes / them.  lablog defines
    value labels for values which are base 10 / logarithms containing the
    antilogged values. labcd defines value / labels in which decimal points

mca from http://fmwww.bc.edu/RePEc/bocode/m
    ´MCA´: module to perform multiple correspondence analysis / The command
    mca produces numerical results as well as graphical / representations for
    multiple correspondence analyses (MCA). mca / actually conducts an
    adjusted simple correspondence analysis on / the Burt matrix constructed

mif2dta from http://fmwww.bc.edu/RePEc/bocode/m
    ´MIF2DTA´: module convert MapInfo Interchange Format boundary files to
    Stata boundary files / This is a program that converts MapInfo Interchange
    / Format boundary files into Stata boundary files to be used / with the
    latest release of the -tmap- package.  / KW: maps / KW: MapInfo /

shp2dta from http://fmwww.bc.edu/RePEc/bocode/s
    ´SHP2DTA´: module to converts shape boundary files to Stata datasets /
    shp2dta reads a shape (.shp) and dbase (.dbf) file from disk and /
    converts them into Stata datasets.  The shape and dbase files / must have
    the same name and be saved in the same directory.  The / user-written

spmap from http://fmwww.bc.edu/RePEc/bocode/s
    ´SPMAP´: module to visualize spatial data / spmap is aimed at visualizing
    several kinds of spatial data, and / is particularly suited for drawing
    thematic maps and displaying / the results of spatial data analyses.
    Proper specification of / spmap options and suboptions, combined with the

tmap from http://fmwww.bc.edu/RePEc/bocode/t
    ´TMAP´: module for simple thematic mapping / This is a revised version of
    the package published in The / Stata Journal 4(4):361-378 (2004) for
    carrying out simple / thematic mapping. This new release should be
    considered as a / beta version: comments and problem reports to the author

triplot from http://fmwww.bc.edu/RePEc/bocode/t
    ´TRIPLOT´: module to generate triangular plots / triplot produces a
    triangular plot of the three variables / leftvar, rightvar and botvar,
    which are plotted on the left, / right and bottom sides of an equilateral
    triangle. Each should / have values between 0 and some maximum value

usmaps from http://fmwww.bc.edu/RePEc/bocode/u
    ´USMAPS´: module to provide US state map coordinates for tmap / This
    package contains several Stata datafiles with US state / geocode
    coordinates for use with Pisati´s tmap package (Stata / Journal, 4:4,
    2004). A do-file illustrates their usage.  / KW: maps / KW: states / KW:

usmaps2 from http://fmwww.bc.edu/RePEc/bocode/u
    ´USMAPS2´: module to provide US county map coordinates for tmap / This
    package contains contains several Stata datafiles with US / county geocode
    coordinates for use with Pisati´s tmap package / (Stata Journal, 4:4,
    2004). A do-file illustrates their usage.  / KW: maps / KW: counties / KW:

(end of search)
```

4.7 Cura dei dati

Alcune considerazioni riguardanti la cura e la sicurezza dei dati e dei programmi:

1. Adibire **una cartella per ciascun progetto** e racchiudere tutti i progetti in una cartella. Personalmente ho una cartella *projects* all'interno della quale ci sono

le cartelle con i vari progetti in corso di svolgimento. Man mano che i progetti terminano vengono spostati nella cartella *ended_progects*

```
G:\projects

. dir
  <dir>   8/25/07  8:16  .
  <dir>   8/25/07  8:16  ..
  <dir>   2/19/04 18:11  ABI
  <dir>   6/02/05  8:28  banche
  <dir>   5/01/05 11:46  bank_efficiency
  <dir>   6/14/07 20:23  BEI
  <dir>   5/05/07  9:19  comune
  <dir>   6/17/06 16:44  conti_intergenerazionali
  <dir>   8/04/07 10:35  coorti
  <dir>   3/11/04 22:16  ended_projects
  <dir>   5/14/05  9:28  ESEV
  <dir>   5/12/07 11:53  gerosa
  <dir>   8/13/04  7:55  instrumental_variables
  <dir>   3/25/07 10:13  isee
  <dir>   8/01/07 17:41  ISMEA
  <dir>   5/01/05 10:17  ISTAT
  <dir>   6/18/05  8:25  medici
  <dir>   5/21/06  8:33  oculisti
  <dir>   8/25/07  8:26  popolazione
  <dir>   6/20/06 11:50  provincia
  <dir>   6/23/07 10:14  scale2000
  <dir>  11/20/04 11:41  scale_equivalenza
  <dir>   6/02/07  8:54  shape
  <dir>   5/01/07 10:25  silc
  <dir>   8/11/07  7:55  s_cuore
```

2. All'interno di ciascuna cartella di progetto stabilire un **ordine di cartelle** che rifletta lo svolgimento logico del lavoro. Per esempio la lettura di dati in formato testo e il salvataggio di questi in formato Stata deve precedere le elaborazioni su questi dati.

```
. cd conti_intergenerazionali
G:\projects\conti_intergenerazionali

. dir
  <dir>   6/17/06 16:44  .
  <dir>   6/17/06 16:44  ..
  <dir>   6/24/06 15:52  00_docs
  <dir>   4/25/06  8:18  01_original_data
  <dir>   6/02/06  9:29  02_final_data
  <dir>   6/02/06  9:29  03_source
  <dir>   6/02/06  9:29  04_separazioni
  <dir>   6/04/06 11:39  05_disoccupazione
  <dir>   6/02/06  9:29  06_povertà
  <dir>   6/25/06  9:13  99_GA
   0.5k   8/30/05  8:50  master.do
```

3. Ci dovrebbe sempre essere un file **master.do** che si occupa di lanciare tutti i files .do nell'ordine corretto.
   master.do di *conti_intergenerazionali*

```
#delimit;
clear;
set mem 250m;
set more off;
capture log close;
```

```
cd 02_final_data;
do read.do /** che lancia, nell´ordine -panel_link.do
                                       -panel_a.do
                                       -panel_h.do
         ****/;
cd ..;

cd 03_source;
do master.do;
cd ..;


cd 04_separazioni;
do master.do;
cd ..;


cd 05_disoccupazione;
do master.do;
cd ..;

cd 99_GA;
do master.do;
cd ..;
```

master.do di *03_source*

```
clear

do rela.do
do coppie.do
do rela_by_wave.do
do hids.do
do sons.do
do occupati.do
```

4. Usare sempre **percorsi relativi**.
5. I files di **dati di partenza** devono rimanere **inalterati**. Se i dati di partenza vengono in qualsiasi modo modificati vanno salvati con un altro nome. Altrimenti si inficia il principio di riproducibilità
6. Dare ai files di log lo stesso nome del file do che li genera.
7. Fare un **backup** giornaliero dei propri progetti (sia files di dati che files .do). Un backup fatto male (o non fatto) può far piangere anche un uomo grande e grosso.
8. I dati sensibili vanno protetti. Si possono separare gli identificativi personali dal resto dei dati e poi i files con questi dati andrebbero criptati.

4.8 Intestazione file .do

Naturalmente questa è solo un'indicazione per nulla vincolante; ciascuno faccia come meglio crede, ma io consiglio di iniziare i files .do così:

```
#delimit;
version 10;
clear;
set mem 250m;
set more off;
capture log close;
log using panel.log, replace;
```

Cosa faccio con questo incipit?

`version 10;` definisco la versione di Stata che sto' usando e quindi tutti i comandi successivi verranno eseguiti secondo quella versione. Ciò è importante per conservare la compatibilità in relazione alle successive versioni di Stata. In altre parole Stata 99 sarà in grado di eseguire questo file .do

`#delimit;` definisco il delimitatore di fine comando

`clear;` elimino eventuali dati in memoria

`set mem 250m;` assegno un adeguato quantitativo di memoria

`set more off;` disabilito lo stop nello scorrimento qualora l'output di un comando ecceda la lunghezza della schermata della finestra dei risultati del programma

`capture log close;` chiudo un eventuale file di log aperto

`log using xxxxxx.log, replace;` avvio la registrazione degli output. Con `replace` sovrascrivo un eventuale file di log con lo stesso nome. Possibilmente assegnare al file `xxxxxx.log` lo stesso nome del file .do.

Se vengono usati dei comandi aggiuntivi non presenti nella distribuzione ufficiale per non bloccare l'esecuzione del file .do, è utile inserire sempre all'inizio il comando `which`:

```
capture which mmerge;
if _rc ssc install mmerge;
```

`capture which` verifica che il comando `mmerge` sia installato. Nel caso in cui non lo fosse, Stata accede all'archivio `ssc` e lo installa. Questo metodo funziona solo se il comando risiede nell'archivio `ssc`, altimenti bisognerà procedere con una installazione manuale.

P.S.: Il nome del file .do dovrebbe essere breve (non più di otto lettere diciamo) e non deve contenere spazi bianchi.

Capitolo 5

Alcuni Concetti di Base

5.1 L'input dei dati

5.1.1 Caricamento dei dati in formato proprietario

Vale la regola generale che la realise più recente legge i dati scritti nelle realise precedenti, ma le precedenti non leggono quelle più recenti. Inoltre bisogna tener presente anche la versione del programma secondo il presente schema

Dati salvati da \ Dati letti da	**StataMP**	**StataSE**	**Intercooled**	**Small**
StataMP	SI	SI	NO	NO
StataSE	SI	SI	NO	NO
Intercooled	SI	SI	SI	SI(?)
Small	SI	SI	SI	SI

Il comando per caricare i dati in formato proprietario di Stata (estensione .dta) è

`use` *filename* [`, clear`]

L'opzione `clear` è necessaria per pulire la memoria dall'eventuale presenza di altri dati, in quanto non ci possono essere 2 database contemporaneamente in memoria. Questo argomento viene trattato in forma maggiormente estesa e dettagliata nel capitolo 6.1 alla pagina 39.

5.1.2 Caricamento dei dati in formato testo

Esistono diversi comandi in Stata per caricare dati in formato testo (ASCII). Val la pena di ricordare che questo formato sarebbe da preferire quando i dati saranno utilizzati anche con altri programmi[1].

[1] I dati in formato testo sono leggeri in termini di dimensione del file, molto raramente si danneggiano e sono utilizzabili anche su piattaforme diverse da quelle Microsoft.

La prima cosa da sapere è se i dati sono delimitati o non delimitati. I dati sono delimitati se ciascuna variabile è separata da un certo carattere, di solito

- '.'
- ',' [2]
- ';'
- '|'
- '<tab>'

Qui viene fatta solo un'introduzione ai dati in formato testo. La trattazione per esteso verrà fatta nel capitolo 6.2 alla pagina 43.

5.1.3 Caricamento dei dati in altri formati proprietari (StatTransfer)

È possibile convertire dataset da altri formati al formato di Stata attraverso il programma commerciale StatTransfer, consigliato dalla stessa Stata Corp. Questo programma è usabile anche direttamente all'interno di Stata tramite appositi comandi che vedremo più avanti (`inputst` e `outputst`) nel capitolo 6.3 alla pagina 47.

5.2 Regole per denominare le variabili

Esistono due metodi per nominare le variabili: assegnare un nome evocativo o assegnare un codice. Per esempio possiamo chiamare `redd` la variabile che contiene l'informazione sul reddito o `age` la variabile che contiene l'informazione sull'età. In questa maniera il nome della variabile ci aiuta a richiamare il suo contenuto informativo. Se però abbiamo centinaia di variabili, assegnare a ciascuna un nome evocativo può diventare problematico. In questo contesto meglio ricorrere ad una nomenclatura di tipo sistematico. Per esempio assegnare `d01 d02 d03 d04 d05` alle risposte delle domande da 1 a 5 o nomi del tipo `score_10 score_11 ...` o ancora `14a_rc d14b_rc d14c_rc d14d_rc d14e_rc d14f_rc`.
Quelle che seguono sono regole (e consigli) cui sono sottoposti i nomi che intendiamo assegnare alle variabili:

1. Ogni variabile deve avere il suo nome
2. Il nome di ciascuna variabile deve essere univoco
3. Il nome delle variabili è *case sensitive* per cui `redd` è diverso da `REDD` o da `Redd`
4. Il nome può contenere lettere (sia maiuscole che minuscole), numeri e il carattere *underscore* (_). Non può contenere:
   (a) spazi
   (b) trattini (-)
   (c) caratteri non alfabetici o non numerici (. , ; : € # § * ^ ? ' =) ([ ] / \ & % $ £ " ! | > <)

[2] I caratteri '.' e ',' non sono consigliati in quanto possono generare confusione in relazione alla sintassi numerica europea e anglosassone.

5. Il nome non può iniziare con un numero
6. La lunghezza non può superare i 32 caratteri anche se per motivi di praticità è consigliabile non superare la decina di caratteri
7. Possibilmente usare solo lettere minuscole (sempre per motivi di praticità)
8. Meglio non usare lettere accentate

5.3 Il qualificatore *in*

Buona parte dei comandi di Stata supportano l'uso del qualificatore *in* che, assieme al qualificatore *if*, consente di restringere l'insieme delle osservazioni su cui applicare il comando. Si noti che questo qualificatore risente dell'ordinamento dei dati, nel senso che fa riferimento alla posizione assoluta dell'osservazione. Un piccolo esempio può aiutare la comprensione di questo concetto. Supponiamo di avere 10 osservazioni per 2 variabili come segue:

```
  sex  age
1.  1   45
2.  2   22
3.  1   11
4.  1   36
5.  2   88
6.  1   47
7.  2   72
8.  2   18
9.  2   17
```

se eseguo i seguenti comandi

```
. list sex age in 2/6

     +-----------+
     | sex   age |
     |-----------|
  2. |   2    22 |
  3. |   1    11 |
  4. |   1    36 |
  5. |   2    88 |
  6. |   1    47 |
     +-----------+

. summ age in 2/6

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |         5        40.8    29.71027         11         88
```

Stata mostra le osservazione dalla 2. alla 6. ed esegue il comando `summ` sulle osservazioni 2.-6.
Se adesso ordino le il dataset in base alla variabile `age`

```
. sort age

. list

     +-----------+
     | sex   age |
```

```
    |-----------|
 1. |   1    11 |
 2. |   2    17 |
 3. |   2    18 |
 4. |   2    22 |
 5. |   1    36 |
    |-----------|
 6. |   1    45 |
 7. |   1    47 |
 8. |   2    72 |
 9. |   2    88 |
    +-----------+
```

e rieseguo gli stessi comandi

```
. list sex age in 2/6

    +-----------+
    | sex   age |
    |-----------|
 2. |   2    17 |
 3. |   2    18 |
 4. |   2    22 |
 5. |   1    36 |
 6. |   1    45 |
    +-----------+

. summ age in 2/6

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |         5        27.6    12.34099         17         45
```

Stata mostra ancora le osservazione dalla 2. alla 6. ed esegue il comando `summ` sulle osservazioni 2.-6. ma con risultati differenti perchè il comando `sort` ha cambiato la posizione delle osservazioni. Da questo esempio si evidenzia che va posta attenzione nell'uso del qualificatore *in* in quanto il comando associato non viene sempre applicato alle stesse osservazioni, ma dipende dall'ordinamento delle osservazioni (`sort`)

5.4 Il qualificatore *if*

La quasi totalità dei comandi di Stata supporta l'uso del qualificatore *if*. Esso ha la funzione di selezionare le osservazioni su cui applicare il comando vincolando la scelta al verificarsi della condizione specificata nell' *if*. Anche in questo caso un esempio aiuta la comprensione. Sempre facendo riferimento al dataset appena usato:

```
. list sex age if sex==1

    +-----------+
    | sex   age |
    |-----------|
 1. |   1    11 |
 5. |   1    36 |
 6. |   1    45 |
 7. |   1    47 |
    +-----------+

. summ sex age if sex==1
```

```
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         sex |         4           1           0         1          1
         age |         4       34.75    16.54035        11         47
```

I comandi vengono eseguiti solo sulle osservazioni che assumono valore 1 nella variabile `sex`. Il risultato in questo caso è invariante rispetto all'ordinamento:

```
. sort age

. list sex age if sex==1

     +-----------+
     | sex   age |
     |-----------|
  1. |   1    11 |
  5. |   1    36 |
  6. |   1    45 |
  7. |   1    47 |
     +-----------+

. summ sex age if sex==1

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         sex |         4           1           0         1          1
         age |         4       34.75    16.54035        11         47
```

5.5 Operatori di relazione

Gli operatori relazionali in Stata restituiscono sempre una risposta vero/falso. Nel caso sia verificata la relazione, viene eseguito il comando, altrimenti no. Gli operatori di relazioni contemplati nella sintassi di Stata sono:

- `>` (strettamente maggiore di)
- `<` (strettamente minore di)
- `>=` (strettamente maggiore di o uguale a)
- `<=` (strettamente minore di o uguale a)
- `==` (uguale a)
- `~=` o `!=` (diverso da)

Si noti che la relazione di uguaglianza esige l'uso doppio del segno di uguaglianza.
Le relazioni si applicano indifferentemente a dati numerici e a dati in formato stringa. Ed ora alcuni esempi:

- $8 > 4$ restituisce vero
- $8 < 4$ restituisce falso
- "nicola" > "beda" restituisce vero perché 'nicola' in ordine alfabetico è successivo a 'beda'
- "nicola" > "Beda" restituisce falso perché le lettere maiuscole sono ordinate prima delle lettere minuscole

Per i dati missing (indicati con il simbolo '.'), vale la relazione:

- `. > #` ovvero un dato numerico missing è sempre maggiore di una dato numerico non missing.
- `"" > "stringa"` ovvero un dato stringa missing è sempre maggiore di una dato stringa non missing.

Si ricorda anche che all'interno della stessa variabile non possono essere presenti contemporaneamente dati stringa e numerici. In tal caso i secondi vengono convertiti nei primi.

5.6 Operatori logici

Gli operatori logici in Stata sono:

- `&` (and)
- `|` (or)
- `~` o `!` (not)

Gli operatori logici vengono usati per stabilire delle relazioni tra due o più espressioni e restituiscono 1 se sono verificate, 0 se non sono verificate.
`&` richiede che entrambe le relazioni siano verificate
`|` richiede che almeno una delle relazioni sia verificata
Ritornando agli esempi precedenti
$8 > 4$ & $8 < 4$ è una relazione non vera (e quindi restituisce 0)
$8 > 4$ | $8 < 4$ è una relazione vera (e quindi restituisce 1)

5.7 Caratteri jolly e sequenze

In Stata è possibile usare i caratteri jolly per indicare gruppi di variabili. Come è prassi in informatica il carattere `*` serve ad indicare qualsiasi carattere e per un numero qualsiasi di volte. Per esempio, avendo la seguente lista di variabili:

```
redd95
spesa1995
redd96
spesa1996
redd97
spesa1997
redd1998
age
risc
sesso
```

- `*` indica tutte le variabili
- `*95` indica `redd95` e `spesa95`
- `r*` indica `redd95`, `redd96`, `redd97` e `risc`

Il carattere `?` invece serve per indicare un qualsiasi carattere per una sola volta; nel nostro esempio:

- `?` indica nessuna variabile perché non c'è nessuna variabile di un solo carattere, qualsiasi esso sia
- `????95` indica solo `redd95`, ma non `spesa95` (solo 4 caratteri prima di 95)
- `redd??` indica `redd95`, `redd96`, `redd97` ma non `redd1998` (solo 2 caratteri dopo `redd`)

Con il simbolo `-` si indica una successione contigua di variabili; sempre nel nostro caso, `redd96-risc` indica `redd96`, `spesa1996`, `redd97`, `spesa1997`, `redd1998`, `age`, `risc`.

Si faccia attenzione che il simbolo `-` dipende da come sono disposte le variabili. Se la variabile `redd97` venisse spostata all'inizio della lista, non rientrerebbe più nell'elenco.

5.8 L'espressione `by`

Molti comandi hanno la caratteristica di essere *byable*, ovvero supportano l'uso del prefisso `by`. In sostanza il `by` serve per ripetere un comando più volte in base ad una certa variabile (categorica). Supponiamo di avere l'età (`age`) di N individui e di sapere per ciascuno di essi se risiede nelle macro regioni nord, centro o sud+isole (`macro3`). Volendo conoscere l'età media per ciascuna delle macro regioni (nord=1, centro=2, sud+isole=3):

```
. summ age if macro3==1

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |     12251    55.90948    15.82015         19        101

. summ age if macro3==2

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |      5253    56.56958    16.03001         19         98

. summ age if macro3==3

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |      9995    55.96738    15.69984         21        102
```

oppure, ricorrendo al `by` e all'uso di una sola riga di comando al posto delle 3 precedenti:

```
. by macro3, sort: summ age

------------------------------------------------------------------------------
-> macro3 = Nord

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |     12251    55.90948    15.82015         19        101

------------------------------------------------------------------------------
-> macro3 = Centro
```

```
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |      5253    56.56958    16.03001         19         98

-------------------------------------------------------------------------------
-> macro3 = Sud & Isole

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |      9995    55.96738    15.69984         21        102
```

Per l'esecuzione tramite `by` bisogna che il dataset sia preventivamente ordinato in base alla variabile categorica, da cui l'uso dell'opzione `sort`. Alternativamente si può ricorrere alla variazione di questo comando:

```
bysort macro3: summ age
```

che da' il medesimo risultato del precedente.
Vedremo in seguito che `by` rientra anche tra le opzioni di molti comandi, per cui esso può assumere la duplice natura di prefisso e di opzione.

5.9 Dati missing

Stata identifica con il simbolo '.' un dato missing numerico. Questa è la sua rappresentazione generale ma c'è la posibilità di definire un sistema di identificazione di valori missing di diversa natura. Per esempio un dato missing per mancata risposta è concettualmente diverso da un dato missing dovuto al fatto che quella domanda non può essere posta. Un dato missing sull'occupazione di un neonato non è una mancata risposta ma una domanda che non può essere posta. In Stata possiamo definire diversi missing secondo la struttura `.a, .b, .c, ... .z` e vale l'ordinamento:

tutti i numeri non missing `< . < .a < .b < ... < .z`

Poi a ciascuno di questi diversi missing possiamo assegnare una sua label:

```
label define 1 "......"
             2 "......"
             ............
             .a "Non risponde"
             .b "Non sa"
             .c "Non appicabile"
```

Quanto esposto precedentemente si riferisce a dati numerici. Per le variabili stringa non esiste nessun metodo di codifica e il dato missing corrisponde ad una cella vuota (nessun simbolo e nessuno spazio). Nel caso si debba fare riferimento ad un dato missing stringa si usano le doppie vigolette come segue:

```
. desc  q01 q03

              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
q01             int    %8.0g                  q01 età
```

```
q03             str29  %29s                  q03 comune di residenza

. summ q01 if q03==""

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         q01 |        12       49.75    29.19877          1         90
```

Capitolo 6

Il Caricamento dei Dati

6.1 Dati in formato proprietario (.dta)

Caricare i dati in formato Stata (.dta) è un'operazione semplice e come vedremo ci sono diverse utili opzioni. Ma prima di caricare un dataset bisogna porre attenzione alla sua dimensione. Come già accennato Stata mantiene tutti i dati nella memoria RAM per cui bisogna allocarne un quantitativo adeguato, il quale, sarà sottratto alla memoria di sistema. Se per esempio dobbiamo caricare un file di dati di 88MB dobbiamo dedicare al programma questo quantitativo aumentato in funzione della eventuale creazione di nuove variabili. Se possibile consiglio di allocare un quantitativo di RAM all'incirca doppio rispetto al dataset di partenza se si dovranno creare molte nuove variabili, altrimenti un incremento del 50% può essere sufficiente dato che un certo quantitativo di RAM viene comunque utilizzato per le elaborazioni. Stata è impostato con una allocazione di default di circa 1.5MB.
Nel momento in cui avviate il programma vi viene fornita l'informazione circa l'attuale allocazione di RAM.

```
Notes:
      1.  (/m# option or -set memory-) 10.00 MB allocated to data
      2.  (/v# option or -set maxvar-) 5000 maximum variables
```

Il comando per allocare un diverso quantitativo di memoria è :

set memory #[b|k|m|g][, permanently]

```
. set mem 250m

Current memory allocation

                    current                                 memory usage
    settable          value     description                 (1M = 1024k)
    ------------------------------------------------------------------
    set maxvar         5000     max. variables allowed           1.909M
    set memory          250M    max. data space                250.000M
    set matsize         400     max. RHS vars in models          1.254M
                                                              -----------
                                                               253.163M
```

e va eseguito *prima* di caricare il dataset, ovvero con nessun dataset in memoria[1]. Inoltre bisogna tener presenti le seguenti limitazioni:

- il quantitativo di RAM dedicato non deve superare la RAM totale del computer e tenete presente che un certo quantitativo serve anche per il normale funzionamento del sistema operativo.
- attualmente Windows ha problemi ad allocare quantitativi superiori ai 950MB[2].

Se volete allocare in maniera permanente un certo quantitativo di RAM in maniera che ad ogni avvio questo sia a disposizione di Stata:

```
set mem #m, perm
```

Se il quantitativo di memoria non è sufficiente, Stata non carica i dati:

```
. use istat03, clear
(Indagine sui Consumi delle Famiglie - Anno 2003)
no room to add more observations
    An attempt was made to increase the number of observations beyond what is
    currently possible.
    You have the following alternatives:

     1.  Store your variables more efficiently; see help compress.
     (Think of Stata´s data area as the area of a rectangle; Stata can trade
      off width and length.)

     2.  Drop some variables or observations; see help drop.

     3.  Increase the amount of memory allocated to the data area using the set
      memory command; see help memory.
r(901);

. set mem 5m

Current memory allocation

                    current                                    memory usage
    settable          value     description                  (1M = 1024k)
    ------------------------------------------------------------------------
    set maxvar         5000     max. variables allowed            1.909M
    set memory            5M    max. data space                   5.000M
    set matsize          400    max. RHS vars in models           1.254M
                                                                -----------
                                                                  8.163M

. use istat03, clear
(Indagine sui Consumi delle Famiglie - Anno 2003)

. desc, short

Contains data from istat03.dta
  obs:         2,000                          Indagine sui Consumi delle
                                                Famiglie - Anno 2003
 vars:           551                          23 Nov 2006 09:13
 size:     2,800,000 (46.6% of memory free)
Sorted by:
```

[1] Ricordo che ci può essere un solo dataset in memoria.

[2] Il problema per la versione italiana dovrebbe essere risolto con il prossimo rilascio del service pack 3 di Windows XP.

Allocato un quantitativo adeguato di RAM, siamo pronti per caricare il nostro dataset. Abbiamo già visto l'uso di base del comando `use` nella sezione 5.1.1 (pagina 29).
Si noti anche che il file di dati può essere caricato da un indirizzo internet.
Una versione più evoluta del comando `use`, è questa:

`use` [*varlist*] [*if*] [*in*] `using` *filename* [`, clear nolabel`]

dove:

- in *varlist* possiamo mettere l'elenco delle variabili da caricare nel caso non le si voglia tutte
- in *if* possiamo specificare di voler caricare solo quelle osservazioni che rispondono a certi criteri
- in *in* possiamo specificare di voler caricare solo un range di osservazioni

E adesso proviamo ad usare i comandi appena visti:

```
. clear

. set mem 15m

Current memory allocation

                    current                                 memory usage
    settable          value     description                 (1M = 1024k)
    ------------------------------------------------------------------
    set maxvar         5000     max. variables allowed           1.909M
    set memory           15M    max. data space                 15.000M
    set matsize          400    max. RHS vars in models          1.254M
                                                               ----------
                                                                 18.163M

. use carica, clear

. desc, short

Contains data from carica.dta
  obs:         1,761
 vars:            80                          18 Oct 2006 10:30
 size:       294,087 (98.1% of memory free)
Sorted by:

. use hhnr persnr sex using carica, clear

. desc, short

Contains data from carica.dta
  obs:         1,761
 vars:             3                          18 Oct 2006 10:30
 size:        22,893 (99.9% of memory free)
Sorted by:

. use if sex==2 using carica, clear

. desc, short

Contains data from carica.dta
  obs:           898
 vars:            80                          18 Oct 2006 10:30
 size:       149,966 (99.0% of memory free)
```

```
Sorted by:


. use in 8/80 using carica, clear

. desc, short

Contains data from carica.dta
  obs:            73
 vars:            80                          18 Oct 2006 10:30
 size:        12,191 (99.8% of memory free)
Sorted by:
```

Ed ecco anche un esempio di dati caricati da internet

```
. use http://www.stata-press.com/data/r9/union.dta, clear
(NLS Women 14-24 in 1968)

. desc, short

Contains data from http://www.stata-press.com/data/r9/union.dta
  obs:        26,200                          NLS Women 14-24 in 1968
 vars:            10                          27 Oct 2004 13:51
 size:       393,000 (92.5% of memory free)
Sorted by:
```

È possibile migliorare l'uso della memoria attraverso un processo che ottimizzi il quantitativo di memoria occupato da ciascuna variabile. Per esempio se una variabile può assumere solo valori interi 1 o 2, è inutile sprecare memoria per i decimali. Il comando deputato a ciò in Stata è:

`compress` [*varlist*]

```
. use istat_long, clear

. desc, short

Contains data from istat_long.dta
  obs:        46,280
 vars:            13                          26 Mar 2004 17:54
 size:     2,406,560 (95.4% of memory free)
Sorted by:  anno  fam_id

. compress
sons_head was float now byte
sons_head_00_18 was float now byte
sons_head_00_05 was float now byte
sons_head_06_14 was float now byte
sons_head_15_18 was float now byte
sons_head_19_oo was float now byte
nc was float now byte
couple was float now byte
parents was float now byte
relatives was float now byte
hhtype was float now byte

. desc, short

Contains data from istat_long.dta
  obs:        46,280
 vars:            13                          26 Mar 2004 17:54
 size:       879,320 (98.3% of memory free)
```

```
Sorted by:  anno  fam_id
```

Come si può notare dalla riga intestata `size:` la dimensione del dataset si è ridotta di un fattore 3 (non male vero?).

6.2 Dati in formato testo

Spesso i dataset vengono forniti in formato testo. Questa scelta è dettata dal fatto che il formato testo è multi piattaforma e che può essere letto da tutti i programmi di analisi statistica. Per l'utilizzo in Stata si distingue tra dati in formato testo delimitato e non delimitato.

6.2.1 Formato testo delimitato

Questi dataset sono caratterizzati dal fatto che ciascuna variabile è divisa dalle altre da un determinato carattere o da tabulazione. Naturalmente non tutti i caratteri sono adatti a fungere da divisori e in generale i più utilizzati sono:

- ','
- ';'
- '|'
- <spazio>
- <tabulazione>

Il comando deputato alla lettura di questi dati è:

`insheet` [*varlist*] `using` *filename* [`,` *options*]

tra le opzioni più importanti:

- `tab` per indicare che i dati sono divisi da tabulazione
- `comma` per indicare che i dati sono divisi da virgola
- `delimiter`("*char*") per specificare tra "" quale carattere fa da divisore (per es. "|")
- `clear` da aggiungere sempre per pulire eventuali altri dati in memoria

per esempio il comando

```
insheet using dati.txt, tab clear
```

legge le variabili contenute nel file `dati.txt` dove una tabulazione funge da divisore.

```
insheet var1 var2 var10 dati.txt, delim("|")
```

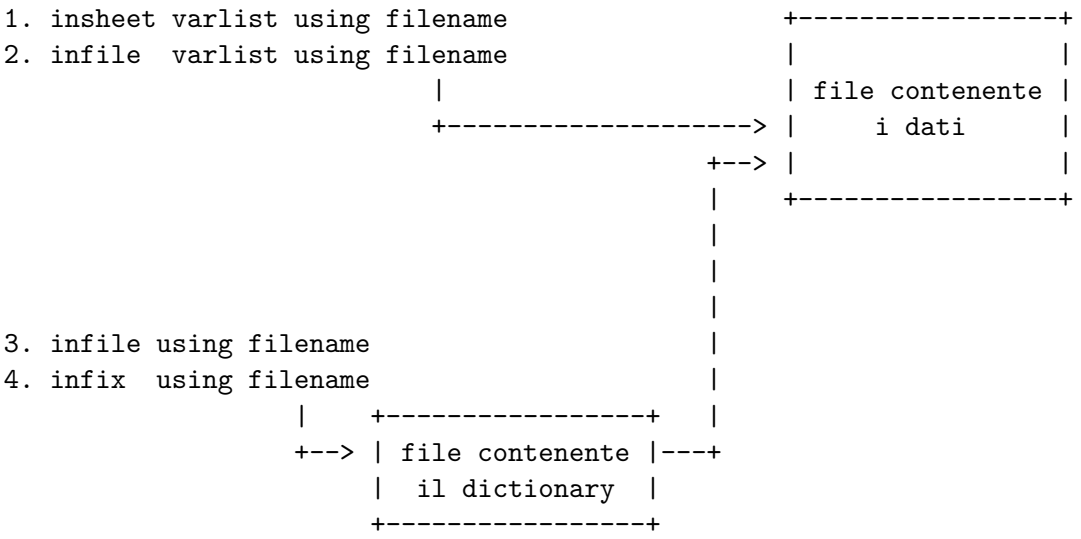
legge tutte le variabili `var1`, `var2` e `var10` nel file `dati.txt` dove il carattere '|' funge da divisore.
Nel caso in cui il divisore sia uno spazio (caso abbastanza raro in realtà) si può usare il comando:

`infile` *varlist* [`_skip`[(#)][*varlist* [`_skip` [(#)]...]]] `using` *filename* [*if*][*in*][`,` *options*]

quest'ultimo comando prevede anche l'uso del file dictionary che sarà trattato per esteso per i dati in formato testo non delimitato.

6.2.2 Formato testo non delimitato

Per capire come Stata può acquisire questo tipo di dati ci serviamo del seguente schema:



I casi 1. e 2. sono tipici dei file di testo delimitati e lo `using` fa riferimento al file che contiene i dati (`filename`).
Nei casi 3. e 4. il procedimento da seguire si snoda nelle seguenti fasi:

a. Si impartisce il comando senza la lista delle variabili e lo `using` fa riferimento al file dictionary (`filename`).
b. Il file dictionary deve avere estensione .dct, altrimenti va indicato completo di nuova estensione nel comando (es.: `infile using filename.txt`)
c. Nel file dictionary si indicano il file che contiene i dati e le variabili da leggere (che possono essere indicate in varie maniere)
d. Le indicazioni contenute nel file dictionary vengono usate per leggere i dati in formato non delimitato.

Adesso analizziamo la struttura di un file dictionary. Anche questo è un semplice file di testo che inizia con la riga:

```
infile dictionary using data.ext
```

oppure

```
infix dictionary using data.ext
```

a seconda del comando che vogliamo utilizzare e dove `data.ext` è il file contenente i dati.
Le varianti e le opzioni all'interno dei file dictionary sono molte. In questa sezione tratteremo solo i casi classici. Per i casi di salti di variabili, di salti di righe o di osservazioni distribuite su 2 o più righe si rimanda ad una prossima versione più completa ed approfondita sull'argomento.

Costruzione del dictionary per il comando `infile`
È un tipo di dictionary poco usato in verità. La struttura è:

```
infile dictionary  using datafile.ext {
nomevar tipo&lenght "label"
...
...
}
```

La parte "più difficile" da costruire è quella centrale in quanto bisogna porre attenzione alla lunghezza delle singole variabili che solitamente sono indicate nella documentazione che accompagna i dati. Per esempio:

```
infile dictionary  using datafile.ext {
var1          %1f     "label della var1"
var2          %4f     "label della var2"
var3        %4.2f     "label della var3"
str12 var4   %12s     "label della var4"
}
```

dove:
`var1` è numerica ed occupa uno spazio, quindi è un intero 0-9
`var2` è numerica, occupa 4 spazi senza decimali (0-9999)
`var3` è numerica, occupa 4 spazi per la parte intera, più uno spazio per il simbolo decimale, più 2 decimali
`var4` è stringa (e questo deve essere specificato prima del nome della variabile) ed occupa 12 spazi.

Data la lunghezza delle singole variabili possiamo ricostruire la struttura del database:

1	1234	4321.11	asdfghjklpoi
1	5678	7456.22	qwertyuioplk
2	9101	9874.33	mnbvcxzasdfr
5	1121	4256.44	yhnbgtrfvcde
9	3141	9632.55	plmqazxdryjn

Un po' complicato vero?? Solo le prime volte, poi ci si fa l'abitudine. Per fortuna nella maggior parte dei casi i dati sono in formato testo delimitato.

Costruzione del dictionary per il comando `infix`
La struttura è:

```
infix dictionary  using datafile.ext {
```

```
nomevar inizio-fine
...
...
}
```

Anche in questo caso la colonna di inizio e quella di fine delle variabili vengono fornite con la documentazione che accompagna i dati. Riprendendo l'esempio precedente il file dictionary sarebbe:

```
infile dictionary  using datafile.ext {
var1         1
var2         2-5
var3         6-12
str12 var4   13-24
}
```

Quello che segue è un estratto dalla documentazione che accompagna il database sui consumi delle famiglie italiane distribuito da ISTAT:

INIZIO	FINE	AMPIEZZA	VARIABILE	CONTENUTO
1429	1429	1	P_7101	Possesso di televisore
1430	1437	8	C_7101	Acquisto televisore
1438	1438	1	P_7102	Possesso di videoregistratore
1439	1446	8	C_7102	Acquisto videoregistratore

Mentre questo è estratto dai Living Standard della World Bank:

VARIABLE	CODE	RT	FROM	LENGTH	TYPE
1 Source of water	Q01	8	9	1	QLN
2 Water piped to house?	Q02	8	10	1	QLN
3 Amount paid water (Rs.)	Q03	8	11	6	QNT
4 Sanitation system	Q04	8	17	1	QLN
5 Garbage disposal	Q05	8	18	1	QLN
6 Amount pd. garbage (Rs.)	Q06	8	19	6	QNT
7 Type of toilet	Q07	8	25	1	QLN

A questo punto, come esercizio sarebbe simpatico provare a costruire il dictionary per questi due esempi. Per i dati ISTAT, per prima cosa da Stata si impartisce il comando:

```
infix using istat_rid.dct, clear
```

come abbiamo visto questo comando richiama il file dictionary `istat_rid.dct` che ha la seguente struttura:

```
infix dictionary using istat_rid.dat {
p_7101           1429-1429
c_7101           1430-1437
p_7102           1438-1438
c_7102           1439-1446
}
```

il quale chiama a sua volta il file dei dati `istat_rid.dat` ottenendo questo output:

```
. infix using istat_rid.dct, clear;

infix dictionary using istat_rid.dat {
```

```
p_7101          1429-1429
c_7101          1430-1437
p_7102          1438-1438
c_7102          1439-1446}
(88 observations read)
```

Per il file della World Bank invece il dictionary ha la seguente struttura:

```
dictionary using RT008.DAT {
  _column(  9) byte   S02C1_01 %1f  "1 Source of water"
  _column( 10) byte   S02C1_02 %1f  "2 Water piped to house?"
  _column( 11) long   S02C1_03 %6f  "3 Amount paid water (Rs.)"
  _column( 17) byte   S02C1_04 %1f  "4 Sanitation system"
  _column( 18) byte   S02C1_05 %1f  "5 Garbage disposal"
  _column( 19) long   S02C1_06 %6f  "6 Amount pd. garbage (Rs.)"
  _column( 25) byte   S02C1_07 %1f  "7 Type of toilet"
}
```

in questo caso usiamo il comando `infile` ottenendo:

```
. infile using Z02C1, clear

dictionary using RT008.DAT {
  _column(  9) byte   S02C1_01 %1f  "1 Source of water"
  _column( 10) byte   S02C1_02 %1f  "2 Water piped to house?"
  _column( 11) long   S02C1_03 %6f  "3 Amount paid water (Rs.)"
  _column( 17) byte   S02C1_04 %1f  "4 Sanitation system"
  _column( 18) byte   S02C1_05 %1f  "5 Garbage disposal"
  _column( 19) long   S02C1_06 %6f  "6 Amount pd. garbage (Rs.)"
  _column( 25) byte   S02C1_07 %1f  "7 Type of toilet"
}

(3373 observations read)
```

6.3 Altri tipi di formati

Per la lettura di dati salvati in altri tipi di formati proprietari (SPSS, SAS, excel, access, ...) si ricorre, almeno per Stata, al programma StatTransfer[3], pacchetto commerciale che di solito si acquista in abbinamento con Stata. Questo programma può essere usato in maniera indipendente o chiamato direttamente da Stata attraverso i comandi:

`inputst [filetype] infilename.ext [switches]`

per importare dati, e

`outputst [filetype] infilename.ext [switches]`

per esportare dati. Per esempio:

```
inputst database.xls /y
inputst database.xls /tdati /y
```

Nel primo caso si importano i dati del file excel `database.xls`, lo switch `/y` serve a pulire eventuali dati già in memoria; nel secondo caso si leggono i dati dal file `database.xls` contenuti nel foglio `dati`. Alla stessa maniera:

```
outputst database.sav /y
```

[3]Sito web: www.stattransfer.com

esporta gli stessi dati in formato SPSS.

Per i dati contenuti in un file excel potete anche copiarli e incollarli direttamente nel 'Data editor' (e viceversa).

6.4 Esportazione dei dati

Poco fa abbiamo visto un esempio di esportazione dei dati. Se i dati devono essere usati da altri utenti che non usano Stata è consigliabile l'esportazione in formato testo delimitato. Il comando che consiglio di usare è:

`outsheet` [*varlist*] `using` [`filename`] [*if*] [*in*] [`, options`]

dove in filename va specificato il nome del file di output e dove le opzioni principali sono:

`comma` per avere i dati separati da ',' al posto della tabulazione che è l'opzione di default
`delimiter("char")` per scegliere un delimitatore alternativo a ',' e alla tabulazione; per esempio ';', tipico dei files .csv
`nolabel` per esportare il valore numerico e non il label eventualmente assegnato alle variabili categoriche
`replace` per sovrascrivere il file di output già eventualmente creato

Una valida alternativa al formato testo puro è il formato XML. Si definisce come un file di testo strutturato ed ha il vantaggio della portabilità tra le applicazioni che lo supportano. Esistono 2 versioni del comando. Questa salva il dataset in memoria nel formato XML:

`xmlsave` *filename* [*if*] [*in*] [`,` *xmlsave_options*]

Questa salva il sottoinsieme delle variabili specificate:

`xmlsave` [*varlist*] `using` *filename* [*if*] [*in*] [`,` *xmlsave_options*]

le possibili *xmlsave_options* sono:

`doctype(dta)` salva il file XML usando le specifiche determinate da Stata per i files .dta
`doctype(excel)` salva il file XML usando le specifiche determinate per i files Excel
`dtd` include il DTD (Document Type Definition) nel file XML. Questa opzione è usata raramente ed inoltre aumenta la dimensione finale del file
`legible` fa in modo che il file XML visivamente più leggibile. Attenzione che anche questa opzione aumenta la dimensione finale del file
`replace` sovrascrive un eventuale *filename* già esistente

Per inciso, per leggere i dati salvati in XML si usa il comando

`xmluse` *filename* [`,` *xmluse_options*]

che qui non viene trattato
Tutti sappiamo che excel è il formato più diffuso per salvare dati (purtroppo!), ma per favore evitate di esportare i dati in tale formato, poiché excel ha il brutto vizio (ma non è il solo) di "interpretare" i dati come in questo caso:

```
01.11.5 P / 01.24 S
01.11.5 P / 01.24 S
01.11.5 P / 01.21 S
01.11.5 P / 01.2 S
1:11:05 PM
1:11:05 PM
1:11:05 PM
```

Gli ultimi tre casi sono stati interpretati da excel come delle ore, invece sono codici ateco (per la precisione codice 01.11.05).

6.5 Cambiare temporaneamente dataset

Come abbiamo già detto Stata consente l'utilizzo di un solo dataset alla volta. Può allora risultare scomodo salvare il dataset sul quale si sta' lavorando per dedicarsi temporaneamente ad un altro e poi riprendere il primo. In questi casi possiamo ricorrere all'accoppiata di comandi

`preserve`

`restore`

Con `preserve` iberniamo il dataset sul quale stiamo lavorando; possiamo quindi fare dei cambiamenti su questo dataset o passare ad utilizzarne un altro. In seguito con il comando `restore` ritorniamo ad utilizzare il dataset precedentemente ibernato. Nell'esempio che segue si parte con un dataset e dopo il comando `preserve` si prendono solo alcune variabili, si salva il dataset e poi si torna a quello di partenza:

```
. desc, short;

Contains data
  obs:        92,033
 vars:            41
 size:    28,162,098 (91.0% of memory free)
Sorted by:
     Note:  dataset has changed since last saved

. preserve;

. keep tot_att cciaa ateco_1l for_giu prov anno;

. gen flag_test=0;

. save tot_veneto, replace;
file tot_veneto.dta saved

. desc, short;

Contains data from tot_veneto.dta
  obs:        92,033
 vars:             7                          26 Nov 2007 18:00
 size:     7,914,838 (97.5% of memory free)
Sorted by:

. restore;

. desc, short;
```

```
Contains data
  obs:        92,033
 vars:            41
 size:    28,162,098 (91.0% of memory free)
Sorted by:
     Note:  dataset has changed since last saved
```

Nell'esempio seguente invece, dal dataset di partenza di volta in volta si selezionano le osservazioni relative ad un dato anno, si esportano e poi si ripristina ogni volta il dataset di partenza:

```
. tab anno;

       anno |      Freq.     Percent        Cum.
------------+-----------------------------------
       2001 |     16,062       18.13       18.13
       2003 |     23,464       26.49       44.62
       2005 |     49,067       55.38      100.00
------------+-----------------------------------
      Total |     88,593      100.00

. preserve;

. keep if anno==2001;
(72531 observations deleted)

. tab anno;

       anno |      Freq.     Percent        Cum.
------------+-----------------------------------
       2001 |     16,062      100.00      100.00
------------+-----------------------------------
      Total |     16,062      100.00

. keep denom loc prov ateco_1l ric_ven tot_att
>      pos_fn roe roi ros rot effic ac_ric
>      tot_c_pers rapp_ind mat_pc mat_ric cp_rv;

. outputst interm_2001.xls /y;

. restore;

. tab anno;

       anno |      Freq.     Percent        Cum.
------------+-----------------------------------
       2001 |     16,062       18.13       18.13
       2003 |     23,464       26.49       44.62
       2005 |     49,067       55.38      100.00
------------+-----------------------------------
      Total |     88,593      100.00

. preserve;

. keep if anno==2003;
(65129 observations deleted)

. tab anno;

       anno |      Freq.     Percent        Cum.
------------+-----------------------------------
       2003 |     23,464      100.00      100.00
------------+-----------------------------------
      Total |     23,464      100.00
```

```
. keep denom loc prov ateco_1l ric_ven tot_att
>      pos_fn roe roi ros rot effic ac_ric
>      tot_c_pers rapp_ind mat_pc mat_ric cp_rv;

. outputst interm_2003.xls /y;

. restore;

. tab anno;

       anno |      Freq.     Percent        Cum.
------------+-----------------------------------
       2001 |     16,062       18.13       18.13
       2003 |     23,464       26.49       44.62
       2005 |     49,067       55.38      100.00
------------+-----------------------------------
      Total |     88,593      100.00

. preserve;

. keep if anno==2005;
(39526 observations deleted)

. tab anno;

       anno |      Freq.     Percent        Cum.
------------+-----------------------------------
       2005 |     49,067      100.00      100.00
------------+-----------------------------------
      Total |     49,067      100.00

. keep denom loc prov ateco_1l ric_ven tot_att
>      pos_fn roe roi ros rot effic ac_ric
>      tot_c_pers rapp_ind mat_pc mat_ric cp_rv;

. outputst interm_2005.xls /y;

. restore;

. tab anno;

       anno |      Freq.     Percent        Cum.
------------+-----------------------------------
       2001 |     16,062       18.13       18.13
       2003 |     23,464       26.49       44.62
       2005 |     49,067       55.38      100.00
------------+-----------------------------------
      Total |     88,593      100.00
```

Capitolo 7

Gestione delle Variabili

7.1 Descrizione di variabili e di valori

Bene, adesso abbiamo caricato il database in Stata ma per renderlo intellegibile occorre:

a. Descrivere il dataset (questo non è così indispensabile!)
b. Descrivere le variabili (questo invece sì)
c. Descrivere i valori delle variabili categoriche (e anche questo)

Per prima cosa diamo una prima occhiata al dataset sfruttando l'output di due comandi:

describe [*varlist*] [, *memory_options*]

che descrive il dataset senza troncare i nomi troppo lunghi delle variabili.

```
. desc, full

Contains data from C:/Programmi/Stata9/ado/base/u/uslifeexp.dta
  obs:           100                          U.S. life expectancy, 1900-1999
 vars:            10                          30 Mar 2005 04:31
 size:         4,200 (99.9% of memory free)   (_dta has notes)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
year            int    %9.0g                  Year
le              float  %9.0g                  life expectancy
le_male         float  %9.0g                  Life expectancy, males
le_female       float  %9.0g                  Life expectancy, females
le_w            float  %9.0g                  Life expectancy, whites
le_wmale        float  %9.0g                  Life expectancy, white males
le_wfemale      float  %9.0g                  Life expectancy, white females
le_b            float  %9.0g                  Life expectancy, blacks
le_bmale        float  %9.0g                  Life expectancy, black males
le_bfemale      float  %9.0g                  Life expectancy, black females
-------------------------------------------------------------------------------
Sorted by:  year
```

Opzioni interessanti del comando sono:

`short` per avere delle informazioni più limitate, in sostanza numero di variabili, numero di osservazioni e spazio occupato (la prima parte dell'output precedente)
`detail` per avere informazioni più dettagliate
`fullnames` per non abbreviare il nome delle variabili

Il secondo comando che prendiamo in esame è:

`codebook` [*varlist*] [*if*] [*in*] [, *options*]

tra le opzioni + utili:

`notes` per visualizzare le note associate alle variabili
`tabulate(#)` per visualizzare i valori delle variabili categoriche
`problems detail` per riportare eventuali problemi del dataset (doppioni, variabili missing, variabili senza label...) [1]
`compact` per avere un report compatto delle variabili

```
. codebook
-------------------------------------------------------------------------------
candidat                                          Candidate voted for, 1992
-------------------------------------------------------------------------------

                  type:  numeric (int)
                 label:  candidat

                 range:  [2,4]                        units:  1
         unique values:  3                        missing .:  0/15

            tabulation:  Freq.   Numeric  Label
                             5         2  Clinton
                             5         3  Bush
                             5         4  Perot

-------------------------------------------------------------------------------
inc                                                            Family Income
-------------------------------------------------------------------------------

                  type:  numeric (int)
                 label:  inc2

                 range:  [1,5]                        units:  1
         unique values:  5                        missing .:  0/15

            tabulation:  Freq.   Numeric  Label
                             3         1  <15k
                             3         2  15-30k
                             3         3  30-50k
                             3         4  50-75k
                             3         5  75k+

-------------------------------------------------------------------------------
frac                                                           (unlabeled)
-------------------------------------------------------------------------------

                  type:  numeric (float)
```

[1] Attenzione che questa opzione su grandi moli di dati può comportare lunghi tempi di esecuzione. Per esempio su circa 4 milioni di osservazioni con un AMD 4000+ ha impiegato circa 25 minuti.

```
                range:  [16,59]                     units:  1
        unique values:  14                       missing .:  0/15

                 mean:   33.3333
            std. dev:   13.1674

          percentiles:        10%       25%       50%       75%       90%
                               18        20        36        42        48

-------------------------------------------------------------------------------
pfrac                                                           (unlabeled)
-------------------------------------------------------------------------------

                type:  numeric (double)

                range:  [2.08,12.3]                 units:  .01
        unique values:  14                       missing .:  0/15

                 mean:   6.73333
            std. dev:   3.25956

          percentiles:        10%       25%       50%       75%       90%
                             2.52       3.6       6.3       8.4      11.4

-------------------------------------------------------------------------------
pop                                                             (unlabeled)
-------------------------------------------------------------------------------

                type:  numeric (double)

                range:  [32219,190527]              units:  1
        unique values:  14                       missing .:  0/15

                 mean:    104299
            std. dev:   50490.5

          percentiles:        10%       25%       50%       75%       90%
                            39035     55764     97587    130116    176586
```

Infine un comando per avere una analisi alternativa delle variabili:

inspect [*varlist*] [*if*] [*in*]

```
. inspect

candidat:  Candidate voted for, 1992          Number of Observations
-----------------------------------       ------------------------------
                                          Total   Integers   Nonintegers
|  #   #   #                 Negative          -          -          -
|  #   #   #                 Zero              -          -          -
|  #   #   #                 Positive         15         15          -
|  #   #   #                                  -----      -----      -----
|  #   #   #                 Total            15         15          -
|  #   #   #                 Missing           -
+----------------------                    -----
2                     4                      15
   (3 unique values)

      candidat is labeled and all values are documented in the label.

inc:  Family Income                           Number of Observations
------------------                        ------------------------------
                                          Total   Integers   Nonintegers
|  #   #   #   #   #         Negative          -          -          -
```

```
|  #   #   #   #   #         Zero              -         -         -
|  #   #   #   #   #         Positive         15        15         -
|  #   #   #   #   #                         -----     -----     -----
|  #   #   #   #   #         Total            15        15         -
|  #   #   #   #   #         Missing           -
+----------------------                       -----
1                     5                       15
   (5 unique values)

      inc is labeled and all values are documented in the label.

frac:                                            Number of Observations
-------                                         ------------------------------
                                                Total   Integers   Nonintegers
|  #                          Negative             -         -          -
|  #       #                  Zero                 -         -          -
|  #       #                  Positive            15        15          -
|  #       #   #                                 -----     -----      -----
|  #       #   #              Total                15        15          -
|  #   .   #   #   #          Missing               -
+----------------------                         -----
16                    59                         15
  (14 unique values)

pfrac:                                           Number of Observations
--------                                        ------------------------------
                                                Total   Integers   Nonintegers
|  #                          Negative             -         -          -
|  #                          Zero                 -         -          -
|  #       #   #   #          Positive            15         1         14
|  #   #   #   #   #                               -----     -----      -----
|  #   #   #   #   #          Total                15         1         14
|  #   #   #   #   #          Missing               -
+----------------------                         -----
2.08               12.3                         15
  (14 unique values)

pop:                                             Number of Observations
------                                          ------------------------------
                                                Total   Integers   Nonintegers
|  #                          Negative             -         -          -
|  #                          Zero                 -         -          -
|  #       #   #   #          Positive            15        15          -
|  #   #   #   #   #                               -----     -----      -----
|  #   #   #   #   #          Total                15        15          -
|  #   #   #   #   #          Missing               -
+----------------------                         -----
32219            190527                         15
  (14 unique values)
```

Non sempre i dataset sono provvisti di label (descrizione) delle variabili e di label (descrizione) dei valori delle categoriche e quindi bisogna provvedere attraverso il comando `label`.

`label variable` *varname* "*label*"

per associare una etichetta di descrizione alla variabile e

`label data` "*label*"

per associare un titolo al dataset.

Infine vediamo l'associazione delle etichette alle variabili categoriche che si compie in due fasi:

a. Definizione di un "oggetto" label che associa a ciascun valore della variabile categorica la sua descrizione
b. Associazione di tale oggetto alla variabile di pertinenza

a. definizione dell'oggetto label

`label define` *nome_oggetto* #$_1$ *"desc 1"* [#$_2$ *"desc 2"* ... #$_n$ *"desc n"*] [`, add modify nofix`]

b. associazione

`label values` *varname nome_oggetto* [`, nofix`]

A partire dall'aggiornamento del 25 febbraio 2008 la sintassi di `label values` si è arricchita con la possibilità di specificare *varlist* al posto di *varname*. Di conseguenza è possibile eseguire `label values` contemporaneamente su più variabili. La nuova sintassi è la seguente:

`label values` *varlist* [*nome_oggetto* | `.`] [`, nofix`]

Se viene specificato `.` invece di *nome_oggetto*, si azzera l'associazione tra le variabili specificate in *varlist* e la descrizione dei valori contenuta in *nome_oggetto*.

In seguito si potranno aggiungere ad una label già esistente dei nuovi valori con il comando

`label define` *nome_oggetto* #$_t$ *"desc t"* [#$_z$ *"desc z"*] ... [`, add`]

oppure modificare le descrizioni di valori già esistenti con

`label define` *nome_oggetto* #$_i$ *"desc i"* [#$_j$ *"desc j"*] ... [`, modify`]

Ecco un esempio di quanto esposto:

```
. tab q06, miss

 Prenotazione |      Freq.     Percent        Cum.
--------------+-----------------------------------
            1 |         94       48.45       48.45
            2 |         80       41.24       89.69
            3 |          3        1.55       91.24
           .a |         17        8.76      100.00
--------------+-----------------------------------
        Total |        194      100.00

** DEFINIZIONE DELL´OGGETTO LABEL IMPO **
. label define impo 1 "Molto importante"
                    2 "Importante"
                    3 "Poco importante"
                    4 "Per nulla importante";

** AGGIUNTA DI UNA NUOVA SPECIFICAZIONE ALL´OGGETTO LABEL impo **
```

```
. label define impo .a "Non risponde", add;

*ASSOCIAZIONE ALLA VARIABILE q06 DELL´OGGETTO LABEL impo **
. label values q06 impo;
. tab q06, miss

        Prenotazione |      Freq.     Percent        Cum.
---------------------+-----------------------------------
    Molto importante |         94       48.45       48.45
          Importante |         80       41.24       89.69
     Poco importante |          3        1.55       91.24
        Non risponde |         17        8.76      100.00
---------------------+-----------------------------------
               Total |        194      100.00
```

La panoramica sul comando `label` si conclude con:

`label dir` per avere la lista degli oggetti label
`label list nome_oggetto` per vedere i valori delle label
`label drop nome_oggetto | _all` per eliminare una label o per eliminarle tutte

```
. label dir

impo
ulss
cod_com
epd1
epd2

. label list impo

impo:
           1 Molto importante
           2 Importante
           3 Poco importante
           4 Per nulla importante
          .a Non risponde
```

Naturalmente, una volta definito un oggetto label lo si può associare a più variabili usando il comando `label values`.

```
label values q06 q08 q11 q17 q23 impo
```

Per manipolare le labels possiamo ricorrere al comando

`labvalch` *valuelabelname*, [{`from`(*numlist*) `to`(*numlist*) | `swap`(*#1 #2*) }
`delete`(*numlist*) `list`]

che consente operazioni di copia/trasferimento e cancellazione delle value labels. Una volta specificato un set di valori label possiamo convertire label attraverso questo procedimento:

`from`(*numlist*) qui si specificano i valori numerici da trasferire
`to`(*numlist*) qui si specificano i valori numerici che assumeranno le nuove label

in from() e to() ci devono essere gli stessi numeri. Se per esempio abbiamo il set di valori denominato rating e che assume i valori 1 “poor” 2 “fair” 3 “OK” 4 “good” 5 “excellent” e vogliamo invertirne l’ordine

```
labvalch rating, from(1/5) to(5/1)
```

Con `swap(#1 #2)` specifico 2 valori della label che voglio vengano scambiati mentre con `delete(numlist)` indico la lista delle label che desidero eliminare

Potrebbero tornare utili anche i comandi `labelbook` e `numlabel`. `labelbook` produce un rapporto di descrizione delle label:

`labelbook` [*lblname-list*] [, *labelbook_options*]

labelbook_options:

`alpha` ordina le label in ordine alfabetico anziché in base alla numerazione assegnata
`length(#)` controlla se ci sono label uguali fino alla lunghezza specificata in #; il valore di default è `length(12)` dato che questa lunghezza è quella tipicamente usata nell'output dei comandi (`tabulate` per esempio)
`problems` descrive potenziali problemi delle label come per esempio:

1. buchi nei valori mappati (per esempio esistono i valori 1 e 3 ma non il valore 2)
2. le descrizioni contengono spazi all'inizio o alla fine
3. stesse descrizioni associate a valori diversi
4. label identiche fino alla lunghezza di 12 caratteri
5. valori di una label non usati da nessuna variabile

`detail` produce un rapporto dettagliato sulle variabili e sulle label

```
. desc d02 d03 d07 d10_1

              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------
d02             byte   %12.0g      D02        Sesso
d03             byte   %23.0g      D03        Istruzione
d07             byte   %12.0g      D07        Tipo di ricovero
d10_1           byte   %25.0g      D10        Medici

. labelbook D02 D03 D07 D10, problems detail

------------------------------------------------------------------
value label D02
------------------------------------------------------------------

      values                                    labels
       range:  [1,2]                      string length:  [7,12]
           N:  3                   unique at full length:  yes
        gaps:  no                    unique at length 12:  yes
  missing .*:  1                             null string:  no
                                 leading/trailing blanks:  no
                                      numeric -> numeric:  no
  definition
           1   Maschio
           2   Femmina
          .a   Non risponde

   variables:  d02
```

```
-------------------------------------------------------------------
value label D03
-------------------------------------------------------------------

      values                                    labels
       range:  [1,5]                      string length:  [6,23]
           N:  6                   unique at full length:  yes
        gaps:  no                    unique at length 12:  yes
  missing .*:  1                             null string:  no
                                leading/trailing blanks:  no
                                     numeric -> numeric:  no
  definition
           1   Nessun titolo
           2   Licenza elementare
           3   Licenza media inferiore
           4   Diploma media superiore
           5   Laurea
          .a   Non risponde

   variables:  d03


-------------------------------------------------------------------
value label D07
-------------------------------------------------------------------

      values                                    labels
       range:  [1,2]                      string length:  [7,12]
           N:  3                   unique at full length:  yes
        gaps:  no                    unique at length 12:  yes
  missing .*:  1                             null string:  no
                                leading/trailing blanks:  no
                                     numeric -> numeric:  no
  definition
           1   Urgente
           2   Programmato
          .a   Non risponde

   variables:  d07


-------------------------------------------------------------------
value label D10
-------------------------------------------------------------------

      values                                    labels
       range:  [1,6]                      string length:  [10,25]
           N:  7                   unique at full length:  yes
        gaps:  no                    unique at length 12:  yes
  missing .*:  1                             null string:  no
                                leading/trailing blanks:  no
                                     numeric -> numeric:  no
  definition
           1   Per nulla soddisfatto
           2   Poco soddisfatto
           3   Soddisfatto
           4   Più che Soddisfatto
           5   Completamente Soddisfatto
           6   Non saprei
          .a   Non risponde

   variables:  d10_1 d10_2 d10_3 d10_4 d10_5 d10_6

no potential problems in dataset C:/Projects/s_cuore/sodd_paz_2008/data/mother.dta
```

`numlabel` aggiunge come prefisso numerico il valore della label

`numlabel` [*lblname-list*] [`, {add|remove}` *numlabel_options*]

numlabel_options:

`add` aggiunge il prefisso numerico alle label
`remove` rimuove il prefisso numerico alle label
`mask(str)` maschera di formattazione per l'aggiunta del valore numerico; di default la maschera è "#."
`force` forza l'aggiunta o la rimozione del valore numerico
`detail` visualizza esempio di label con e senza il prefisso numerico

```
. numlabel D10, add

. tab d10_1, miss

                      Medici |      Freq.     Percent        Cum.
-----------------------------+-----------------------------------
    1. Per nulla soddisfatto |          2        0.14        0.14
         2. Poco soddisfatto |          4        0.29        0.43
              3. Soddisfatto |        208       14.97       15.41
      4. Più che Soddisfatto |        356       25.63       41.04
5. Completamente Soddisfatto |        747       53.78       94.82
               6. Non saprei |          5        0.36       95.18
            .a. Non risponde |         67        4.82      100.00
-----------------------------+-----------------------------------
                       Total |      1,389      100.00

. numlabel D10, remove

. numlabel D10, add mask([#.] )

. tab d10_1, miss

                      Medici |      Freq.     Percent        Cum.
-----------------------------+-----------------------------------
    [1.]Per nulla soddisfatto |          2        0.14        0.14
         [2.]Poco soddisfatto |          4        0.29        0.43
              [3.]Soddisfatto |        208       14.97       15.41
      [4.]Più che Soddisfatto |        356       25.63       41.04
[5.]Completamente Soddisfatto |        747       53.78       94.82
               [6.]Non saprei |          5        0.36       95.18
            [.a.]Non risponde |         67        4.82      100.00
-----------------------------+-----------------------------------
                       Total |      1,389      100.00
```

Utili comandi aggiuntivi sono le `labutil` (`ssc inst labutil`) che consentono tra l'altro di utilizzare una variabile per creare il `label define` per un'altra (`labmask`).

È anche possibile inserire delle note al dataset o a singole variabili con

`notes` [*varname*]: *text*

se non viene specificata nessuna *varname* la nota viene riferita all'intero dataset, altrimenti alla variabile specificata. Poi visualizzarle si usa il comando

`notes` [`list`]

```
note: Valori espressi in Euro
note y1: escluso da capitale finanziario.
note y1: y1 = (yl + yt + ym + yca)
note y2: y2 = (yl + yt + ym + yc)
note yl: yl = (yl1 + yl2)
note yt: yt = (ytp + yta)
note ytp: ytp = (ytp1 + ytp2)

notes

_dta:
  1.  Valori espressi in Euro

y1:
  1.  escluso da capitale finanziario.
  2.  y1 = (yl + yt + ym + yca)

y2:
  1.  y2 = (yl + yt + ym + yc)

yl:
  1.  yl = (yl1 + yl2)

yt:
  1.  yt = (ytp + yta)

ytp:
  1.  ytp = (ytp1 + ytp2)


codebook  y1, all

-------------------------------------------------------------------------------
y1                                                    Reddito disponibile netto
-------------------------------------------------------------------------------

                  type:  numeric (float)

                 range:  [0,418846.53]                units:  .01
         unique values:  2910                     missing .:  6/7147

                  mean:     24801
             std. dev:     18549

          percentiles:        10%       25%       50%       75%       90%
                           8263.31   13427.9   21174.7   31865.4   43950.5

y1:
  1.  escluso da capitale finanziario.
  2.  y1 = (yl + yt + ym + yca)
```

7.2 Controllo delle variabili chiave

Cosa sono le variabili chiave? È quella variabile o quell'insieme di variabili che permettono di identificare in maniera univoca ciascuna osservazione del dataset. Sapere quali sono le variabili chiave è fondamentale per applicare correttamente una serie di comandi (`merge` per esempio) e per interpretare i risultati delle analisi. Stata dispone di un set di istruzioni dedicate alla gestione delle variabili chiave. Il principale è:

`duplicates report` [*varlist*] [*if*] [*in*]

che controlla se le variabili specificate in *varlist* sono delle variabili chiave.

```
. duplicates report  idcode

Duplicates in terms of idcode
--------------------------------------
   copies | observations       surplus
----------+---------------------------
        1 |         2246             0
--------------------------------------
```

in questo caso la variabile `idcode` è variabili chiave perché non ci sono doppioni, mentre nel caso seguente

```
. duplicates report  fam_id

Duplicates in terms of fam_id
--------------------------------------
   copies | observations       surplus
----------+---------------------------
        1 |         6303             0
        2 |        14646          7323
        3 |        18846         12564
        4 |        23956         17967
        5 |         8330          6664
        6 |         2022          1685
        7 |          595           510
        8 |          112            98
        9 |           27            24
       10 |           20            18
       11 |           11            10
       12 |           12            11
--------------------------------------
```

`fam_id` non è variabile chiave perché ci sono osservazioni doppie (14646), triple (18846) e così via.
Se siamo sicuri che ci siano effettivamente delle osservazioni ripetute (il che vuol dire che tutte le variabili dell'osservazione hanno valori uguali e non solo per le variabili chiave), possiamo ricorrere a questo comando per eliminarle:

`duplicates drop` *varlist*[*if*][*in*]`, force`

quindi ritornando all'esempio precedente:

```
duplicates drop  fam_id, force

Duplicates in terms of fam_id
(46874 observations deleted)
```

7.3 Rinominare variabili

Il comando di Stata per cambiare il nome ad una variabile è:

`rename` *old_varname new_varname*

ma molto più potente e ricco di funzioni è il comando aggiuntivo `renvars`[2] nella versione:

[2]Come si fa ad installarlo? Dovreste essere in grado di farlo!

`renvars` [*varlist*]\ *newvarlist* [, `display test`]

che rinomina le variabili della lista prima del simbolo \ con le corrispondenti che lo seguono. Le opzioni

`display` serve per vedere ciascun cambiamento
`test` serve per testare la modifica senza eseguirla

Ma ancora più interessante è la versione:

`renvars` [*varlist*], *transformation_option* [, `display test symbol(`*string*`)`]

con *transformation_option* che possono essere una tra:

`upper` per convertire i nomi in maiuscolo
`lower` per convertire i nomi in minuscolo
`prefix(`*str*`)` per assegnare il prefisso *str* al nome delle variabili
`postfix(`*str*`)` per assegnare *str* come finale del nome delle variabili
`subst(`*str1 str2*`)` per sostituire tutte le ricorrenze *str1* con *str2* (*str2* può anche essere vuoto)
`trim(#)` per prendere solo i primi # caratteri del nome
`trimend(#)` per prendere solo gli ultimi # caratteri del nome

Ecco alcuni esempi con le opzioni appena viste:
Esempi di `rename` *classico*:

```
rename c2  TM1
rename c3  TM2
rename c4  TM3
rename c5  TM4
rename c6  TM5
rename c7  TM6
rename c8  TM7
rename c9  TM8
rename c10 TM9
```

Esempi di `renvars`:

```
. renvars c2-c10  TM1-TM9 , display test;
specification would result in the following renames:
                                c2 -> TM1
                                c3 -> TM2
                                c4 -> TM3
                                c5 -> TM4
                                c6 -> TM5
                                c7 -> TM6
                                c8 -> TM7
                                c9 -> TM8
                               c10 -> TM9

. renvars c2-c10, upper test;
specification would result in the following renames:
                                c2 -> C2
                                c3 -> C3
                                c4 -> C4
                                c5 -> C5
```

```
                                  c6 -> C6
                                  c7 -> C7
                                  c8 -> C8
                                  c9 -> C9
                                 c10 -> C10

. renvars c2-c10, prefix(tax_) test;
specification would result in the following renames:
                                  c2 -> tax_c2
                                  c3 -> tax_c3
                                  c4 -> tax_c4
                                  c5 -> tax_c5
                                  c6 -> tax_c6
                                  c7 -> tax_c7
                                  c8 -> tax_c8
                                  c9 -> tax_c9
                                 c10 -> tax_c10

. renvars c2-c10, postfix(t) test;
specification would result in the following renames:
                                  c2 -> c2t
                                  c3 -> c3t
                                  c4 -> c4t
                                  c5 -> c5t
                                  c6 -> c6t
                                  c7 -> c7t
                                  c8 -> c8t
                                  c9 -> c9t
                                 c10 -> c10t
```

7.4 Ordinare variabili

Per ordinare le osservazioni in base ai valori di una o più variabili si usa il comando

`sort` *varlist*[*in*][`, stable`]

Alternativamente si può usare il più sofisticato

`gsort` [`+|-`] *varname*[[`+|-`] *varname* `...`] [`, generate(`*newvar*`) mfirst`]

il segno `+` (ordinamento crescente) si può omettere per ordinare in senso crescente. Da notare che si può anche combinare un ordinamento in senso crescente per una variabile con l'ordinamento in senso decrescente (`-`) di un'altra[3]. Questo comando funziona anche con variabili stringa.
ESEMPIO opzione `stable`

Invece per spostare le variabili all'inizio della lista si usa:

`order` *varlist*

```
. desc, simple
balance  id       var1

. order id

. desc, simple
id       balance  var1
```

[3]Come vengono ordinati i valori missing?

Per spostare `varname1` al posto di `varname2`:

`move` *varname1 varname2*

```
. desc, simple
id       balance  var1

. move var1 balance

. desc, simple
id       var1     balance
```

Per ordinare in senso alfabetico in base al nome della variabile:

`aorder` [*varlist*]

```
. desc, simple
id       var1     balance

. aorder

. desc, simple
balance  id       var1
```

7.5 Prendere o scartare osservazioni o variabili

La coppia di comandi `keep` e `drop` serve per tenere (`keep`) variabili o osservazioni, oppure a cancellare (`drop`) variabili o osservazioni.
Per tenere variabili, scartando le altre:

`keep` *varlist*

Per tenere osservazioni, scartando le altre:

`keep if` *condizione*

Per eliminare variabili, tenendo le altre:

`drop` *varlist*

Per eliminare osservazioni, tenendo le altre

`drop if` *condizione*

Esiste anche un comando per estrarre da un database un campione casuale di una determinata numerosità. Prima di analizzarlo però vediamo come sia possibile creare "un campione casuale sempre uguale". In genere per produrre un campione casuale si genera un numero casuale detto seme e sulla base di questo si genera l'algoritmo per l'estrazione del campione. Fissando il valore di questo seme, il campione casuale estratto sarà, di conseguenza, sempre uguale. Il comando:

`set seed` #

serve proprio a questo.
Il comando per estrarre il campione casuale è invece:

`sample` # [*if*] [*in*] [`, count by(`*groupvars*`)`]

dove # è la percentuale delle osservazioni del dataset di partenza che deve avere il campione casuale. Se invece vogliamo avere l'estrazione di un determinato numero di osservazioni, occorre usare l'opzione `count` e # sarà il numero di osservazioni da estrarre. Ecco gli esempi per i due casi:

```
. use ita_82-06, clear

. desc, short

Contains data from ita_82-06.dta
  obs:     1,636,402
 vars:            33                          4 Jul 2007 13:19
 size:   121,093,748 (53.8% of memory free)
Sorted by:
     Note:  dataset has changed since last saved

. set seed 74638928

. summ pop_2006

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
    pop_2006 |   1636402    35.90298    219.7405          0      23722

. sample 20
(1309122 observations deleted)

. summ pop_2006

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
    pop_2006 |    327280    35.03108    203.4676          0      23329

. sample 2380, count
(324900 observations deleted)

. summ pop_2006

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
    pop_2006 |      2380    38.21765    181.8105          0       6615
```

7.6 Gestire il formato delle variabili

Stata salva le variabili in diversi formati numerici a seconda delle necessità. L'obiettivo è di minimizzare lo spazio di memoria occupato da ciascuna variabile. Per esempio è inutile sprecare spazio per i decimali per una variabile che può assumere solo valori interi o dedicare 100 caratteri per una variabile stringa quando al massimo arriva a 70. Segue lo schema delle classi di variabili supportate in Stata:

Stringa

`str`# con massimo 244 caratteri di lunghezza

Numeriche

`byte` numeri interi tra -127 e 100;

`int` numeri interi tra -32 740 e 32 740;
`long` numeri interi tra -2 147 483 647 e 2 147 483 620;
`float` numeri reali con 8 cifre decimali di accuratezza compresi tra -1.70141173319*10^38 e 1.70141173319*10^36;
`double` numeri reali con 17 cifre decimali di accuratezza compresi tra -8.9884656743*10^307 e 8.9884656743*10^307.

Passando da `byte` a `int`, a `long` e a `float` la quantità di memoria occupata da una variabile aumenta. Il modo più semplice per ottimizzare lo spazio occupato da ciascuna variabile è il comando `compress`, che abbiamo già visto, altrimenti se vogliamo forzare una certa variabile ad avere un certo formato possiamo usare:

`recast` *type varlist*[`, force`]

dove *type* può assumere i valori `-byte`
`-int`
`-long`
`-float`
`-double`
`-str#`

Discorso diverso è il formato con cui una variabile viene visualizzata e che è indipendente dal formato con cui viene salvata/trattata. In questo caso dobbiamo far ricorso al comando:

`format` *%fmt varlist*

dove *%fmt* può assumere le seguenti forme:

```
    Numerical
      %fmt            Description              Example
------------------------------------------------------
right justified
      %#.#g           general                  %9.0g
      %#.#f           fixed                    %9.2f
      %#.#e           exponential              %10.7e
      %21x            hexadecimal              %21x
      %16H            binary, hilo             %16H
      %16L            binary, lohi             %16L
      %8H             binary, hilo             %8H
      %8L             binary, lohi             %8L

right justified with commas
      %#.#gc          general                  %9.0gc
```

```
      %#.#fc          fixed                   %9.2fc

right justified with leading zeros
      %0#.#f          fixed                   %09.2f

left justified
      %-#.#g          general                 %-9.0g
      %-#.#f          fixed                   %-9.2f
      %-#.#e          exponential             %-10.7e

left justified with commas
      %-#.#gc         general                 %-9.0gc
      %-#.#fc         fixed                   %-9.2fc
-------------------------------------------------------
You may substitute comma (,) for period (.) in any of
the above formats to make comma the decimal point.  In
%9,2fc, 1000.03 is 1.000,03.  Or you can set dp comma.


      date
      %fmt            Description             Example
-------------------------------------------------------
right justified
      %tc             date/time               %tc
      %tC             date/time               %tC
      %td             date                    %td
      %tw             week                    %tw
      %tm             month                   %tm
      %tq             quarter                 %tq
      %th             halfyear                %th
      %ty             year                    %ty
      %tg             generic                 %tg

left justified
      %-tc            date/time               %-tc
      %-tC            date/time               %-tC
      %-td            date                    %-td
      etc.
-------------------------------------------------------
There are lots of variations allowed.  See Formatting
date and time values in [D] dates and times.
```

```
     string
      %fmt            Description              Example
------------------------------------------------------------
right justified
      %#s             string                   %15s

left justified
      %-#s            string                   %-20s

centered
      %~#s            string                   %~12s
------------------------------------------------------------
The centered format is for use with display only.
```

Aggiungo che si può modificare come Stata rappresenta il simbolo decimale con il comando

```
set dp {comma|period} [, permanently]
```

```
. di 123.79
123.79

. set dp comma

. di 123.79
123456,79
```

Capitolo 8

Creare Variabili

8.1 Il comando `generate`

Il comando più usato per creare nuove variabili è:

`generate` [`type`] *newvarname*=*exp*[*if*][*in*]

che funziona anche con variabili stringa. Per esempio

```
gen nome_compl = nome + " " + cognome
```

unisce il dato stringa contenuto nella variabile `nome` con uno spazio e con il contenuto della variabile stringa `cognome`.
Oltre a creare variabili attraverso una espressione algebrica si può ricorrere ad una serie di funzioni già predisposte. Segue una panoramica divisa per tipo delle principali

8.1.1 Funzioni matematiche

`abs(x)` ritorna il valore assoluto di ciascun valore della variabile x:

```
. gen var2=abs(var1)

. clist var1 var2

         var1       var2
 1.       -.4         .4
 2.        .8         .8
 3.      -1.1        1.1
 4.       1.3        1.3
 5.      -1.6        1.6
 6.       1.9        1.9
```

`ceil(x)` arrotonda all'intero superiore
`floor(x)` arrotonda all'intero inferiore
`int(x)` tiene l'intero del numero
`round(x,#)` arrotonda in base a #: 1 all'intero, 0.1 al primo decimale, 0.01 al centesimo
...

```
. gen var2=ceil(var1)
```

```
. gen var3=floor(var1)

. genvar4=int(var1)

. gen var4=int(var1)

. gen var5=round(var1,0.1)

. clist

         var1       var2       var3       var4       var5
  1.      -.4          0         -1          0       -.4
  2.     .8123         1          0          0        .8
  3.     -1.1         -1         -2         -1       -1.1
  4.      1.3          2          1          1        1.3
  5.  -1.6876         -1         -2         -1       -1.7
  6.      1.9          2          1          1        1.9
```

`exp(x)` ritorna il valore esponenziale di `x`. È la funzione inversa di `ln(x)`.
`ln(x)` ritorna il logaritmo naturale di `x`

```
. gen var2=exp( var1)

. gen var3=ln(var2)

. clist

         var1       var2       var3
  1.      -.4     .67032       -.4
  2.     .8123   2.253084      .8123
  3.     -1.1   .3328711       -1.1
  4.      1.3   3.669297        1.3
  5.  -1.6876   .1849629    -1.6876
  6.      1.9   6.685894        1.9
```

`log10(x)` ritorna il logaritmo in base 10 di `x`

```
. gen var2=log10(var1)
(3 missing values generated)

. clist

         var1       var2
  1.      -.4          .
  2.     .8123  -.0902835
  3.     -1.1          .
  4.      1.3   .1139433
  5.  -1.6876          .
  6.      1.9   .2787536
```

`max(x1,x2,...,xn)` ritorna il valore massimo tra i valori di `x1, x2, ..., xn`. I valori missing vengono ignorati.

```
. gen var4=max(var1,var2,var3)

. clist

         var1       var2      var3       var4
  1.      -.4          .         2          2
  2.     .8123  -.0902835         0      .8123
  3.     -1.1          .        -3       -1.1
```

```
  4.       1.3   .1139433      1.31       1.31
  5.   -1.6876          .     -2.01    -1.6876
  6.       1.9   .2787536       1.5        1.9
```

`min(x1,x2,...,xn)` ritorna il valore minimo tra i valori di `x1, x2, ..., xn`. I valori missing vengono ignorati.

```
. gen var4=min(var1,var2,var3)

. clist

         var1       var2      var3       var4
  1.      -.4          .         2       -.4
  2.     .8123  -.0902835         0  -.0902835
  3.     -1.1          .        -3         -3
  4.       1.3   .1139433      1.31    .1139433
  5.   -1.6876          .     -2.01      -2.01
  6.       1.9   .2787536       1.5    .2787536
```

`sum(x)` ritorna la somma incrementale dei valori di `x`

```
. gen var3=sum(var1)

. gen var4=sum(var2)

. clist

         var1       var2       var3       var4
  1.      -.4          .       -.4          0
  2.     .8123  -.0902835      .4123  -.0902835
  3.     -1.1          .     -.6877  -.0902835
  4.       1.3   .1139433   .6122999   .0236598
  5.   -1.6876          .    -1.0753   .0236598
  6.       1.9   .2787536   .8246999   .3024134
```

8.1.2 Funzioni di distribuzione di probabilità e funzioni di densità

`betaden(a,b,x)` ritorna la pdf di una distribuzione beta.

```
. gen var2=betaden(0.1,0.2,var1)

. clist

         var1       var2
  1.        .4   .2351207
  2.     .8123   .3148833
  3.        .1    .591932
  4.        .3    .269264
  5.     .6876   .2433855
  6.        .9   .4751683
```

`binomial(n,k,p)` ritorna le probabilità di una binomiale.

```
. gen var3=binomial(var1,var2,var0)
 (1 missing value generated)

 . clist

          var0       var1       var2       var3
  1.       .01         .4   .2351207          1
  2.       2.2      .8123   .3148833          .
```

```
  3.        .4        3.1   -.591932          0
  4.        .6         .3    .269264          1
  5.        .8      .6876   .2433855          1
  6.       .99         .9  -.4751683          0
```

`chi2(n,x)` ritorna la cumulata di una distribuzione chi-quadrato con n gradi di libertà.

```
. gen var2 = chi2(5,var1)

 . clist

         var1       var2
 1.        .4   .0046704
 2.     .8123   .0237578
 3.       3.1    .315428
 4.        .3   .0023569
 5.     .6876   .0163577
 6.        .9   .0297784
```

`F(n1,n2,f)` ritorna la cumulata di una distribuzione F.

```
. gen var2=F(3,4,var1)

. clist

         var1       var2
 1.        .4    .238771
 2.     .8123   .4500666
 3.       3.1   .8485015
 4.        .3   .1751056
 5.     .6876   .3948669
 6.        .9   .4849114
```

`Fden(n1,n2,f)` ritorna la pdf di una distribuzione F.

```
. gen var2=Fden(3,4,var1)

. clist

         var1       var2
 1.        .4   .6149665
 2.     .8123   .4152625
 3.       3.1   .0639784
 4.        .3   .6557035
 5.     .6876   .4711343
 6.        .9   .3799204
```

`gammaden(a,b,g,x)` ritorna la pdf di una distribuzione Gamma.

```
. gen var2=gammaden(1,2,0.25,var1)

. clist

         var1       var2
 1.        .4   .4638717
 2.     .8123   .3774575
 3.       3.1   .1202542
 4.        .3    .487655
 5.     .6876   .4017412
 6.        .9   .3612637
```

`normalden(x,m,s)` ritorna la normale con media m e deviazione standard s.

```
. gen var2=normalden(var1,2,1.5)

. clist

         var1       var2
 1.        .4   .1505752
 2.     .8123   .1943922
 3.       3.1   .2032553
 4.        .3   .1399282
 5.     .6876   .1813806
 6.        .9   .2032553
```

8.1.3 Funzioni di generazione di numeri random

`uniform()` ritorna numeri random uniformemente distribuiti nell'intervallo [0,1)

```
. gen var1=uniform()

. clist

          var1
 1.   .1369841
 2.   .6432207
 3.   .5578017
 4.   .6047949
 5.    .684176
 6.   .1086679
```

`invnormal(uniform())` ritorna numeri random normalmente distribuiti con media 0 e deviazione standard 1.

```
. gen var2=invnormal(uniform())

. clist

         var2
 1.  .3014338
 2. -1.545905
 3.  .1389086
 4.  1.133268
 5.  -.658371
 6. -1.700496
```

8.1.4 Funzioni stringa

`abbrev(s,n)` ritorna la stringa `s` abbreviata a `n` caratteri.

```
. gen var2=abbrev(var1,5)

. clist

              var1       var2
 1.         io sono      io ~o
 2.          tu sei      tu ~i
 3.         egli e´      egl~´
 4.       noi siamo      noi~o
 5.       voi siete      voi~e
 6.         egli e´      egl~´
```

`length(s)` ritorna la lunghezza della stringa `s`.

```
var1       var2
  1.         io sono          7
  2.          tu sei          6
  3.         egli e´          7
  4.       noi siamo          9
  5.       voi siete          9
  6.         egli e´          7
```

`lower(s)`, `upper(s)` ritornano `s` in lettere minuscole o maiuscole.

```
gen var2=lower(var1)

. gen var3=upper(var1)

. clist

               var1       var2       var3
  1.         Io Sono    io sono    IO SONO
  2.          Tu Sei     tu sei     TU SEI
  3.         Egli E´    egli e´    EGLI E´
  4.       Noi Siamo  noi siamo  NOI SIAMO
  5.       Voi Siete  voi siete  VOI SIETE
  6.         Egli E´    egli e´    EGLI E´
```

`ltrim(s)` `rtrim(s)`, `trim(s)` ritornano `s` senza spazi iniziali, senza spazi finali o senza spazi all'inizio e alla fine

```
. gen var2=ltrim(var1)

. gen var3=rtrim(var1)

. gen var4=trim(var1)

. clist

              var1          var2          var3       var4
  1.      Io Sono       Io Sono          Io Sono    Io Sono
  2.       Tu Sei        Tu Sei           Tu Sei     Tu Sei
  3.      Egli E´       Egli E´          Egli E´    Egli E´
  4.    Noi Siamo     Noi Siamo        Noi Siamo  Noi Siamo
  5.    Voi Siete     Voi Siete        Voi Siete  Voi Siete
  6.      Egli E´       Egli E´          Egli E´    Egli E´
```

`reverse(s)` ritorna `s` invertita.

```
. gen var2=reverse(var1)

. clist

               var1       var2
  1.         Io Sono    onoS oI
  2.          Tu Sei     ieS uT
  3.         Egli E´    ´E ilgE
  4.       Noi Siamo  omaiS ioN
  5.       Voi Siete  eteiS ioV
  6.         Egli E´    ´E ilgE
```

`strmatch(s1,s2)` ritorna 1 se in `s1` viene trovato `s2`, altrimenti ritorna 0.

```
. gen var2=strmatch( var1,"*E´*")

. clist
```

```
              var1       var2
 1.         Io Sono          0
 2.          Tu Sei          0
 3.         Egli E´          1
 4.       Noi Siamo          0
 5.       Voi Siete          0
 6.         Egli E´          1
```

`subinstr(s1,s2,s3,n)` in `s1` viene sostituito `s2` con `s3` per le prime `n` volte che `s2` viene trovato. Con il simbolo '.' si sostituiscono tutte le occorrenze di `s2` con `s`.

```
. gen var2=subinstr(var1,"o","0",.)

. clist

              var1       var2
 1.         Io Sono    I0 S0n0
 2.          Tu Sei     Tu Sei
 3.         Egli E´    Egli E´
 4.       Noi Siamo  N0i Siam0
 5.       Voi Siete  V0i Siete
 6.         Egli E´    Egli E´
```

`subinword(s1,s2,s3,n)` in `s1` viene sostituita la parola `s2` con la parola s3 per le prime n volte che viene trovata. Una parola viene identificata con uno spazio bianco. con il simbolo '.' si sostituiscono tutte le occorenze di `s2` con `s3`.

```
. gen var2=subinword(var1,"E´","&",.)

. clist

              var1       var2
 1.         Io Sono    Io Sono
 2.          Tu Sei     Tu Sei
 3.         Egli E´     Egli &
 4.       Noi Siamo  Noi Siamo
 5.       Voi Siete  Voi Siete
 6.         Egli E´     Egli &
```

`substr(s,n1,n2)` ritorna la stringa di `s` che parte dalla posizione `n1` fino alla posizione `n2`. Se `n1` è un valore negativo vuol dire che si parte a contare dalla fine della stringa `s`.

```
. gen var2=substr(var1,3,3)

. clist

              var1       var2
 1.         Io Sono         So
 2.          Tu Sei         Se
 3.         Egli E´        li
 4.       Noi Siamo        i S
 5.       Voi Siete        i S
 6.         Egli E´        li
```

`word(s,n)` ritorna la n-esima parola all'interno della stringa `s`.

```
. gen var2=word(var1,1)

. clist

              var1       var2
 1.         Io Sono         Io
 2.          Tu Sei         Tu
```

```
  3.         Egli E´       Egli
  4.       Noi Siamo        Noi
  5.       Voi Siete        Voi
  6.         Egli E´       Egli
```

`wordcount(s)` conta il numero di parole all'interno della stringa `s`.

```
. gen var2=wordcount(var1)

. clist

               var1       var2
  1.         Io Sono          2
  2.          Tu Sei          2
  3.         Egli E´          2
  4.       Noi Siamo          2
  5.       Voi Siete          2
  6.         Egli E´          2
```

8.1.5 Funzioni di programmazione

`inlist(z,a,b,...)` ritorna 1 se il contenuto di `z` è presente in uno degli argomenti successivi, altrimenti ritorna 0. Gli argomenti possono essere numerici o stringa. Se sono numerici possono essere in un numero tra 2 e 255, se sono stringa tra 2 e 10.

```
. gen cod_com=.;
(395 missing values generated)

. replace cod_com=2 if inlist(q03,"affi");
(0 real changes made)

. replace cod_com=2 if inlist(q03,"gazzo v.se");
(0 real changes made)

. replace cod_com=6 if inlist(q03,"bardolino","cisano bardolino");
(2 real changes made)
```

(*output omitted*)

```
. replace cod_com=96 if inlist(q03,"alpo villafranca","s.ambrogio di villa vr",
> "villafranca","villafranca di vr");
(2 real changes made)

. replace cod_com=97 if inlist(q03,"zevio","zevio vr");
(2 real changes made)
```

`cond(x,a,b,c)` o `cond(x,a,b)` ritorna `a` se `x` è vero, `b` se `x` è falso e `c` se `x` è missing. Se `c` non è specificato allora in `x` viene messo valore missing.

```
. clist

        var1      var2
  1.       1         1
  2.       1         2
  3.       2         1
  4.       2         2
  5.       2         3
  6.       1         3
  7.       1         2
```

```
. gen ris = cond(var1==1 & var2==2,"YES","NO")

. tab ris

        ris |      Freq.     Percent        Cum.
------------+-----------------------------------
         NO |          5       71.43       71.43
        YES |          2       28.57      100.00
------------+-----------------------------------
      Total |          7      100.00

. clist

         var1      var2        ris
  1.        1         1         NO
  2.        1         2        YES
  3.        2         1         NO
  4.        2         2         NO
  5.        2         3         NO
  6.        1         3         NO
  7.        1         2        YES
```

`inrange(z,a,b)` ritorna 1 se `a < z < b`, altrimenti ritorna 0.

```
. clist

         var1
  1.        .
  2.        0
  3.        1
  4.        2
  5.        3
  6.        4
  7.        5
  8.        6

. gen check=inrange(var1,2,5)

. clist

         var1      check
  1.        .          0
  2.        0          0
  3.        1          0
  4.        2          1
  5.        3          1
  6.        4          1
  7.        5          1
  8.        6          0
```

8.1.6 Funzioni data

`mdy(`M`,`D`,`Y`)` ritorna la data in formato codificato a partire dai valori di M (mese), D (giorno) e Y (anno). Il valore codificato di una data è un intero che identifica il giorno a partire dalla data del 01 gennaio 1960 (01 gennaio 1960=0, 01 gennaio 1960=1 e così via).

```
. summ month day year
```

```
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
       month |      1652    9.214891    3.134451          1         12
         day |      1652    16.77421    8.470449          1         31
        year |      1651    2005.998    .0491769       2005       2006

. clist month day year in 1/7

        month        day       year
  1.        .          .          .
  2.        9          8       2006
  3.       10         29       2006
  4.       11         28       2006
  5.       12          3       2006
  6.       10         10       2006
  7.       10         26       2006

. gen comp = mdy(month,day,year)
(690 missing values generated)

. clist month day year comp in 1/7

        month        day       year       comp
  1.        .          .          .          .
  2.        9          8       2006      17052
  3.       10         29       2006      17103
  4.       11         28       2006      17133
  5.       12          3       2006      17138
  6.       10         10       2006      17084
  7.       10         26       2006      17100

. format comp %dd/N/CY

. clist month day year comp in 1/7

        month        day       year        comp
  1.        .          .          .           .
  2.        9          8       2006   8/09/2006
  3.       10         29       2006  29/10/2006
  4.       11         28       2006  28/11/2006
  5.       12          3       2006   3/12/2006
  6.       10         10       2006  10/10/2006
  7.       10         26       2006  26/10/2006
```

8.1.7 Funzioni per serie temporali

...

8.1.8 Funzioni matriciali

`diag(v)` ritorna la matrice quadrata diagonale costruita a partire dal vettore `v`.

```
. matrix list Y

Y[1,4]
    c1  c2  c3  c4
r1   1   2   3   4

. matrix Ydiag=diag(Y)
```

```
. matrix list Ydiag

symmetric Ydiag[4,4]
    c1  c2  c3  c4
c1   1
c2   0   2
c3   0   0   3
c4   0   0   0   4

. matrix list X

X[4,1]
    c1
r1   1
r2   2
r3   3
r4   4

. matrix Xdiag=diag(X)

. matrix list Xdiag

symmetric Xdiag[4,4]
    r1  r2  r3  r4
r1   1
r2   0   2
r3   0   0   3
r4   0   0   0   4
```

`inv(M)` ritorna l'inversa della matrice `M`.

```
. matrix list X

X[4,4]
      c1    c2    c3    c4
r1    .1    .2    .3    .4
r2    .5    .6    .7    .8
r3    .9  .094   .11   .12
r4   .13   .14   .15   .16

. matrix Xinv = inv(X)

. matrix list Xinv

Xinv[4,4]
            r1          r2          r3          r4
c1   2.697e+13  -4.046e+13    1.233951   1.349e+14
c2   3.641e+15  -5.462e+15  -.08328654   1.821e+16
c3  -7.364e+15   1.105e+16  -3.5352798  -3.682e+16
c4   3.695e+15  -5.543e+15   2.3846154   1.848e+16
```

`colsof(M)` ritorna uno scalare con il numero di righe della matrice `M`; `rowsof(M)` ritorna uno scalare con il numero di colonne della matrice `M`.

```
. matrix list Y

Y[2,4]
    c1  c2  c3  c4
r1   1   2   3   4
r2   5   6   7   8
```

```
. local Ycol = colsof(Y)

. local Yrow = rowsof(Y)

. di "La matrice Y ha `Ycol´ colonne e `Yrow´ righe!"
La matrice Y ha 4 colonne e 2 righe!
```

`nullmat()` serve per eseguire operazioni di uinione per riga o per colonna di matrici. La sua particolarità risiede nel fatto che riesce a lavorare anche con matrici non esistenti. La funzione `nullmat()` informa Stata che se la matrice non esiste deve essere creata con l'elemento indicato successivamente alla funzione. Nell'esempio che segue la matrice `G` non esiste; a questa matrice inesistente viene aggiunto lo scalare `1`, generando in tal modo la matrice `G`. La funzione `nullmat()` funziona sia con l'operatore di unione `,` che con l'operatore `\`.
`nullmat()`

```
. matrix list G
matrix G not found
r(111);

. matrix G = (nullmat(G),1)

. matrix list G

symmetric G[1,1]
    c1
r1   1

. matrix G = (nullmat(G),2)

. matrix list G

G[1,2]
    c1  c2
r1   1   2

. matrix G = (nullmat(G),3)

. matrix list G

G[1,3]
    c1  c2  c3
r1   1   2   3

. matrix G = (nullmat(G),4)

. matrix list G

G[1,4]
    c1  c2  c3  c4
r1   1   2   3   4
```

`trace(M)` ritorna la traccia della matrice `M`.

```
. matrix list X

X[4,4]
      c1    c2    c3    c4
r1    .1    .2    .3    .4
```

```
r2    .5    .6    .7    .8
r3    .9  .094   .11   .12
r4   .13   .14   .15   .16

. matrix Xtrace=trace(X)

. matrix list Xtrace

symmetric Xtrace[1,1]
     c1
r1  .97
```

Tutte le funzioni viste precedentemente funzionano anche con il comando `replace`.

8.2 Lavorare con osservazioni indicizzate

Per indicizzare le osservazioni Stata usa la notazione
varname`[_n]` per indicare la corrente osservazione di *varname*
varname`[_N]` per indicare l'ultima osservazione di *varname*
Per esempio il comando

```
gen y = x[_n]
```

genera una variabile `y` uguale alla corrispondente osservazione di `x`, ovvero è equivalente a scrivere:

```
gen y = x
```

Se invece eseguo:

```
gen y = x[_n-1]
```

la variabile `y` assume il valore della osservazione precedente di `x`. Conseguentemente con

```
gen y = x[_n+1]
```

la variabile `y` assume il valore della osservazione successiva di `x`. Con

```
gen y = x[_N]
```

genero una variabile che assume l'ultimo valore di `x`, ovvero una costante

```
. gen y1=x[_n]

. gen y2 = x[_n-1]
(1 missing value generated)

. gen y3 = x[_n+1]
(1 missing value generated)

. gen y4 = x[_N]

. clist

          x        y1        y2        y3        y4
  1.      1         1         .         2         6
  2.      2         2         1         3         6
  3.      3         3         2         4         6
```

```
  4.       4         4         3         5         6
  5.       5         5         4         6         6
  6.       6         6         5         .         6
```

Utile per creare velocemente una variabile chiave o indice è il comando

```
gen ID = _n
```

che crea una successione con passo 1 dalla prima all'ultima osservazione, secondo l'ordinamento corrente delle osservazioni

Infine con la costruzione:

```
gen y = x[_N-_n+1]
```

genero `y` in ordine inverso di `x`.

```
. gen ID = _n

. gen y = ID[_N-_n+1]

. clist nquest nord eta ID  y

          nquest          nord       eta          ID          y
  1.          34             1        59           1         10
  2.          34             2        58           2          9
  3.          34             3        31           3          8
  4.          34             4        29           4          7
  5.         173             1        54           5          6
  6.         173             2        52           6          5
  7.         173             3        27           7          4
  8.         173             4        24           8          3
  9.         375             1        77           9          2
 10.         375             2        76          10          1
```

Naturalmente possiamo ricorrere anche al prefisso `by`. Per esempio

```
sort  fam_id age
bysort fam_id: gen old = age[_N]
```

genera una variabile con l'età più alta per ciascun `fam_id`

```
. sort  nquest eta

. bysort nquest: gen old = eta[_N]

. clist nquest eta old

          nquest       eta        old
  1.          34        29         59
  2.          34        31         59
  3.          34        58         59
  4.          34        59         59
  5.         173        24         54
  6.         173        27         54
  7.         173        52         54
  8.         173        54         54
  9.         375        76         77
 10.         375        77         77
```

8.3 Estensione del comando generate

Il comando generate prevede una versione potenziata (egen) che va usata solo in abbinamento con una serie di funzioni specificatamente previste. Utenti di Stata hanno ulteriormente potenziato questo comando creando numerose funzioni aggiuntive. Per farsi un'idea si veda egenodd oppure egenmore con

```
ssc desc egenodd
ssc desc egenmore
```

Ritornando al comando egen, la sua sintassi generale è:

egen [type] *newvarname* = *fcn(arguments)* [*if*] [*in*] [, *options*]

dove le principali funzioni *fcn(arguments)* sono:

anycount(varlist), values(integer numlist) ritorna il numero di volte che i valori in values() sono presenti nelle variabili indicate in varlist.

```
. clist  q1a_a q2a_a q2b_a q2c_a q3a_a

      q1a_a     q2a_a     q2b_a     q2c_a     q3a_a
 1.       0         1         1         1         4
 2.       0         1         1         0         1
 3.       0         1         1         1         0
 4.       0         1         1         1         2
 5.       0         1         1         1         4
 6.       0         0         0         0         1
 7.       0         1         1         1         2
 8.       0         1         0         1         1
 9.       0         0         1         0         1
10.       0         1         1         0         0

. egen es0 = anycount(q1a_a q2a_a q2b_a q2c_a q3a_a), values(0)

. egen es1 = anycount(q1a_a q2a_a q2b_a q2c_a q3a_a), values(2/4)

. clist  q1a_a q2a_a q2b_a q2c_a q3a_a es es1

      q1a_a     q2a_a     q2b_a     q2c_a     q3a_a       es0       es1
 1.       0         1         1         1         4         1         1
 2.       0         1         1         0         1         2         0
 3.       0         1         1         1         0         2         0
 4.       0         1         1         1         2         1         1
 5.       0         1         1         1         4         1         1
 6.       0         0         0         0         1         4         0
 7.       0         1         1         1         2         1         1
 8.       0         1         0         1         1         2         0
 9.       0         0         1         0         1         3         0
10.       0         1         1         0         0         3         0
```

cut(*varname*), at(#,#,ldots,#) crea una nuova variabile codificata in base agli intervalli identificati da at() in *varname*

```
. egen cat_age=cut(age), at(17,19,21,23,25)
(109938 missing values generated)

. tab age cat_age

           |                  cat_age
   Age ind |        17         19         21         23 |     Total
-----------+--------------------------------------------+----------
        17 |     2,044          0          0          0 |     2,044
```

```
          18 |     2,199          0          0          0 |     2,199
          19 |         0      2,270          0          0 |     2,270
          20 |         0      2,388          0          0 |     2,388
          21 |         0          0      2,477          0 |     2,477
          22 |         0          0      2,546          0 |     2,546
          23 |         0          0          0      2,593 |     2,593
          24 |         0          0          0      2,696 |     2,696
-----------+--------------------------------------------+----------
       Total |     4,243      4,658      5,023      5,289 |    19,213
```

`max(`*exp*`)` ritorna una variabile (costante) con il valore massimo assunto dalle variabili elencate in *exp*.

```
. clist foreign price

     foreign     price
 1.  Foreign     4,499
 2.  Foreign     6,229
 3. Domestic     4,934
 4. Domestic     3,667
 5. Domestic     4,172
 6.  Foreign     8,129
 7.  Foreign     4,589
 8. Domestic     3,984
 9.  Foreign     5,079
10. Domestic     6,486

. egen es = max(price), by(foreign)

. clist foreign price es

     foreign     price        es
 1.  Foreign     4,499      8129
 2.  Foreign     6,229      8129
 3. Domestic     4,934      6486
 4. Domestic     3,667      6486
 5. Domestic     4,172      6486
 6.  Foreign     8,129      8129
 7.  Foreign     4,589      8129
 8. Domestic     3,984      6486
 9.  Foreign     5,079      8129
10. Domestic     6,486      6486
```

`mean(`*exp*`)` ritorna una variabile (costante) con il valore medio delle variabili elencate in *exp*.

```
. clist  foreign price

     foreign     price
 1.  Foreign     4,499
 2.  Foreign     6,229
 3. Domestic     4,934
 4. Domestic     3,667
 5. Domestic     4,172
 6.  Foreign     8,129
 7.  Foreign     4,589
 8. Domestic     3,984
 9.  Foreign     5,079
10. Domestic     6,486

. egen es0=mean(price)

. egen es1=mean(price), by(foreign)

. clist  foreign price es?
```

```
     foreign     price        es0        es1
  1.  Foreign     4,499     5176.8       5705
  2.  Foreign     6,229     5176.8       5705
  3. Domestic     4,934     5176.8     4648.6
  4. Domestic     3,667     5176.8     4648.6
  5. Domestic     4,172     5176.8     4648.6
  6.  Foreign     8,129     5176.8       5705
  7.  Foreign     4,589     5176.8       5705
  8. Domestic     3,984     5176.8     4648.6
  9.  Foreign     5,079     5176.8       5705
 10. Domestic     6,486     5176.8     4648.6
```

`median(`*exp*`)` ritorna una variabile (costante) con il valore mediano delle variabili elencate in *exp*.

```
. clist  q3?_a

       q3a_a     q3b_a     q3c_a     q3d_a
  1.       4         1         0         0
  2.       1         0         0         0
  3.       0         0         0         0
  4.       2         0         0         0
  5.       4         4         3         1
  6.       1         0         0         0
  7.       2         2         0         0
  8.       1         0         0         0
  9.       1         0         0         0
 10.       0         1         0         0

. egen es0=median(q3a_a)

. egen es2=median(q3a_a-q3d_a)

. clist  q3?_a es?

       q3a_a     q3b_a     q3c_a     q3d_a       es1       es0       es2
  1.       4         1         0         0         1         1         1
  2.       1         0         0         0         0         1         1
  3.       0         0         0         0         0         1         1
  4.       2         0         0         0         1         1         1
  5.       4         4         3         1         1         1         1
  6.       1         0         0         0         0         1         1
  7.       2         2         0         0         1         1         1
  8.       1         0         0         0         0         1         1
  9.       1         0         0         0         0         1         1
 10.       0         1         0         0         0         1         1
```

`min(`*exp*`)` ritorna una variabile (costante) con il valore minimo delle variabili elencate in *exp*.

```
. clist foreign length
foreign length 1. Foreign 149 2. Foreign 170 3. Domestic 198 4.
Domestic 179 5. Domestic 179 6. Foreign 184 7. Foreign 165 8. Domestic
163 9. Foreign 170 10. Domestic 182
. egen es0=min(length)
. egen es1=min(length), by( foreign)
. clist foreign length es?
foreign length es0 es1 1. Foreign 149 149 149 2. Foreign 170 149 149 3.
Domestic 198 149 163 4. Domestic 179 149 163 5. Domestic 179 149 163 6.
```

```
Foreign 184 149 149 7.  Foreign 165 149 149 8.  Domestic 163 149 163 9.
Foreign 170 149 149 10.  Domestic 182 149 163
```

`mode(`*varname*`) [, minmode maxmode nummode(integer) missing]` ritorna una variabile (costante) con il valore della moda delle variabili elencate in *varname*. Se esistono più valori modali, con l'opzione `minmode` si sceglie il minore, con `maxmode` il maggiore, con `nummode(integer)` la moda n-esima espressa da `integer`.

`rowmax(`*varlist*`)` ritorna il valore massimo per ciascuna osservazione tra i valori delle variabili elencate in *varlist*.

`rowmean(`*varlist*`)` ritorna il valore medio per ciascuna osservazione dei valori delle variabili elencate in *varlist*.

`rowmin(`*varlist*`)` ritorna il valore minimo per ciascuna osservazione tra i valori delle variabili elencate in *varlist*.

`rowmiss(`*varlist*`)` ritorna il numero di valori missing per ciascuna osservazione tra i valori delle variabili elencate in *varlist*.

`rownonmiss(`*varlist*`)` ritorna il numero di valori non missing per ciascuna osservazione tra i valori delle variabili elencate in *varlist*.

`rowsd(`*varlist*`)` ritorna la deviazione standard per ciascuna osservazione dei i valori delle variabili elencate in *varlist*.

`sum(`*exp*`)` conta quante osservazioni rispondono al criterio *exp*.

8.4 Sostituire valori in una variabile

Il comando principale per sostituire dei valori secondo una certa funzione è

`replace` *oldvar* `=`*exp*[*if*][*in*]

che può essere usato anche con variabili stringa avendo l'accortezza di racchiudere il valore da sostituire tra virgolette:

```
replace str_var = "stringa" if....
```

Se i valori da considerare sono molti e sono relativi ad una stessa variabile, anziché ricorrere ad una lunga serie di `replace` condizionati, è possibile usare la funzione `inlist` che abbiamo visto tra le funzioni di programmazione del comando `generate`

```
. gen macro1=1 if inlist(regione,1,2,3,7,4,5,6,8);
(43456 missing values generated)

. replace macro1=2 if inlist(regione,9,10,11,12);
(13891 real changes made)

. replace macro1=3 if inlist(regione,13,14,15,16,17,18,19,20);
(29565 real changes made)

. tab regione macro1, miss nolab;

           |              macro1
   Regione |         1          2          3 |     Total
-----------+---------------------------------+----------
```

```
          1 |     4,714          0          0 |     4,714
          2 |     1,526          0          0 |     1,526
          3 |     8,166          0          0 |     8,166
          4 |     3,705          0          0 |     3,705
          5 |     4,994          0          0 |     4,994
          6 |     2,029          0          0 |     2,029
          7 |     2,322          0          0 |     2,322
          8 |     3,968          0          0 |     3,968
          9 |         0      4,236          0 |     4,236
         10 |         0      2,265          0 |     2,265
         11 |         0      2,931          0 |     2,931
         12 |         0      4,459          0 |     4,459
         13 |         0          0      2,653 |     2,653
         14 |         0          0      1,972 |     1,972
         15 |         0          0      6,018 |     6,018
         16 |         0          0      4,963 |     4,963
         17 |         0          0      2,225 |     2,225
         18 |         0          0      3,454 |     3,454
         19 |         0          0      5,426 |     5,426
         20 |         0          0      2,854 |     2,854
------------+---------------------------------+----------
      Total |    31,424     13,891     29,565 |    74,880
```

Altro importante comando è `recode` che consente di ricodificare i valori di una variabile (o più variabili) secondo certi criteri (*erule*) e che prevede numerose possibilità:

`recode` *varlist*`(`*erule*`)` [`(`*erule*`)` ...] [*if*] [*in*] [`,` *options*]

Almeno una regola di ricodifica deve essere definita e può assumere le seguenti forme

Regola	***Esempio***	***Esito***
(# = #)	(8 = 4)	Trasforma tutti i valori 8 in 4
$\#_1$ $\#_2$ = #)	(8 7 = 4)	Trasforma tutti i valori 8 e 7 in 4
$\#_1$ / $\#_2$ = #)	(1/8=4)	Trasforma tutti i valori da 1 a 8 (compresi) in 4
nonmissing = #	(nonmissing = 4)	Trasforma tutti i valori non missing in 4
missing = #	(missing=9)	Trasforma tutti i valori missing in 9
#/max	5/max=5	Trasforma tutti i valori superiori a 5 in 5
min/#	min/8=8	Trasforma tutti i valori inferiori a 8 in 8

Le variabili a cui applicare la trasformazione possono essere più di una. Se non viene specificata nessuna opzione del tipo `generate` o `prefix`, la sostituzione avviene direttamente nelle variabili di *varlist*. Se invece si vogliono creare nuove variabili che contengano la ricodifica si ricorrerà all'opzione

`generate(`*newvar*`)` quando si tratta di una sola variabile

`prefix(`*string*`)` quando si tratta di più variabili che prenderanno lo stesso nome delle variabili specificate in *varlist* con l'aggiunta del prefisso specificato in *string*

Per esempio:

```
recode var1 var2 var3 (1/5=1) (6 8=2) (7=3) (.=9), prefix(rec_)
```

nelle variabili `var1`, `var2` e `var3` ricodifica i valori da 1 a 5 in 1, i 6 e gli 8 in 2, il 7 in 3 e tutti i valori missing in 9, creando tre nuove variabili (`rec_var1`, `rec_var2` e `rec_var3`) con le ricodifiche

```
. recode regione (min/8=1) (9/12=2) (13/max=5), gen(macro2);
(70166 differences between regione and macro2)

. tab regione macro2, miss nolab;

           |   RECODE of regione (Regione)
   Regione |         1          2          5 |     Total
-----------+---------------------------------+----------
         1 |     4,714          0          0 |     4,714
         2 |     1,526          0          0 |     1,526
         3 |     8,166          0          0 |     8,166
         4 |     3,705          0          0 |     3,705
         5 |     4,994          0          0 |     4,994
         6 |     2,029          0          0 |     2,029
         7 |     2,322          0          0 |     2,322
         8 |     3,968          0          0 |     3,968
         9 |         0      4,236          0 |     4,236
        10 |         0      2,265          0 |     2,265
        11 |         0      2,931          0 |     2,931
        12 |         0      4,459          0 |     4,459
        13 |         0          0      2,653 |     2,653
        14 |         0          0      1,972 |     1,972
        15 |         0          0      6,018 |     6,018
        16 |         0          0      4,963 |     4,963
        17 |         0          0      2,225 |     2,225
        18 |         0          0      3,454 |     3,454
        19 |         0          0      5,426 |     5,426
        20 |         0          0      2,854 |     2,854
-----------+---------------------------------+----------
     Total |    31,424     13,891     29,565 |    74,880
```

Si ricordi che `recode` funziona solo per variabili numeriche.

Infine vediamo i comandi

`encode` *varname*[*if*][*in*], `generate(`*newvarname*`)` [`label(`*name*`) noextend`]

che trasforma variabili stringa in variabili numeriche, assegnando ai valori creati il label definito in *name* e

`decode` *varname*[*if*][*in*], `generate(`*newvarname*`)` [`maxlength(`#`)`]

che, viceversa, trasforma variabili numeriche in variabili stringa

Comandi similari ai precedenti ma che si applicano quando le variabili sia stringa che non, contengono caratteri numerici, sono:

`destring` [*varlist*], `{generate(`*newvarlist*`)|replace}` [`destring_options`]

che converte variabili numeriche stringa in numeriche pure. Con l'opzione `ignore("`*chars*`")` si possono specificare caratteri non numerici da rimuovere.
Nell'esempio che segue abbiamo la variabile stringa `balance` che contiene sia dati numerici che stringa. Per poterla rendere numerica per prima cosa si provvederà a convertire i dati stringa in missing (`.a` e `.b`) e poi si applicherà il comando `destring`

```
. desc balance;
              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------
balance         str8   %9s
```

```
tab balance, miss;
    balance |      Freq.     Percent        Cum.
------------+-----------------------------------
    1112.99 |          1        6.67        6.67
    1397.99 |          1        6.67       13.33
      16.11 |          1        6.67       20.00
    2177.22 |          1        6.67       26.67
    2371.62 |          1        6.67       33.33
    2517.15 |          1        6.67       40.00
     273.91 |          1        6.67       46.67
     2751.6 |          1        6.67       53.33
   INS Paid |          4       26.67       80.00
       NULL |          3       20.00      100.00
------------+-----------------------------------
      Total |         15      100.00

. destring balance, replace;
balance contains non-numeric characters; no replace

. replace balance =".a" if balance=="INS Paid";
(4 real changes made)

. replace balance =".b" if balance=="NULL";
(3 real changes made)

. destring balance, replace;
balance has all characters numeric; replaced as double
(7 missing values generated)

. summ balance;
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     balance |         8    1577.324    1044.511      16.11     2751.6

. desc balance;
              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
balance         double %10.0g
```

Il comando `tostring`, viceversa converte variabili numeriche in variabili stringa

`tostring` *varlist*, {`generate(`*newvarlist*`)|replace`} [`tostring_options`]

```
. desc eta

              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
eta             int    %10.0g

. summ eta

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         eta |     74880    41.71912    22.25459          0        102

. tostring eta, replace
eta was int now str3

. desc eta

              storage  display     value
variable name   type   format      label      variable label
```

```
-------------------------------------------------------------------------------
eta             str3   %9s

. summ eta

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         eta |         0


. destring eta, replace
eta has all characters numeric; replaced as int

. desc eta

              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
eta             int    %10.0g

. summ eta

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         eta |     74880    41.71912    22.25459          0        102
```

8.5 Creare variabili dummy

Le variabili dummy sono variabili che assumo valori 0 e 1. In particolare 1 quando la caratteristica in esame è presente, 0 quando è assente. Vediamo adesso due metodi per creare questo tipo di variabili. Il primo sistema si basa sulla tecnica del `replace`; si crea una variabile ponedola pari a 0 e poi si sotituisce il valore 1 secondo un certo criterio:

```
. gen nord=0

. replace nord=1 if ripgeo==1 | ripgeo==2
(61098 real changes made)

. tab nord, miss

       nord |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |     78,925       56.37       56.37
          1 |     61,098       43.63      100.00
------------+-----------------------------------
      Total |    140,023      100.00
```

Oppure in maniera meno pedante

```
. gen north = ripgeo<=2

. tab nord, miss

       nord |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |     78,925       56.37       56.37
          1 |     61,098       43.63      100.00
------------+-----------------------------------
      Total |    140,023      100.00
```

Con il secondo metodo si sfrutta l'opzione `gen` del comando `tabulate`

```
. tab ripgeo, miss gen(ripgeo_)

     ripgeo |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |     32,188       22.99       22.99
          2 |     28,910       20.65       43.63
          3 |     26,768       19.12       62.75
          4 |     37,206       26.57       89.32
          5 |     14,951       10.68      100.00
------------+-----------------------------------
      Total |    140,023      100.00

. tab1 ripgeo_*

-> tabulation of ripgeo_1

   ripgeo== |
     1.0000 |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |    107,835       77.01       77.01
          1 |     32,188       22.99      100.00
------------+-----------------------------------
      Total |    140,023      100.00

-> tabulation of ripgeo_2

   ripgeo== |
     2.0000 |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |    111,113       79.35       79.35
          1 |     28,910       20.65      100.00
------------+-----------------------------------
      Total |    140,023      100.00

-> tabulation of ripgeo_3

   ripgeo== |
     3.0000 |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |    113,255       80.88       80.88
          1 |     26,768       19.12      100.00
------------+-----------------------------------
      Total |    140,023      100.00

-> tabulation of ripgeo_4

   ripgeo== |
     4.0000 |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |    102,817       73.43       73.43
          1 |     37,206       26.57      100.00
------------+-----------------------------------
      Total |    140,023      100.00

-> tabulation of ripgeo_5

   ripgeo== |
     5.0000 |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |    125,072       89.32       89.32
          1 |     14,951       10.68      100.00
------------+-----------------------------------
      Total |    140,023      100.00

. summ ripgeo_*
```

```
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
    ripgeo_1 |    140023    .2298765    .4207548          0          1
    ripgeo_2 |    140023    .2064661    .4047703          0          1
    ripgeo_3 |    140023    .1911686    .3932229          0          1
    ripgeo_4 |    140023    .2657135     .441714          0          1
    ripgeo_5 |    140023    .1067753    .3088285          0          1
```

Si noti che la media delle variabili dummy corrisponde alla percentuale di valori 1.

Capitolo 9

Analisi Quantitativa

9.1 `summarize` e `tabulate`

Per prima cosa è bene distinguere le analisi da condurre su variabili continue e quelle su variabili discrete. Per le prime il comando essenziale è:

summarize [*varlist*] [*if*] [*in*] [*weight*] [, detail]

dove `detail` produce un output con un numero maggiore di informazioni. Inoltre questo comando supporta l'uso dei pesi ([*weight*]) che possono essere uno tra `aweight`, `fweight` e `iweight` e servono per produrre delle statistiche pesate.

```
. summ y1  yt, detail

                  Reddito disponibile netto
-------------------------------------------------------------
      Percentiles      Smallest
 1%         3600              0
 5%        12050              0
10%        17000              0       Obs                8001
25%        27200              0       Sum of Wgt.        8001

50%        42600                      Mean           51212.38
                        Largest       Std. Dev.      38811.92
75%        65964       557239.3
90%        92000       630000.8       Variance       1.51e+09
95%       114100       764335.5       Skewness       4.342934
99%       188500         800000       Kurtosis       52.93251

          Reddito da pensioni e altri trasferimenti
-------------------------------------------------------------
      Percentiles      Smallest
 1%            0              0
 5%            0              0
10%            0              0       Obs                8001
25%            0              0       Sum of Wgt.        8001

50%         8840                      Mean           12683.57
                        Largest       Std. Dev.      15618.38
75%        22100         127850
90%        32500         139300       Variance       2.44e+08
95%        41223         150800       Skewness       1.624059
99%        65000         169000       Kurtosis       7.892386
```

```
. summ y1  yt [aweight=pesofit]

Variable |     Obs      Weight        Mean   Std. Dev.     Min        Max
---------+-----------------------------------------------------------------
      y1 |    8001        8001    49152.49   36364.71       0     800000
      yt |    8001        8001    11932.83   14946.91       0     169000
```

Un comando aggiuntivo simile ma che offre una gamma più ampia di possibilità è `fsum`.

```
. fsum y1 yt, f(10.3) s(n abspct miss mean median sum)

 Variable |         N   Missing    AbsPct      Mean    Median       Sum
----------+-----------------------------------------------------------
       y1 |      8001         0     0.000 51212.380 42600.000 4.098e+08
       yt |      8001         0     0.000 12683.571  8840.000 1.015e+08


. fsum y1 yt, f(10.3) s(n miss mean median sum) uselabel

                      Variable |     N Missing    Mean  Median      Sum
-------------------------------+---------------------------------------
    Reddito disponibile netto |  8001       0 51212.3 42600.0 4.098e+08
Reddito da pens e altri trasf |  8001       0 12683.5  8840.0 1.015e+08
```

Per le variabili discrete il comando principale è `tabulate` che analizzeremo nel caso di analisi di una sola variabile e nel caso di incrocio tra due variabili. Nel caso si voglia analizzare la distribuzione di frequenza di una sola variabile discreta il comando è:

`tabulate` *varname*[*if*][*in*][*weight*][, *tabulate_options*]

tra le *tabulate_options* più importanti:

`missing` per includere anche le osservazioni missing
`nolabel` per visualizzare i codici numerici al posto delle etichette dei valori
`sort` per ordinare in senso discendente in base alla frequenza delle diverse specificazioni della variabile
`sum(`*var*`)` per fare il `sum` di una variabile continua per ciascuna specificazione di *varname*

```
. tab d09a

          Chiarezza della |
              segnaletica |      Freq.     Percent        Cum.
--------------------------+-----------------------------------
    Per nulla soddisfatto |         12        0.55        0.55
         Poco soddisfatto |         46        2.11        2.67
              Soddisfatto |        733       33.70       36.37
      Più che Soddisfatto |        563       25.89       62.25
Completamente Soddisfatto |        795       36.55       98.80
               Non saprei |         26        1.20      100.00
--------------------------+-----------------------------------
                    Total |      2,175      100.00

. tab d09a, miss

          Chiarezza della |
              segnaletica |      Freq.     Percent        Cum.
--------------------------+-----------------------------------
```

```
    Per nulla soddisfatto |         12        0.51        0.51
         Poco soddisfatto |         46        1.96        2.48
              Soddisfatto |        733       31.31       33.79
      Più che Soddisfatto |        563       24.05       57.84
Completamente Soddisfatto |        795       33.96       91.80
               Non saprei |         26        1.11       92.91
             Non risponde |        166        7.09      100.00
--------------------------+-----------------------------------
                    Total |      2,341      100.00

. tab d09a, miss nolab

  Chiarezza |
      della |
segnaletica |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |         12        0.51        0.51
          2 |         46        1.96        2.48
          3 |        733       31.31       33.79
          4 |        563       24.05       57.84
          5 |        795       33.96       91.80
          6 |         26        1.11       92.91
         .a |        166        7.09      100.00
------------+-----------------------------------
      Total |      2,341      100.00

. tab d09a, miss nolab sort

  Chiarezza |
      della |
segnaletica |      Freq.     Percent        Cum.
------------+-----------------------------------
          5 |        795       33.96       33.96
          3 |        733       31.31       65.27
          4 |        563       24.05       89.32
         .a |        166        7.09       96.41
          2 |         46        1.96       98.38
          6 |         26        1.11       99.49
          1 |         12        0.51      100.00
------------+-----------------------------------
      Total |      2,341      100.00


. tab d09a, miss nolab sum(d01)

  Chiarezza |
      della |           Summary of Età
segnaletica |        Mean   Std. Dev.       Freq.
------------+------------------------------------
          1 |   51.363636   21.795746          11
          2 |   48.369565   18.587162          46
          3 |   51.283727   19.146486         719
          4 |    51.70991   18.997164         555
          5 |   53.497409    18.69334         772
          6 |      42.375   17.392683          24
         .a |   63.362416   15.511641         149
------------+------------------------------------
      Total |   52.776801   18.965238        2276
```

Sempre nel caso di analisi di frequenza univariata segnalo il comando aggiuntivo `fre` che consente, tra l'altro, di di esportare i risultati anche in Tex:

`fre` [*varlist*] [*if*] [*in*] [*weight*] [, *options*]

Tra le *options* citiamo:

`format(#)` che indica il numero di decimali (2 è il valore di default)
`nomissing` omette dalla tabella il conteggio dei valori missing
`nolabel` omette il label delle variabili
`novalue` omette il valore numerico delle label
`noname` omette il nome della variabile
`notitle` omette il titolo con il nome e la descrizione della variabile
`nowrap` non manda a capo l'intestazione di riga
`width(#)` specifica la larghezza della colonna delle descrizioni
`include` include tutti i valori possibili della variabile, quindi comprende anche quelli a frequenza zero
`include(`*numlist*`)` include solo i valori specificati in *numlist*
`ascending` visualizza i valori in ordine ascendente di frequenza
`descending` visualizza i valori in ordine discendente di frequenza

```
. fre d09a

d09a -- Chiarezza della segnaletica
-------------------------------------------------------------------------------
                                    |      Freq.    Percent       Valid       Cum.
------------------------------------+------------------------------------------
Valid   1  Per nulla soddisfatto    |         12       0.51        0.55       0.55
        2  Poco soddisfatto         |         46       1.96        2.11       2.67
        3  Soddisfatto              |        733      31.31       33.70      36.37
        4  Più che Soddisfatto      |        563      24.05       25.89      62.25
        5  Completamente Soddisfatto |        795      33.96       36.55      98.80
        6  Non saprei               |         26       1.11        1.20     100.00
        Total                       |       2175      92.91      100.00
Missing .a Non risponde             |        166       7.09
Total                               |       2341     100.00
-------------------------------------------------------------------------------

. fre d09a, nomissing

d09a -- Chiarezza della segnaletica
-----------------------------------------------------------------
                              |      Freq.    Percent       Cum.
------------------------------+----------------------------------
 1 Per nulla soddisfatto      |         12       0.55       0.55
 2 Poco soddisfatto           |         46       2.11       2.67
 3 Soddisfatto                |        733      33.70      36.37
 4 Più che Soddisfatto        |        563      25.89      62.25
 5 Completamente Soddisfatto  |        795      36.55      98.80
 6 Non saprei                 |         26       1.20     100.00
 Total                        |       2175     100.00
-----------------------------------------------------------------

. fre d09a, nolabel

d09a
-------------------------------------------------------------
              |      Freq.    Percent       Valid       Cum.
--------------+----------------------------------------------
Valid   1     |         12       0.51        0.55       0.55
        2     |         46       1.96        2.11       2.67
        3     |        733      31.31       33.70      36.37
        4     |        563      24.05       25.89      62.25
```

```
        5     |        795      33.96      36.55      98.80
        6     |         26       1.11       1.20     100.00
        Total |       2175      92.91     100.00
Missing .a    |        166       7.09
Total         |       2341     100.00
-------------------------------------------------------------

. fre d09a, novalue

d09a -- Chiarezza della segnaletica
-------------------------------------------------------------------------------
                                  |      Freq.    Percent      Valid       Cum.
----------------------------------+--------------------------------------------
Valid   Per nulla soddisfatto     |         12       0.51       0.55       0.55
        Poco soddisfatto          |         46       1.96       2.11       2.67
        Soddisfatto               |        733      31.31      33.70      36.37
        Più che Soddisfatto       |        563      24.05      25.89      62.25
        Completamente Soddisfatto |        795      33.96      36.55      98.80
        Non saprei                |         26       1.11       1.20     100.00
        Total                     |       2175      92.91     100.00
Missing Non risponde              |        166       7.09
Total                             |       2341     100.00
-------------------------------------------------------------------------------

. fre d09a, noname

Chiarezza della segnaletica
-------------------------------------------------------------------------------
                                  |      Freq.    Percent      Valid       Cum.
----------------------------------+--------------------------------------------
Valid   1  Per nulla soddisfatto  |         12       0.51       0.55       0.55
        2  Poco soddisfatto       |         46       1.96       2.11       2.67
        3  Soddisfatto            |        733      31.31      33.70      36.37
        4  Più che Soddisfatto    |        563      24.05      25.89      62.25
        5  Completamente Soddisfatto |     795      33.96      36.55      98.80
        6  Non saprei             |         26       1.11       1.20     100.00
        Total                     |       2175      92.91     100.00
Missing .a Non risponde           |        166       7.09
Total                             |       2341     100.00
-------------------------------------------------------------------------------

. fre d09a, notitle

-------------------------------------------------------------------------------
                                  |      Freq.    Percent      Valid       Cum.
----------------------------------+--------------------------------------------
Valid   1  Per nulla soddisfatto  |         12       0.51       0.55       0.55
        2  Poco soddisfatto       |         46       1.96       2.11       2.67
        3  Soddisfatto            |        733      31.31      33.70      36.37
        4  Più che Soddisfatto    |        563      24.05      25.89      62.25
        5  Completamente Soddisfatto |     795      33.96      36.55      98.80
        6  Non saprei             |         26       1.11       1.20     100.00
        Total                     |       2175      92.91     100.00
Missing .a Non risponde           |        166       7.09
Total                             |       2341     100.00
-------------------------------------------------------------------------------

. fre d09a, ascending

d09a -- Chiarezza della segnaletica
-------------------------------------------------------------------------------
                                  |      Freq.    Percent      Valid       Cum.
----------------------------------+--------------------------------------------
Valid   1  Per nulla soddisfatto  |         12       0.51       0.55       0.55
        6  Non saprei             |         26       1.11       1.20       1.75
```

```
        2  Poco soddisfatto          |         46       1.96       2.11       3.86
        4  Più che Soddisfatto       |        563      24.05      25.89      29.75
        3  Soddisfatto               |        733      31.31      33.70      63.45
        5  Completamente Soddisfatto |        795      33.96      36.55     100.00
        Total                        |       2175      92.91     100.00
Missing .a Non risponde              |        166       7.09
Total                                |       2341     100.00
-------------------------------------------------------------------------------

. fre d09a, descending

d09a -- Chiarezza della segnaletica
-------------------------------------------------------------------------------
                                     |      Freq.    Percent      Valid       Cum.
-------------------------------------+-----------------------------------------
Valid   5  Completamente Soddisfatto |        795      33.96      36.55      36.55
        3  Soddisfatto               |        733      31.31      33.70      70.25
        4  Più che Soddisfatto       |        563      24.05      25.89      96.14
        2  Poco soddisfatto          |         46       1.96       2.11      98.25
        6  Non saprei                |         26       1.11       1.20      99.45
        1  Per nulla soddisfatto     |         12       0.51       0.55     100.00
        Total                        |       2175      92.91     100.00
Missing .a Non risponde              |        166       7.09
Total                                |       2341     100.00
-------------------------------------------------------------------------------

. fre d11_3, include

d11_3 -- Chiarezza informazioni ricevute dal personale di segreteria
-------------------------------------------------------------------------------
                                     |      Freq.    Percent      Valid       Cum.
-------------------------------------+-----------------------------------------
Valid   1  Per nulla soddisfatto     |          0       0.00       0.00       0.00
        2  Poco soddisfatto          |          3       0.60       0.66       0.66
        3  Soddisfatto               |        202      40.64      44.69      45.35
        4  Più che soddisfatto       |        111      22.33      24.56      69.91
        5  Completamente soddisfatto |        135      27.16      29.87      99.78
        6  Non saprei                |          1       0.20       0.22     100.00
        Total                        |        452      90.95     100.00
Missing .a Non risponde              |         45       9.05
Total                                |        497     100.00
-------------------------------------------------------------------------------
```

Con la sintassi

`fre` [*varlist*] `using` *filename* [*if*] [*in*] [*weight*] [, *options export_opts*]

possiamo esportare la tabella prodotta da `fre`. Tra le *export_opts*:

`tab` che esporta in formato delimitato da tabulazione (da indicare nell'estensione di *filename*)
`tex` che esporta in formato Tex (da indicare nell'estensione di *filename*)
`pre(`*strlist*`)` testo da visulizzare prima della tabella
`post(`*strlist*`)` testo da visulizzare dopo la tabella
`replace` sovrascrive *filename* se già esistente
`append` aggiunge il risultato di `fre` al *filename* già esistente

```
. fre d09a using tables.tex, pre(Testo Iniziale) post(Testo Finale)
(output written to tables.tex)
```

E questo è il risultato importato in LaTex:

Testo Iniziale
d09a — Chiarezza della segnaletica

			Freq.	Percent	Valid	Cum.
Valid	1	Per nulla soddisfatto	12	0.51	0.55	0.55
	2	Poco soddisfatto	46	1.96	2.11	2.67
	3	Soddisfatto	733	31.31	33.70	36.37
	4	Più che Soddisfatto	563	24.05	25.89	62.25
	5	Completamente Soddisfatto	795	33.96	36.55	98.80
	6	Non saprei	26	1.11	1.20	100.00
	Total		2175	92.91	100.00	
Missing	.a	Non risponde	166	7.09		
Total			2341	100.00		

Testo Finale

Se le variabili sono molte, anziché fare un `tab` per ciascuna, si può ricorrere al comando

`tab1` *varlist*[*if*][*in*][*weight*][`,` *tab1_options*]

che produce la distribuzione di frequenza per ciascuna variabile specificata in *varlist*.

Nel caso di incrocio tra due variabili discrete il comando è:

`tabulate` *varname1 varname2* [*if*][*in*][*weight*][`,` *options*]

Le opzioni più importanti sono:

`chi2` per calcolare il χ^2 di Pearson
`exact` [`(#)`] riporta il test esatto di Fisher
`gamma` riporta il test gamma di Goodman e Kruskal
`taub` riporta il test tau-b (τ) di Kendall
`V` riporta la V di Cramer
`column` riporta la frequenza relativa per colonna
`row` riporta la frequenza relativa per riga
`cell` riporta la frequenza relativa per ciascuna cella
`nofreq` sopprime la frequenza nelle celle (da usare solo in abbinamento con `column`, `row` o `cell`)
`sum(`*var*`)` per fare il `sum` di una variabile continua per ciascuna combinazione di *varname1* e *varname2*

```
. tab a10 a11

           |                a11
       a10 |         0          1          2 |     Total
-----------+---------------------------------+----------
         0 |         9          3          1 |        13
         1 |         2          6          0 |         8
         2 |         1          2          2 |         5
-----------+---------------------------------+----------
     Total |        12         11          3 |        26
```

```
. tab a10 a11, column

+-------------------+
| Key               |
|-------------------|
|     frequency     |
| column percentage |
+-------------------+

           |               a11
       a10 |         0          1          2 |     Total
-----------+---------------------------------+----------
         0 |         9          3          1 |        13
           |     75.00      27.27      33.33 |     50.00
-----------+---------------------------------+----------
         1 |         2          6          0 |         8
           |     16.67      54.55       0.00 |     30.77
-----------+---------------------------------+----------
         2 |         1          2          2 |         5
           |      8.33      18.18      66.67 |     19.23
-----------+---------------------------------+----------
     Total |        12         11          3 |        26
           |    100.00     100.00     100.00 |    100.00


. tab a10 a11, row

+----------------+
| Key            |
|----------------|
|   frequency    |
| row percentage |
+----------------+

           |               a11
       a10 |         0          1          2 |     Total
-----------+---------------------------------+----------
         0 |         9          3          1 |        13
           |     69.23      23.08       7.69 |    100.00
-----------+---------------------------------+----------
         1 |         2          6          0 |         8
           |     25.00      75.00       0.00 |    100.00
-----------+---------------------------------+----------
         2 |         1          2          2 |         5
           |     20.00      40.00      40.00 |    100.00
-----------+---------------------------------+----------
     Total |        12         11          3 |        26
           |     46.15      42.31      11.54 |    100.00


. tab a10 a11, column row

+-------------------+
| Key               |
|-------------------|
|     frequency     |
|  row percentage   |
| column percentage |
+-------------------+

           |               a11
       a10 |         0          1          2 |     Total
-----------+---------------------------------+----------
```

```
          0 |         9          3          1 |        13
            |     69.23      23.08       7.69 |    100.00
            |     75.00      27.27      33.33 |     50.00
------------+-----------------------------------+----------
          1 |         2          6          0 |         8
            |     25.00      75.00       0.00 |    100.00
            |     16.67      54.55       0.00 |     30.77
------------+-----------------------------------+----------
          2 |         1          2          2 |         5
            |     20.00      40.00      40.00 |    100.00
            |      8.33      18.18      66.67 |     19.23
------------+-----------------------------------+----------
      Total |        12         11          3 |        26
            |     46.15      42.31      11.54 |    100.00
            |    100.00     100.00     100.00 |    100.00
```

```
. tab a10 a11, column row cell
```

```
+--------------------+
| Key                |
|--------------------|
|     frequency      |
|  row percentage    |
| column percentage  |
|  cell percentage   |
+--------------------+
```

```
           |               a11
       a10 |         0          1          2 |     Total
-----------+-----------------------------------+----------
         0 |         9          3          1 |        13
           |     69.23      23.08       7.69 |    100.00
           |     75.00      27.27      33.33 |     50.00
           |     34.62      11.54       3.85 |     50.00
-----------+-----------------------------------+----------
         1 |         2          6          0 |         8
           |     25.00      75.00       0.00 |    100.00
           |     16.67      54.55       0.00 |     30.77
           |      7.69      23.08       0.00 |     30.77
-----------+-----------------------------------+----------
         2 |         1          2          2 |         5
           |     20.00      40.00      40.00 |    100.00
           |      8.33      18.18      66.67 |     19.23
           |      3.85       7.69       7.69 |     19.23
-----------+-----------------------------------+----------
     Total |        12         11          3 |        26
           |     46.15      42.31      11.54 |    100.00
           |    100.00     100.00     100.00 |    100.00
           |     46.15      42.31      11.54 |    100.00
```

```
. tab a10 a11, column row cell nofreq
```

```
+--------------------+
| Key                |
|--------------------|
|  row percentage    |
| column percentage  |
|  cell percentage   |
+--------------------+
```

```
           |               a11
       a10 |         0          1          2 |     Total
-----------+-----------------------------------+----------
```

```
          0 |     69.23      23.08       7.69 |    100.00
            |     75.00      27.27      33.33 |     50.00
            |     34.62      11.54       3.85 |     50.00
------------+-----------------------------------+----------
          1 |     25.00      75.00       0.00 |    100.00
            |     16.67      54.55       0.00 |     30.77
            |      7.69      23.08       0.00 |     30.77
------------+-----------------------------------+----------
          2 |     20.00      40.00      40.00 |    100.00
            |      8.33      18.18      66.67 |     19.23
            |      3.85       7.69       7.69 |     19.23
------------+-----------------------------------+----------
      Total |     46.15      42.31      11.54 |    100.00
            |    100.00     100.00     100.00 |    100.00
            |     46.15      42.31      11.54 |    100.00


. tab a10 a11, chi2

            |               a11
       a10 |         0          1          2 |     Total
-----------+---------------------------------+----------
          0 |         9          3          1 |        13
          1 |         2          6          0 |         8
          2 |         1          2          2 |         5
-----------+---------------------------------+----------
      Total |        12         11          3 |        26

          Pearson chi2(4) =  10.7803   Pr = 0.029

. tab a10 a11, sum(pain)

            Means, Standard Deviations and Frequencies of pain

            |              a11
       a10 |         0          1          2 |     Total
-----------+---------------------------------+----------
          0 | 91.975309  87.962962  77.777779 | 89.957265
            | 5.4582514  12.525694          0 | 7.9006867
            |         9          3          1 |        13
-----------+---------------------------------+----------
          1 | 93.055557   65.74074          . | 72.569444
            | 9.8209251  15.180667          . | 18.392012
            |         2          6          0 |         8
-----------+---------------------------------+----------
          2 | 77.777779  40.277777         50 | 51.666666
            |         0  5.8925556  3.9283722 |  15.78697
            |         1          2          2 |         5
-----------+---------------------------------+----------
      Total | 90.972223  67.171716   59.25926 |  77.24359
            | 6.9191447  20.631174  16.276293 | 19.657864
            |        12         11          3 |        26
```

Di nuovo, se vogliamo incrociare a coppie più di 2 variabili il comando da usare è:

`tab2` *varlist* [*if*] [*in*] [*weight*] [, *options*]

che restituisce le distribuzioni di frequenza per ciascuna coppia delle variabili specificate in *varlist*.

```
. tab2  s1 s2 s3 s4

-> tabulation of s1 by s2
```

```
           |                    s2
        s1 |         0          1          2          3          4 |     Total
-----------+-------------------------------------------------------+----------
         0 |         2          8          1          2          3 |        16
         1 |         1          1          2          1          0 |         5
         2 |         0          0          3          0          0 |         3
         3 |         0          0          1          0          0 |         1
         4 |         1          0          0          0          0 |         1
-----------+-------------------------------------------------------+----------
     Total |         4          9          7          3          3 |        26


-> tabulation of s1 by s3

           |               s3
        s1 |         0          1          2 |     Total
-----------+---------------------------------+----------
         0 |        16          0          0 |        16
         1 |         3          1          1 |         5
         2 |         1          1          1 |         3
         3 |         1          0          0 |         1
         4 |         1          0          0 |         1
-----------+---------------------------------+----------
     Total |        22          2          2 |        26


-> tabulation of s1 by s4

           |               s4
        s1 |         0          1          4 |     Total
-----------+---------------------------------+----------
         0 |        13          0          3 |        16
         1 |         3          0          2 |         5
         2 |         1          1          1 |         3
         3 |         1          0          0 |         1
         4 |         1          0          0 |         1
-----------+---------------------------------+----------
     Total |        19          1          6 |        26


-> tabulation of s2 by s3

           |               s3
        s2 |         0          1          2 |     Total
-----------+---------------------------------+----------
         0 |         3          1          0 |         4
         1 |         9          0          0 |         9
         2 |         4          1          2 |         7
         3 |         3          0          0 |         3
         4 |         3          0          0 |         3
-----------+---------------------------------+----------
     Total |        22          2          2 |        26


-> tabulation of s2 by s4

           |               s4
        s2 |         0          1          4 |     Total
-----------+---------------------------------+----------
         0 |         2          0          2 |         4
         1 |         8          0          1 |         9
         2 |         5          1          1 |         7
         3 |         2          0          1 |         3
         4 |         2          0          1 |         3
-----------+---------------------------------+----------
```

```
     Total |        19          1          6 |        26

-> tabulation of s3 by s4

           |                s4
        s3 |         0          1          4 |     Total
-----------+---------------------------------+----------
         0 |        18          0          4 |        22
         1 |         0          0          2 |         2
         2 |         1          1          0 |         2
-----------+---------------------------------+----------
     Total |        19          1          6 |        26
```

9.1.1 Qualcosa di più avanzato

Per produrre tabelle di statistiche più evolute possiamo ricorre al comando `table`

`table` *rowvar* [`colvar` [*supercolvar*]] [*if*] [*in*] [*weight*] [, *options*]

dove in *rowvar* [`colvar` [*supercolvar*]] inseriamo variabili categoriche (fino ad un massimo di 3).
Tra le *options* inseriamo `contents(`*clist*`)` dove *clist* può essere scelto tra

`mean` *varname* media di *varname*
`sd` *varname* deviazione standard di *varname*
`sum` *varname* sommatoria di *varname*
`n` *varname* numero di casi (missing esclusi) di *varname*
`max` *varname* valore massimo di *varname*
`min` *varname* valore minimo di *varname*
`median` *varname* mediana di *varname*
`p1` *varname* 1° percentile di *varname*
`p2` *varname* 2° percentile di *varname*
...
...
`p98` *varname* 98° percentile di *varname*
`p99` *varname* 98° percentile di *varname*
`iqr` *varname* range interquartile (p75-p25) di *varname*

si tenga presente che al massimo si possono inserire 5 statistiche in *clist*.

```
. table fsize, c(mean w_food mean w_cloth mean w_house
         mean w_trasporti mean  w_eduricr) format(%5.3f) row col;

------------------------------------------------------------------------------
Numero di |
component |
i         | mean(w_food)  mean(w_cl~h)  mean(w_ho~e)  mean(w_tr~i)  mean(w_ed~r)
----------+-------------------------------------------------------------------
        2 |        0.328         0.137         0.154         0.243         0.138
        3 |        0.357         0.142         0.151         0.219         0.130
        4 |        0.373         0.139         0.139         0.217         0.132
        5 |        0.393         0.133         0.133         0.209         0.132
        6 |        0.416         0.123         0.129         0.198         0.134
```

```
        7 |        0.433         0.107         0.130         0.203         0.128
        8 |        0.527         0.072         0.091         0.171         0.139
        9 |        0.378         0.116         0.150         0.215         0.141
          |
    Total |        0.366         0.139         0.144         0.220         0.132
------------------------------------------------------------------------------

. table fsize, c(mean w_food iqr w_food sd w_food n  w_food) format(%5.3f) row col

-------------------------------------------------------------------
Numero di |
component |
i         | mean(w_food)   iqr(w_food)    sd(w_food)     N(w_food)
----------+--------------------------------------------------------
        2 |        0.328         0.159         0.117         1,175
        3 |        0.357         0.154         0.108         3,603
        4 |        0.373         0.145         0.105         4,699
        5 |        0.393         0.149         0.107         1,115
        6 |        0.416         0.137         0.103           162
        7 |        0.433         0.132         0.109            26
        8 |        0.527         0.111         0.078             2
        9 |        0.378         0.248         0.176             2
          |
    Total |        0.366         0.151         0.109        10,784
-------------------------------------------------------------------

. table fsize  nch05, c(mean w_food mean w_cloth mean w_house mean
           w_trasporti mean  w_eduricr) format(%5.3f) row col;

-----------------------------------------------
Numero di |
component |    Figli del capofamiglia <=5
i         |     0      1      2      3  Total
----------+------------------------------------
        2 | 0.328                       0.328
          | 0.137                       0.137
          | 0.154                       0.154
          | 0.243                       0.243
          | 0.138                       0.138
          |
        3 | 0.363  0.344                0.357
          | 0.131  0.166                0.142
          | 0.152  0.149                0.151
          | 0.224  0.210                0.219
          | 0.129  0.131                0.130
          |
        4 | 0.373  0.377  0.365         0.373
          | 0.134  0.152  0.162         0.139
          | 0.138  0.140  0.142         0.139
          | 0.224  0.196  0.198         0.217
          | 0.132  0.135  0.133         0.132
          |
        5 | 0.392  0.397  0.394  0.357  0.393
          | 0.130  0.139  0.152  0.184  0.133
          | 0.134  0.135  0.116  0.155  0.133
          | 0.215  0.191  0.197  0.136  0.209
          | 0.129  0.139  0.141  0.168  0.132
          |
        6 | 0.415  0.406  0.460         0.416
          | 0.122  0.125  0.124         0.123
          | 0.125  0.142  0.119         0.129
          | 0.205  0.187  0.168         0.198
          | 0.132  0.140  0.130         0.134
          |
        7 | 0.415  0.439  0.459  0.522  0.433
```

```
            | 0.105  0.108  0.106  0.122  0.107
            | 0.119  0.126  0.161  0.173  0.130
            | 0.240  0.186  0.146  0.022  0.203
            | 0.120  0.140  0.128  0.161  0.128
            |
          8 | 0.582  0.471                0.527
            | 0.083  0.061                0.072
            | 0.075  0.107                0.091
            | 0.161  0.181                0.171
            | 0.099  0.180                0.139
            |
          9 | 0.254  0.502                0.378
            | 0.117  0.115                0.116
            | 0.181  0.119                0.150
            | 0.313  0.117                0.215
            | 0.135  0.147                0.141
            |
      Total | 0.366  0.363  0.372  0.371  0.366
            | 0.133  0.157  0.159  0.178  0.139
            | 0.144  0.144  0.138  0.157  0.144
            | 0.225  0.202  0.197  0.126  0.220
            | 0.131  0.133  0.134  0.168  0.132
------------------------------------------------
```

Se invece abbiamo variabili continue di cui vogliamo avere ulteriori statistiche oltre a quelle fornite da `summarize` possiamo ricorrere a:

`tabstat` *varlist*[*if*][*in*][*weight*][`,` *options*]

dove in *varlist* inseriamo la liste delle variabili continue e tra le *options* in *statistics*() possiamo scegliere tra:

`mean`
`n`
`sum`
`max`
`min`
`sd`
`variance`
`cv` coefficiente di variazione (sd/mean)
`semean` errore standard della media (sd/sqrt(n))
`skewness` (indice di simmetria o indice di Pearson)
`kurtosis` (indice di curtosi o di appiattimento)
`p1`
`p5`
`p10`
`p25`
`median`
`p50`
`p75`
`p90`
`p95`
`p99`

`range` = max - min
`iqr` = p75 - p25
`q` equivalente a p25 p50 p75

```
. tabstat tax_1-tax_8 if nfigli==1,
   by(prof_nmr) stats(sum mean cv p1 p99) col(stat) long notot;

prof_nmr     variable |       sum      mean        cv        p1       p99
----------------------+--------------------------------------------------
6               tax_1 |  81598.11  799.9815  1.965136         0  5761.707
                tax_2 | -18165.78 -178.0959 -1.018802 -565.1445         0
                tax_3 |         0         0         .         0         0
                tax_4 |         0         0         .         0         0
                tax_5 |  31253.79  306.4097  1.099732         0  1039.958
                tax_6 |  54422.84  533.5573  1.099732         0  1810.899
                tax_7 |  812.8596  7.969212  1.099732         0  27.04759
                tax_8 |   1013.78  9.939017  1.099732         0  33.73313
----------------------+--------------------------------------------------
7               tax_1 |  221761.5  2174.132  1.409285         0   8196.94
                tax_2 |  -21492.1 -210.7069 -1.001891 -568.2259         0
                tax_3 | -4326.881  -42.4204 -1.678715  -220.315         0
                tax_4 |         0         0         .         0         0
                tax_5 |  36993.11  362.6776  1.034861         0  1113.488
                tax_6 |  64416.83  631.5376  1.034861         0  1938.939
                tax_7 |  962.1299  9.432646  1.034861         0     28.96
                tax_8 |  1199.946  11.76418  1.034861         0  36.11824
----------------------+--------------------------------------------------
8               tax_1 |  383905.5  3763.779  1.378702         0  13930.61
                tax_2 |     -1766 -17.31372 -1.068916 -58.87674         0
                tax_3 | -4017.233 -39.38464  -1.64311 -201.7244         0
                tax_4 |         0         0         .         0         0
                tax_5 |  52293.15  512.6779  1.008443         0  1556.057
                tax_6 |  91059.09  892.7361  1.008443         0  2709.592
                tax_7 |  1360.059  13.33391  1.008443         0  40.47047
                tax_8 |  1696.234  16.62974  1.008443         0  50.47383
----------------------+--------------------------------------------------
9               tax_1 |  335143.8  3285.724  1.336498         0  12298.62
                tax_2 | -1770.913 -17.36189 -1.160488 -68.73045         0
                tax_3 | -4191.114 -41.08936 -1.643618 -194.5484         0
                tax_4 |         0         0         .         0         0
                tax_5 |  50404.86  494.1653  1.049721         0  1543.804
                tax_6 |  87770.97  860.4997  1.049721         0  2688.256
                tax_7 |  1310.947  12.85242  1.049721         0  40.15179
                tax_8 |  1634.983  16.02925  1.049721         0  50.07639
----------------------+--------------------------------------------------
10              tax_1 |  221761.5  2174.132  1.409285         0   8196.94
                tax_2 |  -21492.1 -210.7069 -1.001891 -568.2259         0
                tax_3 | -4326.881  -42.4204 -1.678715  -220.315         0
                tax_4 |         0         0         .         0         0
                tax_5 |  36993.11  362.6776  1.034861         0  1113.488
                tax_6 |  64416.83  631.5376  1.034861         0  1938.939
                tax_7 |  962.1299  9.432646  1.034861         0     28.96
                tax_8 |  1199.946  11.76418  1.034861         0  36.11824
------------------------------------------------------------------------
```

9.2 Analisi della correlazione

Per l'analisi della correlazione abbiamo a disposizione due comandi. Il primo è:

`correlate` [*varlist*] [*if*] [*in*] [*weight*] [, *correlate_options*]

che visualizza la matrice delle correlazioni o la matrice delle covarianze a seconda dell'opzione scelta. Le due opzioni più importanti sono `means` che visualizza la media, la deviazione standard, il minimo e il massimo delle variabili scelte e `covariance` che invece visualizza la matrice delle covarianze. Attenzione che `correlate` considera solo le osservazioni contemporaneamente valide per tutte le osservazioni.

```
. corr basale q2a q2b q2c q3a q3b q3c q3d, means
(obs=153)

  Variable |         Mean    Std. Dev.          Min          Max
-----------+--------------------------------------------------------
    basale |     .7254902     .9193483            0            3
       q2a |     .5228758     .5629453            0            2
       q2b |     .2418301     .4870176            0            2
       q2c |     .0326797     .1783809            0            1
       q3a |     .6797386     1.004291            0            5
       q3b |     .1764706     .5862195            0            3
       q3c |     .0718954     .3461171            0            3
       q3d |     .0653595     .3378181            0            3

           |   basale      q2a      q2b      q2c      q3a      q3b      q3c      q3d
-----------+------------------------------------------------------------------------
    basale |   1.0000
       q2a |  -0.1149   1.0000
       q2b |   0.0317   0.3276   1.0000
       q2c |  -0.0653   0.1563   0.2113   1.0000
       q3a |  -0.2312   0.1585   0.1190   0.0588   1.0000
       q3b |  -0.0804   0.1571   0.1030   0.0703   0.3313   1.0000
       q3c |  -0.1030   0.0759   0.0133   0.1748   0.1045   0.0019   1.0000
       q3d |  -0.1537   0.0613   0.1032   0.2918   0.3142   0.0743   0.1846   1.0000
```

Qui un esempio di matrice delle correlazioni

```
. corr basale q2a q2b q2c q3a q3b q3c q3d, covariance
(obs=153)

           |   basale      q2a      q2b      q2c      q3a      q3b      q3c      q3d
-----------+------------------------------------------------------------------------
    basale |  .845201
       q2a | -.059469  .316907
       q2b |   .01419  .089826  .237186
       q2c | -.010707  .015695  .018361   .03182
       q3a | -.213493  .089611  .058222  .010535   1.0086
       q3b | -.043344  .051858  .029412  .007353  .195046  .343653
       q3c | -.032766  .014792  .002236  .010793  .036335  .000387  .119797
       q3d |  -.04773  .011653  .016985  .017587  .106596  .014706  .021586  .114121
```

Il secondo comando è:

`pwcorr` [*varlist*] [*if*] [*in*] [*weight*] [, *pwcorr_options*]

che analizza le correlazioni indipendentemente per ciascuna coppia di variabili, per cui la numerosità fa' riferimento al numero di osservazioni valido per ciascuna coppia di variabili. Le opzioni principali sono:

`obs` visualizza il numero di osservazioni per ciascuna coppia di variabili
`sig` visualizza il livello di significatività della correlazione
`star(#)` visualizza con `*` i livelli di significatività inferiori al valore #
`bonferroni` usa il metodo di Bonferroni per correggere il livello di significatività

`sidak` usa il metodo di Sidak per correggere il livello di significatività

```
. pwcorr basale q2a q2b q2c q3a q3b q3c q3d, star(0.05) obs sig

             | valsalva      q2a      q2b      q2c      q3a      q3b      q3c      q3d
-------------+------------------------------------------------------------------------
    valsalva |   1.0000
             |
             |      153
             |
         q2a |   0.0507   1.0000
             |   0.5338
             |      153      162
             |
         q2b |   0.0218   0.3270*  1.0000
             |   0.7895   0.0000
             |      153      162      162
             |
         q2c |   0.0246   0.1559*  0.2148*  1.0000
             |   0.7631   0.0476   0.0061
             |      153      162      162      162
             |
         q3a |  -0.1235   0.1642*  0.1228   0.0602   1.0000
             |   0.1283   0.0368   0.1195   0.4464
             |      153      162      162      162      162
             |
         q3b |   0.0031   0.1102   0.0830   0.0606   0.3190*  1.0000
             |   0.9694   0.1628   0.2939   0.4433   0.0000
             |      153      162      162      162      162      162
             |
         q3c |  -0.1163   0.0779   0.0189   0.1766*  0.1055  -0.0031   1.0000
             |   0.1522   0.3245   0.8115   0.0246   0.1814   0.9683
             |      153      162      162      162      162      162      162
             |
         q3d |  -0.1587   0.0634   0.1078   0.2932*  0.3117*  0.0640   0.1864*  1.0000
             |   0.0501   0.4232   0.1723   0.0002   0.0001   0.4182   0.0175
             |      153      162      162      162      162      162      162      162
             |
```

9.3 Analisi outliers

Tralasciando tutta la teoria attinente al problema dei dati anomali (o outliers), vediamo come Stata ci permette di identificarli. Il comando ufficiale è:

`hadimvo` *varlist*[*if*][*in*], `generate(`*newvar1* [*newvar2*]`)` [`p(`#`)`]

dove la nuova variabile generata *newvar1* (o le nuove variabili generate nel caso di analisi multivariata) sono delle dummy che identificato con 1 i valori candidati ad essere outliers. Il secondo comando, che va aggiunto tramite `ssc`, è:

`grubbs` *varlist*[*if*][*in*][, *options*]

dove tra le opzioni teniamo presenti:

`drop` che elimina le osservazioni identificate come outliers
`generate(newvar1 ...)` che crea variabili dummy per identificare i valori outliers
`log` che visualizza il log delle iterazioni

Come sempre segue l'esempio di utilizzo dei due comandi. Notate la differenza nei tempi di esecuzione tra i due comandi e il fatto che restituiscono il medesimo risultato.

```
. local st = "$S_TIME";

. hadimvo isee, gen(flag_isee);

Beginning number of observations:         13239
              Initially accepted:             2
             Expand to (n+k+1)/2:          6620
                Expand,  p = .05:         13226
              Outliers remaining:            13

. elapse `st´;
Elapsed time was 24 minutes, 33 seconds.

. list id isee if flag_isee==1, sep(0);

       +--------------------+
       |    id        isee |
       |--------------------|
13227. |  6874     63867.1 |
13228. |  6875    64486.92 |
13229. |  6876    67414.06 |
13230. |  6877    72366.19 |
13231. | 13234    73208.93 |
13232. | 13235   73449.813 |
13233. | 13236   77971.492 |
13234. | 13237   78410.148 |
13235. | 13238   84195.203 |
13236. | 13239   88253.977 |
13237. |  6878    93771.63 |
13238. |  6880   127160.35 |
13239. |  6879   127160.35 |
       +--------------------+

. drop flag_isee;

. local st = "$S_TIME";

. grubbs isee, gen(flag_isee) log;
Variable: isee (0/1 variable recording which observations are outliers: flag_isee).
Iteration = 1. T-value: 10.7325 so 127160.35 is an outlier
Iteration = 2. T-value: 7.5214 so 93771.63 is an outlier
Iteration = 3. T-value: 6.9903 so 88253.977 is an outlier
Iteration = 4. T-value: 6.6000 so 84195.203 is an outlier
Iteration = 5. T-value: 6.0350 so 78410.148 is an outlier
Iteration = 6. T-value: 5.9999 so 77971.492 is an outlier
Iteration = 7. T-value: 5.5568 so 73449.813 is an outlier
Iteration = 8. T-value: 5.5394 so 73208.93 is an outlier
Iteration = 9. T-value: 5.4617 so 72366.19 is an outlier
Iteration = 10. T-value: 4.9720 so 67414.06 is an outlier
Iteration = 11. T-value: 4.6833 so 64486.92 is an outlier
Iteration = 12. T-value: 4.6252 so 63867.1 is an outlier
12 outliers. No more outliers

. elapse `st´;
Elapsed time was 6 seconds.

. list id isee if flag_isee==1, sep(0);

       +--------------------+
       |    id        isee |
       |--------------------|
```

```
 1. |  6874     63867.1 |
 2. |  6875    64486.92 |
 3. |  6876    67414.06 |
 4. |  6877    72366.19 |
 5. | 13234    73208.93 |
 6. | 13235   73449.813 |
 7. | 13236   77971.492 |
 8. | 13237   78410.148 |
 9. | 13238   84195.203 |
10. | 13239   88253.977 |
11. |  6878    93771.63 |
12. |  6879   127160.35 |
13. |  6880   127160.35 |
    +-------------------+
```

Infine vi presento un metodo grafico che si basa sull'utilizzo dei boxplot. Data la complessità della sintassi dei grafici, in questo caso mi astengo dal fornirvi la descrizione dettagliata del comando. Impartiamo il seguente comando:

```
graph box isee,  marker(1, mlabel(isee))
```

e otteniamo questa figura

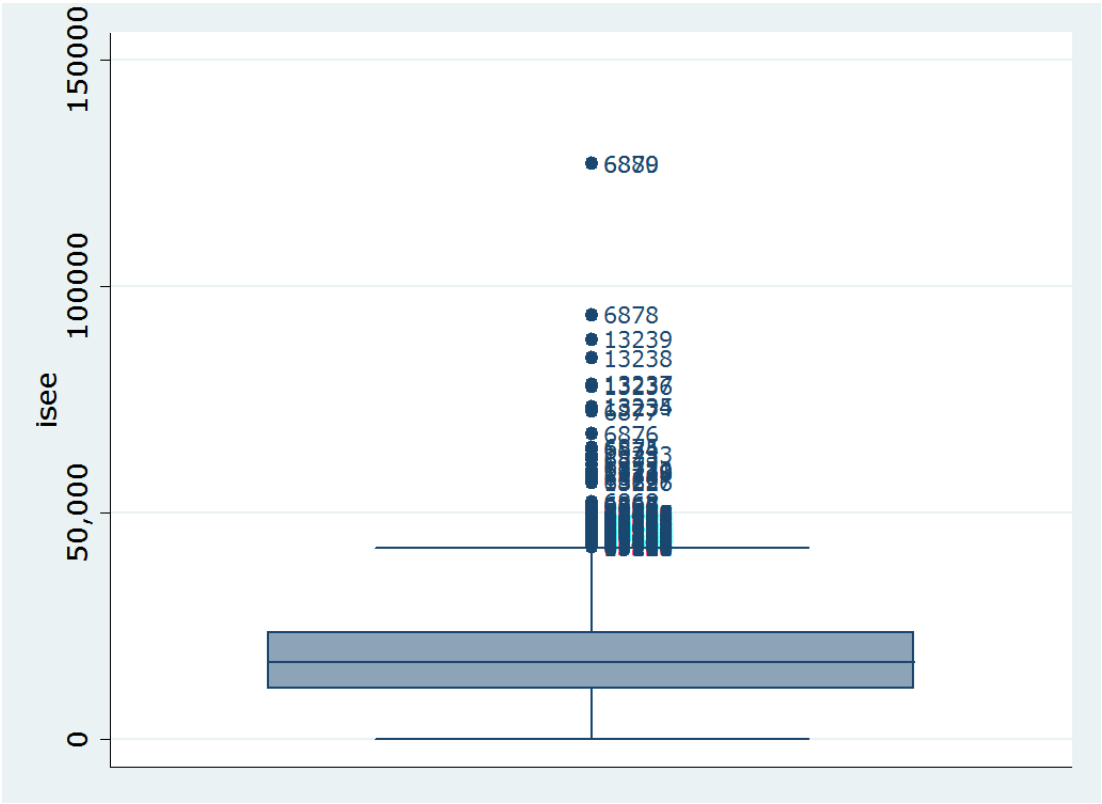


Figura 9.1: Box Plot

La scatola centrale si chiama box e la linea al suo interno identifica la mediana della variabile. All'interno del box vi stanno tutti i valori compresi tra il 25° e il 75° percentile

e si definisce tale intervallo come range interquartile (iqr = valore della variabile al 75° percentile - valore della variaile al 25° percentile). I valori compresi tra il 75° percentile e + $\frac{3}{2}$ (iqr) si definiscono valori adiacenti superiori e graficamente sono rappresentati dal segmento superiore che parte dal box centrale. In maniera analoga i valori compresi tra il 25° percentile e - $\frac{3}{2}$ (iqr) si definiscono valori adiacenti inferiori e graficamente sono rappresentati dal segmento inferiore che parte dal box centrale. I valori che inferiormente e superiormente sono oltre i valori adiacenti sono potenziali outliers e graficamente sono rappresentati con dei punti. Vediamo adesso come identificare questi valori come segue:

```
. gen flag_isee = 0

. summ isee, detail

                            isee
-------------------------------------------------------------
      Percentiles      Smallest
 1%            0              0
 5%       1455.6              0
10%         5429              0       Obs                13239
25%     11021.11              0       Sum of Wgt.        13239

50%     17076.28                      Mean           17837.32
                        Largest       Std. Dev.      10186.18
75%     23550.96       88253.98
90%     30706.24       93771.63       Variance       1.04e+08
95%     36165.33       127160.4       Skewness       .8533689
99%     46270.08       127160.4       Kurtosis       6.205226

. replace flag_isee=1 if isee > r(p75)+3/2*(r(p75)-r(p25))
(289 real changes made)

. replace flag_isee=1 if isee < r(p25)-3/2*(r(p75)-r(p25))
(0 real changes made)

. list id isee if flag_isee==1, sep(0)

       +-------------------+
       |    id        isee |
       |-------------------|
12951. |  6716    42386.39 |
12952. | 13117   42388.102 |
12953. | 13116   42388.102 |
12954. | 13118   42401.871 |
12955. | 13119   42405.672 |

  (output omitted)

13226. | 13233   62853.719 |
13227. |  6874     63867.1 |
13228. |  6875    64486.92 |
13229. |  6876    67414.06 |
13230. |  6877    72366.19 |
13231. | 13234    73208.93 |
13232. | 13235   73449.813 |
13233. | 13236   77971.492 |
13234. | 13237   78410.148 |
13235. | 13238   84195.203 |
13236. | 13239   88253.977 |
13237. |  6878    93771.63 |
13238. |  6879   127160.35 |
13239. |  6880   127160.35 |
```

```
+------------------+
```

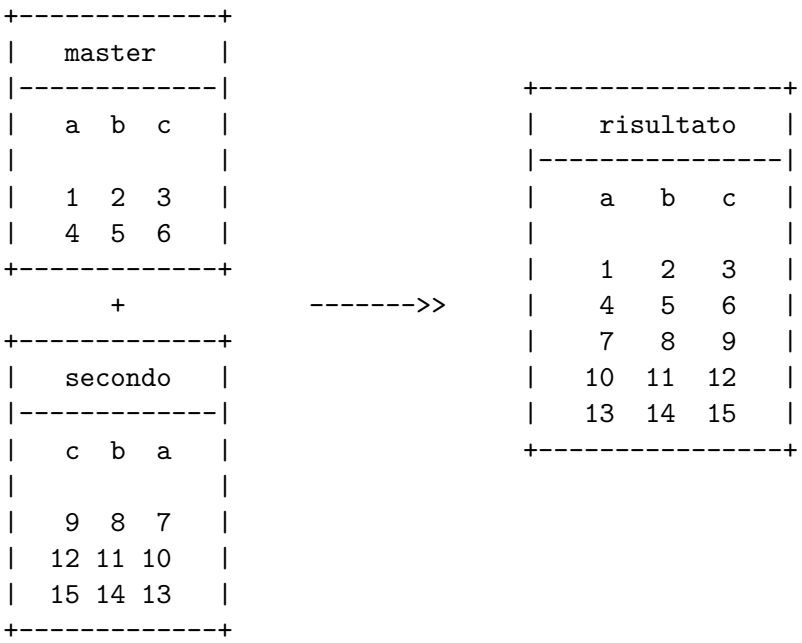
Come si vede con questo metodo i valori indiziati di essere outliers sono assai più numerosi (289).

Capitolo 10

Trasformare Dataset

10.1 Aggiungere osservazioni

In genere quando si aggiungono osservazioni provenienti da un altro dataset si richiede che le variabili abbiano lo stesso nome, mentre non è richiesto che siano nello stesso ordine di posizione; vedi `secondo` dove la disposizione delle variabili è `c`, `b`, `a`. Graficamente ciò che vogliamo ottenere è:

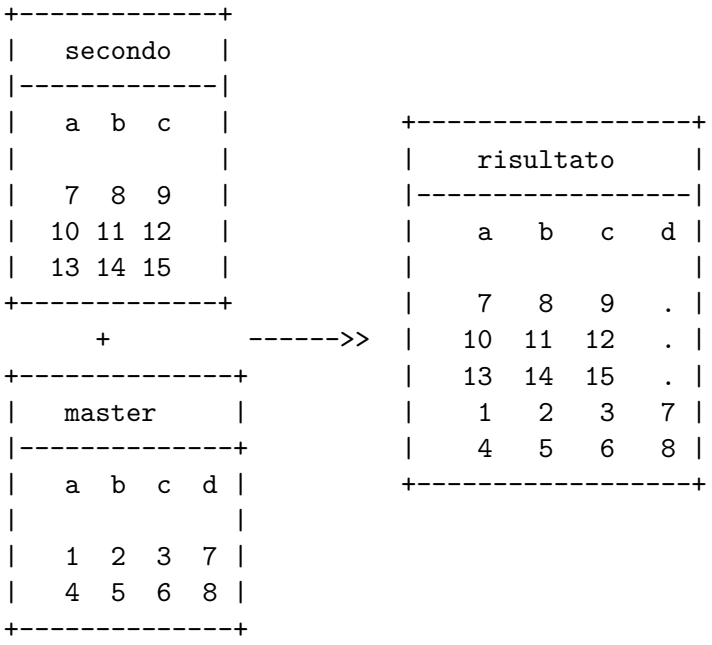


Il primo passo è caricare il primo dataset, poi si aggiungono le osservazioni del secondo dataset tramite il comando:

`append using` *filename* [`,` *options*]

dove in *filename* specifichiamo il dataset da aggiungere e dove tra le *options* c'è `keep(`*varlist*`)` dove è possibile specificare le variabili da prendere nel dataset che viene aggiunto.

Si tenga presente che se una variabile è presente in uno solo dei due dataset, nel dataset unione (`risultato`) per quella variabile ci saranno delle osservazioni missing in corrispondenza delle osservazioni del dataset in cui non era presente.



In conclusione il comando `append` serve essenzialmente per aggiungere osservazioni, e solo indirettamente per aggiungere variabili.

10.2 Aggiungere variabili

Condizione necessaria per aggiungere ad un dataset (`master`) variabili provenienti da un altro dataset (`slave`) è che in entrambi siano presenti una o più variabili che permettano di stabilire una relazione tra le osservazioni del primo e del secondo dataset. In altre parole ci devono essere delle variabili chiave che mi permettano di assegnare ciascun valore X_{ij} proveniente dal dataset `slave` ad una determinata osservazione del dataset `master`.
Noto che spesso gli utilizzatori alle prime armi hanno difficoltà

- ad individuare le variabili chiave
- a selezionare le variabili da inserire nelle *match-variable(s)* per il merge dei dati

Ho già discusso altrove le problematiche relative all'individuazione delle variabili chiave (7.2 Controllo delle variabili chiave). Vi rimando a quella sezione se ancora aveste dei dubbi in merito alla questione.
Invece il discorso relativo al secondo punto è più complesso e per affrontarlo parto da un altro punto di vista. Quanto dovete fare un merge vi troverete in una delle seguenti situazioni:

1. match 1:1 - ad ogni osservazione del master corrisponde una e una sola osservazione dello slave
2. match 1:n - ad ogni osservazione del master corrispondono n osservazioni dello slave
3. match n:1 - ad n osservazioni del master corrispondono 1 sola osservazione dello slave
4. match n:n - ad n osservazioni del master corrispondono n osservazioni dello slave

Il caso 1. è il più semplice; master e slave hanno lo stesso insieme di variabili chiave ed è il caso presentato nella prima rappresentazione grafica che segue. Quindi nelle *match-variable(s)* vanno indicate queste variabili chiave. I casi 2. e 3. sono speculari e sono caratterizzati dal fatto di avere un insieme di variabili chiave diverso tra master e slave. In questo caso le *match-variable(s)* corrispondono alle variabili chiave del dataset 1 (il master nel caso 2. e lo slave nel caso 3.). Rimane vero però che le variabili *match-variable(s)* devono essere presenti in entrambi i dataset. Infine il caso n:n è un caso limite che qui non tratteremo.
In Stata il comando che permettere di compiere questa operazione è `merge` . Nelle versioni precedenti di Stata tale comando prevedeva che i dataset `master` e `slave` fossero già ordinati in base alle variabili chiave per cui non era molto agevole da usare[1]. Qui si farà ricorso al comando `mmerge`, che permette di superare questa limitazione e che in più ha delle opzioni assai utili.
Anche in questo caso diamo una rappresentazione grafica del problema da risolvere:

```
+-------------+       +------------+        +------------------------+
|   master    |       |    slave   |        |        risultato       |
|-------------|       |------------|        |------------------------|
| id   a   b  |       |  id    d   |        | id   a   b   d  _merge |
|             |       |            |        |                        |
|  1  10  11  |   +   |   1   18   |   =    |  1  10  11  18      3  |
|  2  12  13  |       |   3   19   |        |  2  12  13   .      1  |
|  3  14  15  |       |   4   20   |        |  3  14  15  19      3  |
|  5  16  17  |       +------------+        |  4   .   .  20      2  |
+-------------+                             |  5  16  17   .      1  |
                                            +------------------------+
```

[1] Questa limitazione non è più presente nella versione attuale del comando `merge`

`master` è il dataset attualmente in memoria, al quale vogliamo aggiungere la variabile `d` presente nel dataset `slave`. La variabile che ci permette di raccordare i due dataset è `id`. Con essa possiamo assegnare il valore 18 che troviamo in corrispondenza di `id=1` in `slave` all'osservazione sempre con `id=1` in `master`.
Quando associamo le osservazioni si possono presentare 3 casi.

- osservazione presente in entrambi i dataset (`_merge=3`)
- osservazione presente solo nel dataset slave (`_merge=2`)
- osservazione presente solo nel dataset master (`_merge=1`)

Si può intuire dal dataset risultato cosa succede in termini di creazione di dati missing per i casi di `_merge=1` e `_merge=2`.
Quello mostrato è il caso più semplice in cui `id` è variabile chiave per entrambi i dataset. Ma se non fosse così, come mostrato di seguito?

```
+---------------+     +------------+     +---------------------------+
|     master    |     |    slave   |     |          risultato        |
|---------------|     |------------|     |---------------------------|
| f_id  p_id  a |     | f_id   b   |     | f_id  p_id  a   b  _merge |
|               |     |            |     |                           |
|  1      1  18 |  +  |   1   18   |  =  |  1      1  18  18     3   |
|  1      2  13 |     |   2   19   |     |  1      2  13  18     3   |
|  1      3  15 |     |   3   20   |     |  1      3  15  18     3   |
|  2      1  17 |     +------------+     |  2      1  17  19     3   |
|  2      2  16 |                        |  2      2  16  19     3   |
|  3      1  20 |                        |  3      1  20  20     3   |
+---------------+                        +---------------------------+
```

In questo caso l'informazione contenuta in `b` si "spalma" su tutti i `p_id` che hanno lo stesso `f_id` di master. Consiglio, per comodità, di usare sempre come `master` il dataset che ha il numero maggiore di variabili chiave.

Come accennato in precedenza il comando `mmerge` non prevede che i due dataset siano preventivamente ordinati; questa è la sua sintassi:

`mmerge` *match-variable(s)* `using` *filename* [`,` *options*]

dove in *match-variable(s)* vanno indicate la/le variabile/i che servono da raccordo tra i due dataset, in *filename* il percorso e il nome del dataset che funge da `slave`. Tra le principali *options* citiamo:

`ukeep(`*varlist*`)` dove in *varlist* vengono indicate le variabili che vogliamo aggiungere dal dataset *filename*. Se non viene indicato nulla, verranno aggiunte tutte le variabili di *filename*.

`udrop(`*varlist*`)` dove in *varlist* vengono indicate le variabili che non vogliamo aggiungere dal dataset *filename*.

`update` Piccola spiegazione. Di default il dataset `master` è inviolabile ossia nel dataset risultante dal merge saranno tenuti i valori delle variabili del dataset `master` anche se queste stesse sono presenti nello `slave` ed hanno valori diversi. Se viene specificata questa opzione verrà preso il valore del dataset `using` nel caso in cui nel `master` vi siano valori missing.

`replace` deve essere specificata anche l'opzione `update`. Con questa opzione i valori non missing del master saranno sostituiti con i corrispondenti valori non missing dello `slave` quando non corrispondenti. Comunque un valore non missing del `master` non verrà mai sostituito con un valore missing dello `slave`.

`uname(`*stub*`)` specifica il prefisso da aggiungere alle variabili importate dal dataset `slave`.

Alcune di queste opzioni sono un po' complicate quindi vediamo alcuni esempi.

```
. use master,clear

. clist

          mat      corso        fac
  1.    VR0001        M50          M
  2.    VR0002                     E
  3.    VR0003        S07          S
  4.    VR0004        L12          L
  5.    VR0005        L08          L
  6.    VR0006        S33          S
  7.    VR0007        E07          E
  8.    VR0008        G02          G

. use SLAVE, clear

. clist

          mat      corso        fac     disab    partime
  1.    VR0001        M50          M         1         PT
  2.    VR0002        E01          E         0
  3.    VR0003          .          S         0
  4.    VR0004        L12          L         0         PT
  5.    VR0005        L08          L         0
  6.    VR0006        S33          S         1
  7.    VR0007        E07          E         0
  8.    VR0008        G08          G         0         PT
```

Opzione `ukeep(`*varlist*`)`

```
. use MASTER, clear

. mmerge mat using SLAVE, ukeep(disab)

--------------------------------------------------------------------------------
merge specs           |
       matching type  | auto
  mv´s on match vars  | none
  unmatched obs from  | both
----------------------+---------------------------------------------------------
  master         file | MASTER.dta
                  obs |      8
                 vars |      3
          match vars  | mat  (key)
  --------------------+---------------------------------------------------------
  using          file | SLAVE.dta
```

```
                obs |      8
               vars |      2  (selection via udrop/ukeep)
         match vars | mat  (key)
--------------------+-------------------------------------------------------
result          file | MASTER.dta
                obs |      8
               vars |      6  (including _merge)
         ------------+-------------------------------------------------------
             _merge |      8  obs both in master and using data      (code==3)
-------------------------------------------------------------------------------

. clist

          mat      corso        fac     disab                          _merge
  1.    VR0001        M50          M         1     both in master and using data
  2.    VR0002                     E         0     both in master and using data
  3.    VR0003        S07          S         0     both in master and using data
  4.    VR0004        L12          L         0     both in master and using data
  5.    VR0005        L08          L         0     both in master and using data
  6.    VR0006        S33          S         1     both in master and using data
  7.    VR0007        E07          E         0     both in master and using data
  8.    VR0008        G02          G         0     both in master and using data
```

Opzione `udrop(`*varlist*`)`

```
. use MASTER, clear

. mmerge mat using SLAVE, udrop(disab)

-------------------------------------------------------------------------------
merge specs          |
       matching type | auto
  mv´s on match vars | none
  unmatched obs from | both
--------------------+-------------------------------------------------------
  master        file | MASTER.dta
                obs |      8
               vars |      3
         match vars | mat  (key)
  ------------------+-------------------------------------------------------
  using         file | SLAVE.dta
                obs |      8
               vars |      4  (selection via udrop/ukeep)
         match vars | mat  (key)
  ------------------+-------------------------------------------------------
        common vars | corso fac
--------------------+-------------------------------------------------------
result          file | MASTER.dta
                obs |      8
               vars |      6  (including _merge)
         ------------+-------------------------------------------------------
             _merge |      8  obs both in master and using data      (code==3)
-------------------------------------------------------------------------------

. clist

          mat      corso        fac    partime                          _merge
  1.    VR0001        M50          M         PT     both in master and using data
  2.    VR0002                     E                both in master and using data
  3.    VR0003        S07          S                both in master and using data
  4.    VR0004        L12          L         PT     both in master and using data
  5.    VR0005        L08          L                both in master and using data
  6.    VR0006        S33          S                both in master and using data
  7.    VR0007        E07          E                both in master and using data
  8.    VR0008        G02          G         PT     both in master and using data
```

Opzione `update` (Prestate attenzione a cosa succede alla variabile `corso`)

```
. use MASTER, clear

. mmerge mat using SLAVE

--------------------------------------------------------------------------------
merge specs          |
       matching type | auto
  mv´s on match vars | none
  unmatched obs from | both
---------------------+----------------------------------------------------------
  master        file | MASTER.dta
                 obs |      8
                vars |      3
         match vars | mat  (key)
  -------------------+----------------------------------------------------------
  using         file | SLAVE.dta
                 obs |      8
                vars |      5
         match vars | mat  (key)
  -------------------+----------------------------------------------------------
        common vars | corso fac
---------------------+----------------------------------------------------------
result          file | MASTER.dta
                 obs |      8
                vars |      7  (including _merge)
         -----------+----------------------------------------------------------
              _merge |      8  obs both in master and using data      (code==3)
--------------------------------------------------------------------------------

. drop _merge

. clist

          mat      corso        fac     disab    partime
  1.    VR0001        M50          M         1         PT
  2.    VR0002                     E         0
  3.    VR0003        S07          S         0
  4.    VR0004        L12          L         0         PT
  5.    VR0005        L08          L         0
  6.    VR0006        S33          S         1
  7.    VR0007        E07          E         0
  8.    VR0008        G02          G         0         PT

. use MASTER, clear

. mmerge mat using SLAVE, update

--------------------------------------------------------------------------------
merge specs          |
       matching type | auto
  mv´s on match vars | none
  unmatched obs from | both
---------------------+----------------------------------------------------------
  master        file | MASTER.dta
                 obs |      8
                vars |      3
         match vars | mat  (key)
  -------------------+----------------------------------------------------------
  using         file | SLAVE.dta
                 obs |      8
                vars |      5
         match vars | mat  (key)
  -------------------+----------------------------------------------------------
```

```
        common vars | corso fac
--------------------+-----------------------------------------------------------
result          file | MASTER.dta
                 obs |      8
                vars |      7  (including _merge)
           ---------+-----------------------------------------------------------
             _merge |      5  obs in both, master agrees with using data  (code==3)
                    |      1  obs in both, missing in master data updated  (code==4)
                    |      2  obs in both, master disagrees with using data  (code==5)
-------------------------------------------------------------------------------

. clist

   mat      corso      fac    disab    partime                                    _merge
VR0001        M50        M        1         PT     in both, master agrees with using data
VR0002        E01        E        0              in both, missing in master data updated
VR0003        S07        S        0           in both, master disagrees with using data
VR0004        L12        L        0         PT     in both, master agrees with using data
VR0005        L08        L        0                in both, master agrees with using data
VR0006        S33        S        1                in both, master agrees with using data
VR0007        E07        E        0                in both, master agrees with using data
VR0008        G02        G        0         PT  in both, master disagrees with using data
```

Opzione `replace`

```
. use MASTER, clear

. mmerge mat using SLAVE, update replace

-------------------------------------------------------------------------------
merge specs          |
       matching type | auto
  mv´s on match vars | none
  unmatched obs from | both
---------------------+---------------------------------------------------------
  master         file | MASTER.dta
                  obs |      8
                 vars |      3
           match vars | mat  (key)
  -------------------+---------------------------------------------------------
  using          file | SLAVE.dta
                  obs |      8
                 vars |      5
           match vars | mat  (key)
  -------------------+---------------------------------------------------------
        common vars | corso fac
--------------------+-----------------------------------------------------------
result          file | MASTER.dta
                 obs |      8
                vars |      7  (including _merge)
           ---------+-----------------------------------------------------------
             _merge |      5  obs in both, master agrees with using data  (code==3)
                    |      1  obs in both, missing in master data updated  (code==4)
                    |      2  obs in both, master disagrees with using data  (code==5)
-------------------------------------------------------------------------------

. clist

   mat      corso      fac    disab    partime                                    _merge
VR0001        M50        M        1         PT     in both, master agrees with using data
VR0002        E01        E        0              in both, missing in master data updated
VR0003          .        S        0           in both, master disagrees with using data
VR0004        L12        L        0         PT     in both, master agrees with using data
VR0005        L08        L        0                in both, master agrees with using data
VR0006        S33        S        1                in both, master agrees with using data
```

```
VR0007        E07        E        0                in both, master agrees with using data
VR0008        G08        G        0         PT  in both, master disagrees with using data
```

Opzione `uname(`*stub*`)`

```
. use MASTER, clear

. mmerge mat using SLAVE, ukeep(corso fac disab) uname(chk_)

-------------------------------------------------------------------------------
merge specs           |
       matching type  | auto
  mv´s on match vars  | none
  unmatched obs from  | both
----------------------+--------------------------------------------------------
  master         file | MASTER.dta
                  obs |      8
                 vars |      3
           match vars | mat  (key)
  --------------------+--------------------------------------------------------
  using          file | SLAVE.dta
                  obs |      8
                 vars |      4  (selection via udrop/ukeep)
           match vars | mat  (key)
----------------------+--------------------------------------------------------
result           file | MASTER.dta
                  obs |      8
                 vars |      8  (including _merge)
             ---------+--------------------------------------------------------
              _merge |      8  obs both in master and using data      (code==3)
-------------------------------------------------------------------------------

. drop _merge

. clist

          mat      corso        fac  chk_corso    chk_fac  chk_di~b
  1.    VR0001        M50          M        M50          M         1
  2.    VR0002                     E        E01          E         0
  3.    VR0003        S07          S          .          S         0
  4.    VR0004        L12          L        L12          L         0
  5.    VR0005        L08          L        L08          L         0
  6.    VR0006        S33          S        S33          S         1
  7.    VR0007        E07          E        E07          E         0
  8.    VR0008        G02          G        G08          G         0
```

In conclusione il comando `mmerge` serve essenzialmente per aggiungere variabili, e solo indirettamente per aggiungere osservazioni.
Come detto, il comando `mmerge` è un comando aggiunto, quindi non rientra nella dotazione standard di Stata. Il suo utilizzo al posto del comando `merge` standard di Stata aveva senso fino ad un paio di realese fa del programma. A partire dalla 11 questo comando funziona in maniera egeregia e si può tranquillamente usare al posto di `mmerge`; questa è la sua sintassi:

`merge {1:1 | m:1 | 1:m | m:m}` *varlist* `using` *filename* `[ ,` *options*`]`

rammentato che in *varlist* vanno indicate la/le variabile/i che servono da raccordo tra i due dataset ed avendo già spiegato il significato di 1:1, m:1, 1:m e m:m, passiamo ad analizzare le opzioni del comando:

`keepusing(`*varlist*`)` vengono indicate le variabili che vogliamo aggiungere dal dataset *filename*. Se non viene indicato nulla, verranno aggiunte tutte le variabili di *filename*.

`generate(`*newvar*`)` nome della variabile che desccrive l'esito del merging. Se non viene specificato nulla viene creata di default la variabile `_merge`.

`nogenerate` non viene creata la variabile `_merge`.

`nolabel` non copia i label values dallo `using` *filename*.

`nonotes` non copia le note dallo `using` *filename*.

`update` come già spiegato per il comando `mmerge`, di default il dataset `master` è inviolabile ossia nel dataset risultante dal merge saranno tenuti i valori delle variabili del dataset `master` anche se queste stesse sono presenti nello `slave` ed hanno valori diversi. Se viene specificata questa opzione verrà preso il valore del dataset `using` nel caso in cui nel `master` vi siano valori missing.

`replace` deve essere specificata anche l'opzione `update`. Con questa opzione i valori non missing del master saranno sostituiti con i corrispondenti valori non missing dello `slave` quando non corrispondenti. Comunque un valore non missing del `master` non verrà mai sostituito con un valore missing dello `slave`.

`noreport` sopprime la visualizzazione dell'output dell'operazione di merge

`force` consente il merge anche quando una stessa variabile è stringa in un database e numerica nell'altro

`assert(`*results*`)` specifica che risultati del merge devono verificarsi. In caso contrario viene generato un messaggio di errore.

`keep(`*results*`)` specifica quali risultati del merge devono essere presi e di conseguenza quali debbano essere scartati. In altre parole si selezionano le osservazioni che rispettano certi risultati del merge.

10.3 Collassare un dataset

Collassare un dataset vuol dire ridurre il numero delle sue osservazioni, trasformando le informazioni contenute nelle righe che si vanno ad eliminare secondo una certa funzione. È il caso per esempio di un dataset con informazioni sugli individui che viene collassato in un dataset che contenga informazioni aggregate per singola famiglia di appartenenza. Il comando da usare è:

`collapse` *clist* [*if*] [*in*] [*weight*] [`,` *options*]

dove in *clist* si elencano le variabili che verranno collassate con la funzione da applicare secondo lo schema

`(stat)` *varlist*

o

`(stat)` *new_var=varname*

Le funzioni applicabili in `stat` sono:

- `mean` (opzione di default)
- `sd`

- sum
- count
- max
- min
- iqr
- median
- p1, p2,....p50, ..., p98, p99

Infine in `by` vanno le variabili che faranno da variabili chiave per il dataset collassato. Se ne deduce che questo tipo di trattamento è applicabile solo a variabili numeriche; le variabili che non vengono specificate in *clist* o in `by` verranno cancellate.

```
. list;

     +---------------------------+
     | gpa   hour   year   number |
     |---------------------------|
  1. | 3.2     30      1        3 |
  2. | 3.5     34      1        2 |
  3. | 2.8     28      1        9 |
  4. | 2.1     30      1        4 |
  5. | 3.8     29      2        3 |
     |---------------------------|
  6. | 2.5     30      2        4 |
  7. | 2.9     35      2        5 |
  8. | 3.7     30      3        4 |
  9. | 2.2     35      3        2 |
 10. | 3.3     33      3        3 |
     |---------------------------|
 11. | 3.4     32      4        5 |
 12. | 2.9     31      4        2 |
     +---------------------------+

. collapse (count) n_gpa=gpa (mean) gpa (min) mingpa=gpa (max) maxgpa=gpa
  (mean) meangpa=gpa, by(year);

. list;

     +-----------------------------------------------------+
     | year   n_gpa        gpa   mingpa   maxgpa    meangpa |
     |-----------------------------------------------------|
  1. |    1       4        2.9      2.1      3.5        2.9 |
  2. |    2       3   3.066667      2.5      3.8   3.066667 |
  3. |    3       3   3.066667      2.2      3.7   3.066667 |
  4. |    4       2       3.15      2.9      3.4       3.15 |
     +-----------------------------------------------------+
```

10.4 `reshape` di un dataset

Prima di spiegare il `reshape`, occorre introdurre il concetto di wide form e di long form. Si considerino i seguenti esempi di dataset:

```
wide form:                              long form:

id sex inc80  inc81                     id year sex   inc
```

```
----------------------                ----------------------
1     0   5000    5500                  1    80    0   5000
2     1   2000    2200                  1    81    0   5500
3     0   3000    2000                  2    80    1   2000
                                        2    81    1   2200
                                        3    80    0   3000
                                        3    81    0   2000
```

I due dataset sono identici, contengono le stesse informazioni, quello che cambia è l'organizzazione delle informazioni. In Stata esiste il comando `reshape` per trasformare un dataset in wide form in un dataset in long form e viceversa. Ma capirne il meccanismo non è semplicissimo per cui cercherò di frazionare il procedimento e dare delle linee guida. Innanzitutto si noti che il passaggio da un form all'altro comporta un cambiamento delle variabili chiave (`id` in wide, `id` e `year` in long) e in particolare nel passaggio

- da long a wide: una variabile chiave viene perduta e i suoi valori andranno a far parte del nome di un certo gruppo di variabili
- da wide a long: si genera una nuova variabile chiave i cui valori discendono da una parte del nome di un certo gruppo di variabili

Quindi la prima informazione che dobbiamo conoscere è quali sono le variabili chiave del dataset di partenza e quali saranno le variabili chiave del dataset risultante dall'applicazione del comando `reshape`.
Per applicare il comando dobbiamo definire tre elementi ma il loro contenuto è funzione del tipo di `reshape`, per cui :

- da long a wide

  a. elemento `i`: variabile (o variabili) che saranno le variabili chiave nel dataset finale (nel nostro esempio `id`)
  b. elemento `j`: variabile appartenente al gruppo delle variabili chiave del datasset di partenza, che verrà eliminata e i cui valori andranno ad aggiungersi al nome delle nuove variabili del gruppo X_{ij}; nel nostro esempio `year`
  c. elemento X_{ij}: gruppo di variabili il cui valore cambia in funzione delle variabili degli elementi i e j.

- da wide a long

  a. elemento `i`: variabile (o variabili) che sono le variabili chiave nel dataset di partenza (nel nostro esempio `id`)
  b. elemento `j`: variabile chiave (`year` nel nostro esempio) che si aggiungerà alle variabili appartenenti all'elemento `i` e i cui valori sono presi dal nome delle variabili del gruppo X_{ij}; nel nostro esempio `inc80` e `inc81`.

c. elemento X_{ij}: gruppo di variabili il cui valore cambia in funzione delle variabili degli elementi i e j.

le rimanenti variabili non rientrano in nessuno dei precedenti elementi e sono quelle variabili che non sono variabili chiave e che non sono funzione delle variabili degli elementi i e j.
La sintassi per il passaggio nelle due forme è:
per passare da wide a long

`reshape long` *stubnames@*`, i(`*varlist*`) j(`*varname*`)`

per passare da long a wide

`reshape wide` *stubnames*`, i(`*varlist*`) j(`*varname*`)`

in *stubnames* vanno le variabili dell'elemento X_{ij}

Nel nostro esempio sarà:

```
reshape long inc@, i(id) j(year)

reshape wide inc, i(id) j(year)
```

In conclusione il passaggio da wide a long, ha come effetto l'incremento del numero di osservazioni e la diminuzione del numero di variabili, il passaggio da long a wide la diminuzione del numero di osservazione e l'incremento del numero di variabili. Il tutto viene documentato dall'output del comando.
Si tenga presente che tutte le variabili che apparterebbero al gruppo X_{ij} vanno indicate, pena un messaggio di errore. Nel caso non si volessero trasformare alcune variabili è bene cancellarle (`drop`) prima del `reshape`.
Esempio di `reshape long`

```
. desc rela* sesso* eta* statociv* titstu* conprof* ateco* posprof* presenza*, simple
rela1       sesso11     statociv9   conprof7    posprof5
rela2       sesso12     statociv10  conprof8    posprof6
rela3       eta1        statociv11  conprof9    posprof7
rela4       eta2        statociv12  conprof10   posprof8
rela5       eta3        titstu1     conprof11   posprof9
rela6       eta4        titstu2     conprof12   posprof10
rela7       eta5        titstu3     ateco1      posprof11
rela8       eta6        titstu4     ateco2      posprof12
rela9       eta7        titstu5     ateco3      presenza1
rela10      eta8        titstu6     ateco4      presenza2
rela11      eta9        titstu7     ateco5      presenza3
rela12      eta10       titstu8     ateco6      presenza4
sesso1      eta11       titstu9     ateco7      presenza5
sesso2      eta12       titstu10    ateco8      presenza6
sesso3      statociv1   titstu11    ateco9      presenza7
sesso4      statociv2   titstu12    ateco10     presenza8
sesso5      statociv3   conprof1    ateco11     presenza9
sesso6      statociv4   conprof2    ateco12     presenza10
sesso7      statociv5   conprof3    posprof1    presenza11
sesso8      statociv6   conprof4    posprof2    presenza12
sesso9      statociv7   conprof5    posprof3
sesso10     statociv8   conprof6    posprof4

. clist  ID rela1 sesso1 eta1 rela2 sesso2 rela3 sesso3 in 1/10
```

```
           ID   rela1    sesso1    eta1  rela2   sesso2  rela3    sesso3
 1.      5546       1         2      62      .        .      .         .
 2.      4530       1         1      71      2        2      3         2
 3.      6419       1         1      51      2        2      3         2
 4.     23864       1         1      69      5        2      .         .
 5.      5622       1         1      62      5        2      .         .
 6.      8877       1         1      42      2        2      3         2
 7.      3867       1         1      70      2        2      .         .
 8.     16369       1         1      40      2        2      3         2
 9.     10748       1         1      64      2        2      .         .
10.     17607       1         1      40      2        2      3         2

. reshape long rela@ sesso@ eta@ statociv@ titstu@ conprof@ ateco@ posprof@ pres
> enza@, i(ID) j(pers_id)
(note: j = 1 2 3 4 5 6 7 8 9 10 11 12)

Data                               wide   ->   long
-----------------------------------------------------------------------------
Number of obs.                       10   ->     120
Number of variables                 117   ->      19
j variable (12 values)                    ->   pers_id
xij variables:
                rela1 rela2 ... rela12    ->   rela
              sesso1 sesso2 ... sesso12   ->   sesso
                   eta1 eta2 ... eta12    ->   eta
     statociv1 statociv2 ... statociv12   ->   statociv
          titstu1 titstu2 ... titstu12    ->   titstu
        conprof1 conprof2 ... conprof12   ->   conprof
             ateco1 ateco2 ... ateco12    ->   ateco
        posprof1 posprof2 ... posprof12   ->   posprof
     presenza1 presenza2 ... presenza12   ->   presenza
-----------------------------------------------------------------------------

. drop if rela==. & sesso==.
(93 observations deleted)

. clist  ID pers_id rela sesso rela eta in 1/20

           ID    pers_id   rela     sesso   rela      eta
 1.      3867          1      1         1      1       70
 2.      3867          2      2         2      2       72
 3.      4530          1      1         1      1       71
 4.      4530          2      2         2      2       69
 5.      4530          3      3         2      3       35
 6.      5546          1      1         2      1       62
 7.      5622          1      1         1      1       62
 8.      5622          2      5         2      5       76
 9.      6419          1      1         1      1       51
10.      6419          2      2         2      2       49
11.      6419          3      3         2      3       29
12.      6419          4      4         2      4       85
13.      8877          1      1         1      1       42
14.      8877          2      2         2      2       37
15.      8877          3      3         2      3       15
16.     10748          1      1         1      1       64
17.     10748          2      2         2      2       64
18.     16369          1      1         1      1       40
19.     16369          2      2         2      2       33
20.     16369          3      3         2      3       12
```

Esempio di `reshape wide`

```
. clist nquest nord ireg anasc sesso eta staciv
```

```
     nquest   nord   ireg   anasc   sesso   eta  staciv
  1.      34      1      8    1943       1    59       1
  2.      34      2      8    1944       2    58       1
  3.      34      3      8    1971       1    31       2
  4.      34      4      8    1973       1    29       2
  5.     173      1     18    1948       1    54       1
  6.     173      2     18    1950       2    52       1
  7.     173      3     18    1975       1    27       2
  8.     173      4     18    1978       1    24       2
  9.     375      1     16    1925       2    77       4
 10.     375      2     16    1926       1    76       1

. reshape wide ireg anasc sesso eta , i(nquest) j(nord)
(note: j = 1 2 3 4)
staciv not constant within nquest
Type "reshape error" for a listing of the problem observations.
r(9);

** Ooops! ho dimenticato staciv. Adesso rimedio.

. reshape wide ireg anasc sesso eta staciv, i(nquest) j(nord)
(note: j = 1 2 3 4)

Data                               long   ->   wide
-----------------------------------------------------------------------------
Number of obs.                       10   ->       3
Number of variables                   7   ->      21
j variable (4 values)              nord   ->   (dropped)
xij variables:
                                   ireg   ->   ireg1 ireg2 ... ireg4
                                  anasc   ->   anasc1 anasc2 ... anasc4
                                  sesso   ->   sesso1 sesso2 ... sesso4
                                    eta   ->   eta1 eta2 ... eta4
                                 staciv   ->   staciv1 staciv2 ... staciv4
-----------------------------------------------------------------------------

. clist nquest anasc1 anasc2 anasc3 anasc4

          nquest    anasc1    anasc2    anasc3    anasc4
  1.          34      1943      1944      1971      1973
  2.         173      1948      1950      1975      1978
  3.         375      1925      1926         .         .
```

10.5 Contrarre un dataset

La contrazione consiste nella sostituzione del dataset corrente con un nuovo dataset costituito da tutte le possibili combinazioni tra un gruppo di variabili a cui è possibile aggiungere anche nuove variabili con le frequenze e le percentuali di ciascuna combinazione. Per spiegare il fenomeno ricorriamo ad un piccolo esempio. Abbiamo un dataset che tra le altre variabili contiene le variabili dummy `higher_edu_M_l` e `higher_edu_F_l` dove il valore 1 significa che il soggetto è laureato, 0 non laureato:

```
. tab1 higher_edu_M_l higher_edu_F_l, miss

-> tabulation of higher_edu_M_l

     (mean) |
higher_edu_ |
        M_l |      Freq.     Percent        Cum.
```

```
------------+-----------------------------------
          0 |      4,310       88.59       88.59
          1 |        500       10.28       98.87
          . |         55        1.13      100.00
------------+-----------------------------------
      Total |      4,865      100.00

-> tabulation of higher_edu_F_l

     (mean) |
higher_edu_ |
        F_l |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |      4,387       90.17       90.17
          1 |        443        9.11       99.28
          . |         35        0.72      100.00
------------+-----------------------------------
      Total |      4,865      100.00
```

e questa è la frequenza combinata delle due variabili:

```
. tab  higher_edu_M_l higher_edu_F_l, miss

    (mean) |
higher_edu |      (mean) higher_edu_F_l
      _M_l |         0          1          . |     Total
-----------+---------------------------------+----------
         0 |     4,082        209         19 |     4,310
         1 |       269        229          2 |       500
         . |        36          5         14 |        55
-----------+---------------------------------+----------
     Total |     4,387        443         35 |     4,865
```

Quello che si vuole ottenere è un nuovo dataset che in questo caso sarà costituito da 9 osservazioni, ovvero da tutte le possibili combinazioni delle due variabili. Questo è possibile tramite il comando

`contract` *varlist* [*if*] [*in*] [*weight*] [, *options*]

dove le possibili *options* sono:

`freq(`*newvar*`)` nome della variabile che contiene le frequenze della combinazione. Il nome di default è `_freq`

`cfreq(`*newvar*`)` nome della variabile che contiene le frequenze cumulate della combinazione

`percent(`*newvar*`)` nome della variabile che contiene le percentuali della combinazione

`cpercent(`*newvar*`)` nome della variabile che contiene le percentuali cumulate della combinazione

`float` fa in modo che le variabili delle percentuali siano in formato `float`

`format(`*format*`)` formato di visualizzazione per le variabili delle percentuali: di default è `format(%8.2f)`

`zero` include anche le combinazioni con frequenza zero

`nomiss` cancella le osservazioni con valori missing

Applichiamo il comando al nostro piccolo esempio

```
. contract higher_edu_M_l higher_edu_F_l, freq(_freq) cfreq(_cfreq)
```

```
                                        percent(_perc) cpercent(_cperc);

. clist;

     highe~M_l  highe~F_l         _freq      _cfreq      _perc    _cperc
  1.         0          0          4082        4082      83.91     83.91
  2.         0          1           209        4291       4.30     88.20
  3.         0          .            19        4310       0.39     88.59
  4.         1          0           269        4579       5.53     94.12
  5.         1          1           229        4808       4.71     98.83
  6.         1          .             2        4810       0.04     98.87
  7.         .          0            36        4846       0.74     99.61
  8.         .          1             5        4851       0.10     99.71
  9.         .          .            14        4865       0.29    100.00
```

oppure volendo escludere le osservazioni missing otterremo un dataset con 4 osservazioni

```
. tab higher_edu_M_l higher_edu_F_l;

    (mean) |
higher_edu | (mean) higher_edu_F_l
      _M_l |         0          1 |     Total
-----------+----------------------+----------
         0 |     4,082        209 |     4,291
         1 |       269        229 |       498
-----------+----------------------+----------
     Total |     4,351        438 |     4,789

. contract higher_edu_M_l higher_edu_F_l, freq(_freq) cfreq(_cfreq)
           percent(_perc) cpercent(_cperc) nomiss;

. clist;

     highe~M_l  highe~F_l         _freq      _cfreq      _perc    _cperc
  1.         0          0          4082        4082      85.24     85.24
  2.         0          1           209        4291       4.36     89.60
  3.         1          0           269        4560       5.62     95.22
  4.         1          1           229        4789       4.78    100.00
```

Per un'applicazione pratica di `contract` vi rimando al capitolo 19 a pagina 181.

Capitolo 11

Lavorare con Date e Orari

11.1 La teoria

Vi siete mai chiesti come vengono trattate le date e gli orari dai software? Ebbene bisogna scindere ciò che viene rappresentato a video da quello che il programma elabora. Prendiamo ad esempio le seguenti visualizzazioni di una data:

```
. desc data_in data_out

              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
data_in         long   %dD_m_Y
data_out        long   %dD_m_Y

. clist data_in data_out in 1

       data_in   data_out
  1. 01 Jan 06  22 Feb 06

. format data_in data_out %td

. clist data_in data_out in 1

       data_in   data_out
  1. 01jan2006  22feb2006

. format data_in data_out %tdddMonthYY

. clist data_in data_out in 1

          data_in       data_out
  1.    1January06   22February06

. format data_in data_out %tdDD/NN/CCYY

. clist data_in data_out in 1

       data_in    data_out
  1. 01/01/2006  22/02/2006
```

Come vedete ci sono diverse visualizzazioni della stessa informazione che in realtà è un numero intero:

```
. summ  data_in data_out in 1

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     data_in |         1       16802           .      16802      16802
    data_out |         1       16854           .      16854      16854
```

Infatti il meccanismo si basa sull'assegnazione alla data del 01/01/1960 del valore 0, al 02/01/1960 del valore 1 e così via. Quindi il calcolo dei giorni tra due date si riduce ad una differenza tra due numeri interi. La stessa logica si applica ai dati rigurdanti le ore. In questo caso si ragiona in termine di millisecondi (ms) a partire dalle ore 0, 0 minuti, 0 secondi e 0 millisecondi del 1 gennaio 1960. Per i calcoli si considerino le seguenti equivalenze:

- 1 secondo = 1000ms
- 1 minuto = 60sec × 1000ms = 60000ms
- 1 ora = 60min × 60sec × 1000ms = 3600000ms
- 1 giorno = 24h × 60min × 60sec × 1000ms = 86400000ms

E per date e ore antecedenti il primo gennaio 1960? Ovviamente si ricorre a numeri interi negativi!
In Stata sono possibili diversi sistemi di misura e a partire dalla versione 10 è stata introdotta anche la possibilità di usare la misura basata sulle ore. A seconda del formato che associamo al dato avremo le seguenti interpretazioni:

Formato	**Rappresenta**	**Valore -1**	**Valore 0**	**Valore 1**
`%tc`	orario	31 Dicembre 1959 23:59:59:999	1 Gennaio 1960 00:00:00:000	1 Gennaio 1960 00:00:00:001
`%tC`	orario	31 Dicembre 1959 23:59:59:999	1 Gennaio 1960 00:00:00:000	1 Gennaio 1960 00:00:00:001
`%td`	giorni	31 Dicembre 1959	1 Gennaio 1960	2 Gennaio 1960
`%tw`	settimane	1959 settimana 52	1960 settimana 1	1960 settimana 2
`%tm`	mesi	1959 dicembre	1960 gennaio	1960 febbraio
`%tq`	quadrimestri	1959 quarto quadrimestre	1960 primo quadrimestre	1960 secondo quadrimestre

%th	semestri	1959 secondo semestre	1960 primo semestre	1960 secondo semestre
%ty	anni	1 ac	0	1 dc
%tg	generico	-1	0	1

Vediamo adesso a cosa corripondono i valori di `var1` in formato %tc e %tC:

```
. gen double var2 = var1

. format var2 %tc

. gen double var3 = var1

. format var3 %tC

. clist

              var1                 var2                 var3
 1.         -100000  31dec1959 23:58:20  31dec1959 23:58:20
 2.              0  01jan1960 00:00:00  01jan1960 00:00:00
 3.            100  01jan1960 00:00:00  01jan1960 00:00:00
 4.           1000  01jan1960 00:00:01  01jan1960 00:00:01
 5.          10000  01jan1960 00:00:10  01jan1960 00:00:10
 6.         100000  01jan1960 00:01:40  01jan1960 00:01:40
 7.      100000000  02jan1960 03:46:40  02jan1960 03:46:40
 8.   100000000000  03mar1963 09:46:40  03mar1963 09:46:40
```

A questo punto spieghiamo la differenza tra %tc e %tC. Le due configurazioni sono uguali eccetto per il fatto che %tC tiene conto anche dei secondi inseriti in seguito a correzioni astronomiche per la sincronizzazione con la rotazione della terra (standard GMT). Invece %tc si basa sull'orologio atomico (standard UTC).
Per completare il quadro vediamo i valori dei rimanenti %t per diversi valori di var1:

```
. gen var4 = var1

. format var4 %td

. gen var5 = var1

. gen var5 = var1

. format var5 %tw

. gen var6 = var1

. format var6 %tm

. gen var7 = var1

. format var7 %tq

. gen var8 = var1

. format var8 %th
```

```
. gen var9 = var1

. format var9 %ty

. gen var10 = var1

. format var10 %tg

. clist var1-var5, noobs
```

var1	var2	var3	var4	var5
-10	31dec1959 23:58:20	31dec1959 23:58:20	22dec1959	1959w43
0	01jan1960 00:00:00	01jan1960 00:00:00	01jan1960	1960w1
10	01jan1960 00:00:00	01jan1960 00:00:00	11jan1960	1960w11
100	01jan1960 00:00:01	01jan1960 00:00:01	10apr1960	1961w49
200	01jan1960 00:00:10	01jan1960 00:00:10	19jul1960	1963w45
500	01jan1960 00:01:40	01jan1960 00:01:40	15may1961	1969w33
800	02jan1960 03:46:40	02jan1960 03:46:40	11mar1962	1975w21
1000	03mar1963 09:46:40	03mar1963 09:46:40	27sep1962	1979w13

```
. clist var1 var6-var10, noobs
```

var1	var6	var7	var8	var9	var10
-10	1959m3	1957q3	1955h1	-10	-10
0	1960m1	1960q1	1960h1	0	0
10	1960m11	1962q3	1965h1	10	10
100	1968m5	1985q1	2010h1	0100	100
200	1976m9	2010q1	2060h1	0200	200
500	2001m9	2085q1	2210h1	0500	500
800	2026m9	2160q1	2360h1	0800	800
1000	2043m5	2210q1	2460h1	1000	1000

11.2 Visualizzazione delle date e delle ore

Le possibilità di visualizzazione sono ampie e si basano sul concatenamento di diversi codici. Partiamo dalla seguente lista di codici:

Codice	**Rappresenta**	**Visualizza**
`CC`	secolo - 1	01-99
`cc`	secolo -1	1-99
`YY`	anno a due cifre	00-99
`yy`	anno a due cifre	0-99
`JJJ`	giorno dell'anno	001-366
`jjj`	giorno dell'anno	1-366
`Mon`	mese	Jan, Feb,..., Dec
`Month`	mese	January, February,..., December
`mon`	mese	jan, feb,..., dec
`month`	mese	january, february,..., december
`NN`	mese	01-12
`nn`	mese	1-12
`DD`	giorno del mese	01-31
`dd`	giorno del mese	1-31

Codice	**Rappresenta**	**Visualizza**
`DAYNAME`	giorno della settimana	Sunday, Monday, ... (allineato?)
`Dayname`	giorno della settimana	Sunday, Monday, ... (non allineato?)
`Day`	giorno della settimana	Sun, Mon,...
`Da`	giorno della settimana	Su, mo,...
`day`	giorno della settimana	sun, mon,...
`da`	giorno della settimana	su, mo,...
`h`	semestre	1-2
`q`	quadrimestre	1-4
`WW`	settimana	01-52
`ww`	settimana	1-52
`HH`	ora	00-23
`Hh`	ora	00-12
`hH`	ora	0-23
`hh`	ora	0-12
`MM`	minuti	00-59
`mm`	minuti	0-59
`SS`	secondi	00-60
`ss`	secondi	0-60
`.s`	decimi	.0-.9
`.ss`	centesimi	.00-.99
`.sss`	millesimi	.000-.999
`am`	visualizza am o pm	am o pm
`a.m.`	visualizza a.m. o p.m.	a.m. o p.m.
`AM`	visualizza AM o PM	AM o PM
`A.M.`	visualizza A.M. o P.M.	A.M. o P.M.
`.`	visualizza un punto	.
`,`	visualizza una virgola	,
`:`	visualizza i due punti	:
`-`	visualizza un trattino	-
`_`	visualizza uno spazio	
`/`	visualizza la barra inclinata a dx	/
`\`	visualizza la barra inclinata a sx	\
`!tc`	visualizza il carattere c	c
`+`	separatore della sintassi[1]	

The maximum length of a format specifier is 48 characters; the example shown above is 34 characters.

[1] Serve solo a separare i vari codici affinché siano più leggibili, ma non ha nessun effetto su quanto visualizzato

11.3 Ricavare date da variabili stringa

```
. desc data_nascita

              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------------------
data_nascita    str10  %10s

. list  data_nascita in 1/8, sep(0)

     +------------+
     | data_nas~a |
     |------------|
  1. | 30/05/1982 |
  2. | 17/12/1982 |
  3. | 22/02/1982 |
  4. | 28/08/1981 |
  5. | 14/02/1982 |
  6. | 22/06/1982 |
  7. | 02/07/1982 |
  8. | 18/10/1982 |
     +------------+


. gen double da_nas = date(data_nascita,"DMY")

. format  da_nas %tdDD/NN/CCYY

. list  da_nas in 1/8, sep(0)

     +------------+
     | data_nas~a |
     |------------|
  1. | 30/05/1982 |
  2. | 17/12/1982 |
  3. | 22/02/1982 |
  4. | 28/08/1981 |
  5. | 14/02/1982 |
  6. | 22/06/1982 |
  7. | 02/07/1982 |
  8. | 18/10/1982 |
     +------------+

. format  da_nas %tdDD-NN-CCYY

. list  da_nas in 1/8, sep(0)

     +------------+
     |     da_nas |
     |------------|
  1. | 30-05-1982 |
  2. | 17-12-1982 |
  3. | 22-02-1982 |
  4. | 28-08-1981 |
  5. | 14-02-1982 |
  6. | 22-06-1982 |
  7. | 02-07-1982 |
  8. | 18-10-1982 |
     +------------+
```

TO BE CONTINUED...

11.4 Visualizzazione delle ore

11.5 Operazioni con date e ore

Capitolo 12

Macros e Cicli

12.1 Macros

In Stata esistono due tipi di macros: `local` e `global`. La distinzione tra le due è attinente alla programmazione per cui in questa sede le possiamo considerare come equivalenti. La loro funzione è quella di un contenitore in cui inserire numeri o stringhe da richiamare in un secondo momento. I modi per assegnare un contenuto sono diversi e comunque prevedono l'assegnazione di un nome. Per evitare problemi meglio scegliere nomi diversi da quelli assegnati alle variabili.

```
local A 2+2
local A = 2+2
local B "hello world"
local B = "hello world"

global A 2+2
global A = 2+2
global B "hello  world"
global B = "hello world"
```

Si vede che è possibile assegnare il contenuto alla macro sia con il segno = che senza. La differenza è sostanziale quando si assegnano valori o espressioni numeriche. Vediamo un esempio:

```
. local A 2+2

. local B = 2+2

. di `A'
4

. di `B'
4

. di "`A'"
2+2

. di "`B'"
4
```

Con `local A 2+2` sto' assegnano ad `A` 2+2, che sarà interpretato come operazione algebrica se lo uso direttamente (`di `A'`), come stringa se lo uso con `"` (`di "`A'"`). Con

`local B = 2+2` invece sarà sempre interpretato come operazione algebrica. È importante essere a conoscenza di questa differenza nel momento in cui si richiamano le macros create perché sono diversi i contenuti di `A`. Stesso discorso vale per le `global`.

Vediamo ora come richiamare le macros:

- le `local` si richiamano con l'espressione `` `local_name' ``
- le `global` si richiamano con l'espressione `$local_name`

il simbolo ` si ottiene premendo ALT + 96 sul tastierino numerico

Adesso vediamo qualche uso pratico. Per esempio possiamo definire una lista di variabili da utilizzare successivamente in diverse situazioni:

```
local list = "inc2001 inc2000 inc1999 inc1998 inc1997 inc1996 inc1995"

. di "`list'
inc2001 inc2000 inc1999 inc1998 inc1997 inc1996 inc1995

. summ `list'

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     inc2001 |      1460    32961.99    44469.03          0     480000
     inc2000 |      1476    32833.82    44145.67          0     600000
     inc1999 |      1393    31891.78    41724.69          0     480000
     inc1998 |      1400    31550.77    40743.81          0     410000
     inc1997 |      1369    31438.15    37784.66          0     350000
-------------+--------------------------------------------------------
     inc1996 |      1364     32373.4    40198.93          0     600000
     inc1995 |      1413    30598.08     36889.4          0     360000

. regress  inc2002 `list'

      Source |       SS       df       MS              Number of obs =    1328
-------------+------------------------------           F(  7,  1320) =  745.90
       Model |  2.2244e+12     7  3.1777e+11           Prob > F      =  0.0000
    Residual |  5.6234e+11  1320   426018392           R-squared     =  0.7982
-------------+------------------------------           Adj R-squared =  0.7971
       Total |  2.7867e+12  1327  2.1000e+09           Root MSE      =   20640

------------------------------------------------------------------------------
     inc2002 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     inc2001 |   .8007813   .0340487    23.52   0.000     .7339858    .8675769
     inc2000 |   .1072529   .0472282     2.27   0.023     .0146023    .1999035
     inc1999 |   .0850962   .0556251     1.53   0.126     -.024027    .1942195
     inc1998 |  -.0983748   .0463245    -2.12   0.034    -.1892526   -.0074971
     inc1997 |   .1441902   .0399267     3.61   0.000     .0658633     .222517
     inc1996 |  -.1652574   .0452349    -3.65   0.000    -.2539975   -.0765173
     inc1995 |   .0531481   .0438402     1.21   0.226    -.0328559    .1391521
       _cons |   3392.282   755.4408     4.49   0.000     1910.286    4874.278
------------------------------------------------------------------------------
```

Vedremo tra poco come utilizzare le `local` all'interno dei cicli e successivamente come usarle per catturare e riutilizzare l'output dei comandi.
Vediamo però anche come utilizzare un altro oggetto chiamato `scalar`. Esso serve per assegnare valori numerici scalari e si costruisce così:

`scalar` [`define`] *scalar_name* =*exp*

```
. scalar w=5

. scalar q=3

. scalar t=w+q

. di t
8
```

12.2 I cicli

I cicli sono delle procedure che permettono di compiere azioni ripetitive in maniera più veloce ed efficiente usando poche righe di codice. I metodi di implementazione sono diversi a seconda del contesto in cui si vogliono utilizzare. Analizziamo adesso il metodo `foreach` la cui sintassi generale è:

`foreach` *lname* `{in|of` *listtype*`}` *list* `{`
commands referring to `` `lname' ``
`}`

in pratica succederà che di volta in volta tutti gli oggetti specificati in *list* verranno assegnati a *lname* e quindi eseguiti in base alla lista di comandi specificata tra le due parentesi graffe (*commands referring to `lname'*).
Sono possibili le seguenti costruzioni di `foreach`:
I costruzione

`foreach` *lname* `in` *any_list* `{`
`....  lista di comandi`
`}`

È la costruzione più generale e in *any_list* possiamo inserire una qualsiasi lista: nomi di variabili, stringhe e numeri.

II costruzione

`foreach` *lname* `of local` *lmacname* `{`
`....  lista di comandi`
`}`

previa specificazione del contenuto di una `local` *lmacname*, possiamo utilizzare il suo contenuto in questo tipo di ciclo.

III costruzione

`foreach` *lname* `of global` *gmacname* `{`
`....  lista di comandi`
`}`

previa specificazione del contenuto di una `global` *gmacname*, possiamo utilizzare il suo contenuto in questo tipo di ciclo.

IV costruzione

```
foreach lname of varlist varlist {
        ....  lista di comandi
}
```

utilizzeremo questa costruzione solo quando faremo riferimento ad una serie di variabili già esistenti.

V costruzione

```
foreach lname of newvarlist newvarlist {
        ....  lista di comandi
}
```

costruzione poco usata dove in *newvarlist* si indica una lista di nuove variabili che verranno create all'interno del ciclo

VI costruzione

```
foreach lname of numlist numlist {
        ....  lista di comandi
}
```

che consente di sfruttare le proprietà delle *numlist* di Stata (che vedremo tra poco).

Per capire meglio vediamo alcuni esempi. Per la prima costruzione:

```
. foreach obj in var1 var2 var8 var10 {
  2. summ `obj'
  3. gen `obj'_10 = `obj' / 10
  4. summ `obj'_10
  5. }

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
        var1 |       888    .4868541    .2868169   .0003254   .9990993

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     var1_10 |       888    .0486854    .0286817   .0000325   .0999099

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
        var2 |       888    .4839523    .2927765    .004523   .9999023

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     var2_10 |       888    .0483952    .0292776   .0004523   .0999902

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
```

```
        var8 |       888    .4880482    .2916573   .0005486   .9985623

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     var8_10 |       888    .0488048    .0291657   .0000549   .0998562

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
       var10 |       888    .5048937    .2783813    .003708   .9995353

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
    var10_10 |       888    .0504894    .0278381   .0003708   .0999535
```

quello che accade è che ad ogni ciclo in `obj` viene sostituita in sequenza `var1`, `var2`, `var8` e infine `var10` in questa maniera:

```
primo ciclo (`obj´ = var1):
summ `obj´                      summ var1
gen `obj´_10 = `obj´ / 10       gen var1_10 = var1 / 10
summ `obj´_10                   summ var1_10

secondo ciclo (`obj´ = var2):
summ `obj´                      summ var2
gen `obj´_10 = `obj´ / 10       gen var2_10 = var2 / 10
summ `obj´_10                   summ var2_10

terzo ciclo (`obj´ = var8):
summ `obj´                      summ var8
gen `obj´_10 = `obj´ / 10       gen var8_10 = var8 / 10
summ `obj´_10                   summ var8_10

quarto e ultimo ciclo  (`obj´ = var10):
summ `obj´                      summ var10
gen `obj´_10 = `obj´ / 10       gen var10_10 = var10 / 10
summ `obj´_10                   drop var10_10
```

Vediamo ora un esempio per la seconda costruzione. Per prima cosa dobbiamo definire la `local` e poi rifacciamo lo stesso ciclo:

```
local lista = "var1 var2 var8 var10"

. foreach obj of local lista {
  2. summ `obj'
  3. gen `obj'_10 = `obj' / 10
  4. summ `obj'_10
  5. }

  (output omitted) ... tanto è uguale al precedente
```

Si noti che la `local` all'interno del ciclo `foreach` viene richiamata SENZA l'uso degli apostrofi.

Per la terza costruzione definiamo la `global`

```
. global lista = "var1 var2 var8 var10"

. foreach obj of global lista {
  2. summ `obj'
  3. gen `obj'_10 = `obj' / 10
  4. summ `obj'_10
  5. }
```

(*output omitted*) `... idem come sopra`

Anche qui è da notare che la `global` viene richiamata senza il simbolo $ davanti.

Per la quarta costruzione possiamo sfruttare le possibilità offerte da Stata in merito alla selezione delle variabili su cui eseguire i comandi:

```
. foreach obj of varlist var? {
  2. summ `obj'
  3. gen `obj'_10 = `obj' / 10
  4. summ `obj'_10
  5. }

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
        var1 |       888    .4868541    .2868169   .0003254   .9990993

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     var1_10 |       888    .0486854    .0286817   .0000325   .0999099

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
        var2 |       888    .4839523    .2927765    .004523   .9999023

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     var2_10 |       888    .0483952    .0292776   .0004523   .0999902

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
        var8 |       888    .4880482    .2916573   .0005486   .9985623

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     var8_10 |       888    .0488048    .0291657   .0000549   .0998562
```

Notare che `var10` non viene considerata perchè non rientra in `var?`. Tralasciando la quinta costruzione, vediamo un esempio della sesta, annidandola però all'interno di un altro ciclo (ebbene sì, i cicli possono essere inseriti all'interno di altri cicli):

```
. foreach obj of varlist var1? {
  2. foreach expo of numlist 2/4 6 {
  3. gen `obj'_`expo' = `obj'^(`expo')
  4. }
  5. summ `obj'_*
  6. }

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     var10_2 |       888    .3323266     .286364   .0000137   .9990708
     var10_3 |       888    .2448896    .2705386   5.10e-08   .9986064
     var10_4 |       888    .1923529    .2533452   1.89e-10   .9981424
     var10_6 |       888    .1330797     .224473   2.60e-15   .9972148

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     var11_2 |       888    .3293853    .2923062   1.48e-06   .9904835
     var11_3 |       888    .2443889    .2775777   1.80e-09   .9857593
     var11_4 |       888    .1938413    .2602036   2.19e-12   .9810576
     var11_6 |       888    .1366885     .229854   3.24e-18   .9717214

    Variable |       Obs        Mean    Std. Dev.       Min        Max
```

```
-------------+--------------------------------------------------------
     var12_2 |       888     .341181    .3057852   9.02e-07   .9987798
     var12_3 |       888    .2591221    .2929291   8.56e-10   .9981703
     var12_4 |       888    .2098038    .2770279   8.13e-13    .997561
     var12_6 |       888    .1528551    .2492659   7.33e-19   .9963439
```

In pratica per ciascuna variabile il cui nome inizia `var1#` viene costruita una variabile con la sua trasformazione al quadrato, al cubo, alla quarta e alla sesta. Anche in questo caso esaminiamo la successione delle operazioni:

```
primo loop del ciclo principale (`obj´ = var10)
  primo loop del ciclo annidato (`expo´ = 2)
    gen `obj´_`expo´ = `obj´^(`expo´)     gen var10_2 = var10^(2)

    secondo loop ciclo annidato (`expo´ = 3)
    gen `obj´_`expo´ = `obj´^(`expo´)     gen var10_3 = var10^(3)

    terzo loop ciclo annidato (`expo´ = 4)
    gen `obj´_`expo´ = `obj´^(`expo´)     gen var10_4 = var10^(4)

    quarto loop ciclo annidato (`expo´ = 6)
    gen `obj´_`expo´ = `obj´^(`expo´)     gen var10_6 = var10^(6)

  chiusura loop del ciclo annidato
summ `obj´_*                              summ var10_*


secondo loop del ciclo principale (`obj´ = var11)
  primo loop del ciclo annidato (`expo´ = 2)
    gen `obj´_`expo´ = `obj´^(`expo´)     gen var11_2 = var11^(2)

    secondo loop ciclo annidato (`expo´ = 3)
    gen `obj´_`expo´ = `obj´^(`expo´)     gen var11_3 = var11^(3)

    terzo loop ciclo annidato (`expo´ = 4)
    gen `obj´_`expo´ = `obj´^(`expo´)     gen var11_4 = var11^(4)

    quarto loop ciclo annidato (`expo´ = 6)
    gen `obj´_`expo´ = `obj´^(`expo´)     gen var11_6 = var11^(6)

  chiusura loop del ciclo annidato
summ `obj´_*                              summ var11_*

terzo loop del ciclo principale (`obj´ = var12)
  primo loop del ciclo annidato (`expo´ = 2)
    gen `obj´_`expo´ = `obj´^(`expo´)     gen var12_2 = var12^(2)

    secondo loop ciclo annidato (`expo´ = 3)
    gen `obj´_`expo´ = `obj´^(`expo´)     gen var12_3 = var12^(3)

    terzo loop ciclo annidato (`expo´ = 4)
    gen `obj´_`expo´ = `obj´^(`expo´)     gen var12_4 = var12^(4)

    quarto loop ciclo annidato (`expo´ = 6)
    gen `obj´_`expo´ = `obj´^(`expo´)     gen var12_6 = var12^(6)

  chiusura loop del ciclo annidato
summ `obj´_*                              summ var12_*
```

Infine esiste un'altra costruzione da usare però solo con serie numeriche:

`forvalues` *lname* `=` *range* `{`

commands referring to `lname'

```
}
```

dove *range* può assumere le seguenti configurazioni:

- $\#_1(\#_d)\#_2$: *lname* assume valori da $\#_1$ a $\#_2$ con passo pari a $\#_d$
- $\#_1/\#_2$: *lname* assume valori da $\#_1$ a $\#_2$ con passo pari a 1
- $\#_1$ ($\#_t$) to $\#_2$: *lname* assume valori da $\#_1$ a $\#_2$ con passo pari a $\#_t$ - $\#_1$
- $\#_1$ ($\#_t$) : $\#_2$: idem come sopra

Un esempio:

```
forvalues n = 1(1)90 {
replace var`n' = var`n' + alvar`n'
}
```

che esegue il `replace` su sulle 90 variabili `var1`, `var2`, ..., `var90`.

Capitolo 13

Catturare Informazioni dagli Output

Ogni volta che eseguite un comando, Stata salva parte dell'output del comando e altri valori che vengono calcolati durante l'esecuzione in particolari `local` che possono essere richiamate ed utilizzate. Il comando per vedere l'elenco dei risultati salvati è `return list`:

```
. summ price

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
       price |        74    6165.257    2949.496       3291      15906

. return list

scalars:
                  r(N) =  74
              r(sum_w) =  74
               r(mean) =  6165.256756756757
                r(Var) =  8699525.974268789
                 r(sd) =  2949.495884768919
                r(min) =  3291
                r(max) =  15906
                r(sum) =  456229

. summ price, detail

                            Price
-------------------------------------------------------------
      Percentiles      Smallest
 1%         3291           3291
 5%         3748           3299
10%         3895           3667       Obs                  74
25%         4195           3748       Sum of Wgt.          74

50%       5006.5                      Mean           6165.257
                        Largest       Std. Dev.      2949.496
75%         6342          13466
90%        11385          13594       Variance        8699526
95%        13466          14500       Skewness       1.653434
99%        15906          15906       Kurtosis       4.819188
```

```
. return list

scalars:
                  r(N) =  74
              r(sum_w) =  74
               r(mean) =  6165.256756756757
                r(Var) =  8699525.97426879
                 r(sd) =  2949.49588476892
           r(skewness) =  1.653433511704859
           r(kurtosis) =  4.819187528464004
                r(sum) =  456229
                r(min) =  3291
                r(max) =  15906
                 r(p1) =  3291
                 r(p5) =  3748
                r(p10) =  3895
                r(p25) =  4195
                r(p50) =  5006.5
                r(p75) =  6342
                r(p90) =  11385
                r(p95) =  13466
                r(p99) =  15906
```

Invece nel caso di una regressione si deve usare `ereturn list`:

```
. regress price mpg rep78 weight length foreign

      Source |       SS       df       MS              Number of obs =      69
-------------+------------------------------           F(  5,    63) =   15.90
       Model |   321789308     5  64357861.7           Prob > F      =  0.0000
    Residual |   255007650    63  4047740.48           R-squared     =  0.5579
-------------+------------------------------           Adj R-squared =  0.5228
       Total |   576796959    68  8482308.22           Root MSE      =  2011.9

------------------------------------------------------------------------------
       price |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         mpg |  -26.01325   75.48927    -0.34   0.732    -176.8665      124.84
       rep78 |   244.4242    318.787     0.77   0.446    -392.6208    881.4691
      weight |   6.006738    1.03725     5.79   0.000      3.93396    8.079516
      length |  -102.2199   34.74826    -2.94   0.005    -171.6587   -32.78102
     foreign |   3303.213   813.5921     4.06   0.000     1677.379    4929.047
       _cons |   5896.438   5390.534     1.09   0.278    -4875.684    16668.56
------------------------------------------------------------------------------

. ereturn list

scalars:
                  e(N) =  69
               e(df_m) =  5
               e(df_r) =  63
                  e(F) =  15.8997005734978
                 e(r2) =  .5578900919500578
               e(rmse) =  2011.899719996595
                e(mss) =  321789308.4202555
                e(rss) =  255007650.4493099
               e(r2_a) =  .5228020040095862
                 e(ll) =  -619.6398259855126
               e(ll_0) =  -647.7986144493904

macros:
            e(cmdline) : "regress price mpg rep78 weight length foreign"
              e(title) : "Linear regression"
                e(vce) : "ols"
             e(depvar) : "price"
```

```
                e(cmd) : "regress"
         e(properties) : "b V"
            e(predict) : "regres_p"
              e(model) : "ols"
          e(estat_cmd) : "regress_estat"

matrices:
                  e(b) :  1 x 6
                  e(V) :  6 x 6

functions:
             e(sample)
```

Ritornando al primo esempio, tutti gli `r()` sono dei risultati che possiamo richiamare all'interno dei comandi o che possiamo salvare in `local`. Infatti bisogna tener presente che i valori salvati in `r()` cambiano dopo l'esecuzione del comando e contengono solo quelli relativi all'ultimo comando eseguito.
Se per esempio voglio costruire una variabile (`var3`) che sia la moltiplicazione di una variabile (`var2`) per la media di un'altra (`var1`), dovrò fare:

```
. summ var2

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
        var2 |        88    .5022995    .2882645   .0057233   .9844069

. summ var1

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
        var1 |        88    .4676347    .2849623   .0369273   .9763668

. return list

scalars:
                 r(N) =  88
             r(sum_w) =  88
              r(mean) =  .4676347244530916
               r(Var) =  .0812035311082154
                r(sd) =  .284962332788415
              r(min) =  .0369273088872433
              r(max) =  .9763667583465576
              r(sum) =  41.15185575187206

. gen var3 = var2 * r(mean)

. summ var3

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
        var3 |        88    .2348927    .1348025   .0026764   .4603429
```

Oppure se voglio salvare in una `local` la sommatoria di una variabile

```
. summ var1

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
        var1 |        88    .4676347    .2849623   .0369273   .9763668

. return list
```

```
scalars:
                  r(N) =  88
              r(sum_w) =  88
               r(mean) =  .4676347244530916
                r(Var) =  .0812035311082154
                 r(sd) =  .284962332788415
                r(min) =  .0369273088872433
                r(max) =  .9763667583465576
                r(sum) =  41.15185575187206

. local sommatoria_var1 = r(sum)

. di `sommatoria_var1'
41.151856
```

Capitolo 14

Esportazione dell'output

L'esportazione dei risultati prodotti in Stata è un argomento molto sentito dagli utenti e nel corso del tempo sono stati sviluppati molti comandi che cercano di dare una risposta a questo problema. Stata 14 prima e Stata 15 poi, hanno introdotto una serie di comandi che vanno in questa direzione come per esempio .....
Personalmente dividerei questo tipo di comandi in due categorie. Quelli che esportano un singolo output (export singolo) e quelli che consentono di produrre un documento unico con tutti i vari risultati (export completo). Senza la pretesa di essere esaustivo vi segnalo alcuni di questi comandi.

- export singolo
  - export excel
  - estout
  - ...
- export completo
  - texdoc
  - weaver
  - markdoc
  - ...

Vi rimando agli help dei singoli comandi se volete vedere cosa fanno e come lo fanno. Nelle sezioni successive invece vi illustrerò alcuni comandi che ho creato io con lo scopo di esportare in Latex e in Microsoft Excel gli output di alcuni comandi di Stata. Per quanto possibile ho cercato riprodurre la sintassi del comando Stata tralasciando quelle opzioni che o non sono riuscito a riprogrammare o che non hanno nessuna attinenza con l'output. Alcune parti sono ancora incomplete e ci possono essere degli errori per cui faccio affidamento sul feedback degli utenti. Lo spunto per scrivere questi comandi è nato dalla "fatica" di dover produrre report statistici per varie ricerche e quindi al

momento mi sono concentrato solo su quegli output che sono funzionali alla soluzione di questo problema. Io li uso in abbinamento ai pacchetti `weaver` o `texdoc` per produrre dei report finali. Questo è un esempio dove tutte le tabelle sono state prodotte con uno dei comandi che vi presento più sotto.
Al momento sono in fase più o meno avanzata i seguenti comandi:

• arraytex e arrayxls: non esiste in Stata ma serve per esportare le statistiche descrittive delle domande di tipo array (chiamate anche batterie di item) fretex e frexls: esportazione dell'output del comando `fre`. Si tenga presente che fre è un sostituto del comando `tabulate oneway` • mrtabtex e mrtabxls: esportano l'output del comando `mrtab` • tabletex e tablexls: esportano l'output del comando `table` • tabstattex e tabstatxls: esportano l'output del comando `tabstat` ed è da considerarsi anche come sostituto anche del comando `summarize` • tabtex e tabxls: esportano l'output del comando `tabulate twoway`

14.1 arraytex e arrayxls

14.2 fretex e frexls

14.3 mrtabtex e mrtabxls

14.4 tabletex e tablexls

14.5 tabstattex e tabstatxls

14.6 tabtex e tabxls

Capitolo 15

Mappe

Ora anche in Stata è possibile rappresentare i dati su base geografica (georeferenziazione) grazie all'ottimo lavoro di Maurizio Pisati tramite il comando `spmap`. Per prima cosa bisogna procurarsi i dati geografici che in genere vi verranno forniti in uno dei due formati che sono standard di fatto: shape file e MapInfo Interchange Format. Per poterli utilizzare con `spmap` occorre convertirli in database di Stata. Gli shape file vengono convertiti attraverso il comando

`shp2dta using` *shpfilename* `, database(`*filename*`) coordinates(`*filename*`) [replace genid(`*newvarname*`) gencentroids(`*stub*`)]`

dove:

shpfilename è il file con estensione .shp

`database(`*filename*`)` è il nome del nuovo dataset di Stata che conterrà le informazioni del .dbf file.

`coordinates(`*filename*`)` è il nome del nuovo dataset di Stata che conterrà le informazioni dello .shp file ovvero le coordinate per disegnare i confini degli oggetti da rappresentare.

`replace` sovrascrive i file specificati in `database(`*filename*`)` e in `coordinates(`*filename*`)`.

`genid(`*newvarname*`)` specifica il nome di una nuova variabile in `database(`*filename*`)` che sarà un identificativo delle diverse aree geografiche. I valori assunti da questa variabile corrispondono a quelli della variabile `_ID` presente nel file `coordinates(`*filename*`)`.

`gencentroids(`*stub*`)` genera le variabili `x_stub` e `y_stub` in `database(`*filename*`)` che contengono le coordinate dei centroidi delle aree geografiche.

I files MapInfo sono solitamente due con lo stesso nome ma uno con estensione .mif che contiene le coordinate dei poligoni da disegnare e l'altro con estensione .mid che contiene i dati riferiti alle aree geografiche. Il comando per convertire questo tipo di dati è

`mif2dta` *rootname*`, genid(`*newvarname*`) [gencentroids(`*stub*`) ]`

dove:

rootname è il nome comune dei due files .mif e .mid

`genid(`*newvarname*`)` specifica il nome di una nuova variabile che sarà un identificativo delle diverse aree geografiche.

`gencentroids(`*stub*`)` genera le variabili `x_stub` e `y_stub` in che contengono le coordinate dei centroidi delle aree geografiche.

Il comando genera due database .dta: rootname-Database.dta e rootname-Coordinates.dta.

Bene! Ora abbiamo i files in formato Stata e pronti ad essere utilizzati con il comando `spmap` per la rappresentazione geografica. `spmap` è veramente ricco di opzioni per cui ho riportato l'help del comando in Appendice (pag. 207) assieme ad alcuni esempi grafici. Qui si discuterà di alcuni aspetti inerenti l'utilizzo. Le coordinate dei centroidi non sempre sono corrette nel posizionare gli elementi che si vogliono rappresentare al centro dell'area geografica per cui bisogna correggerli. Questa operazione non è difficile dato che si basano su coordinate cartesiane, comunque bisogna investirci un po' di tempo. Ecco un esempio pratico in cui si riportano le iniziali dei comuni della provincia di Verona, prima senza e poi con la correzione delle coordinate dei centroidi.

```
. local PR "vr";

. /*** conversione shape file nel formato voluto da Stata per il comando spmap***/;
. shp2dta using `PR'_comuni.shp, database(`PR') coordinates(`PR'_coord)
>  replace genid(ID) gencentroids(c);

. use `PR'.dta, clear;

. rename ID _ID;

. spmap sup using "`PR'_coord", id(_ID) fcolor(Blues2) clnumber(98) ocolor(white ..)
>  label(label(nom_com_abb) x(x_c) y(y_c) size(1.5) )
>  legenda(off) title("`sch'", size(*0.8));

. graph export map_pre.png, replace;
(note: file map_pre.png not found)
(file map_pre.png written in PNG format)
```

e questa è la mappa risultante.
OK, adesso la serie di correzioni delle coordinate e il relativo risultato

```
. replace y_c=y_c + 1100 if cod_com==30;
(1 real change made)

. replace x_c=x_c - 800 if cod_com==30;
(1 real change made)

. replace y_c=y_c - 400 if cod_com==36;
(1 real change made)

. replace x_c=x_c + 1000 if cod_com==36;
(1 real change made)

  (output omitted)

. replace y_c=y_c + 400 if cod_com==10;
(1 real change made)

. replace y_c=y_c - 600 if cod_com==56;
(1 real change made)
```

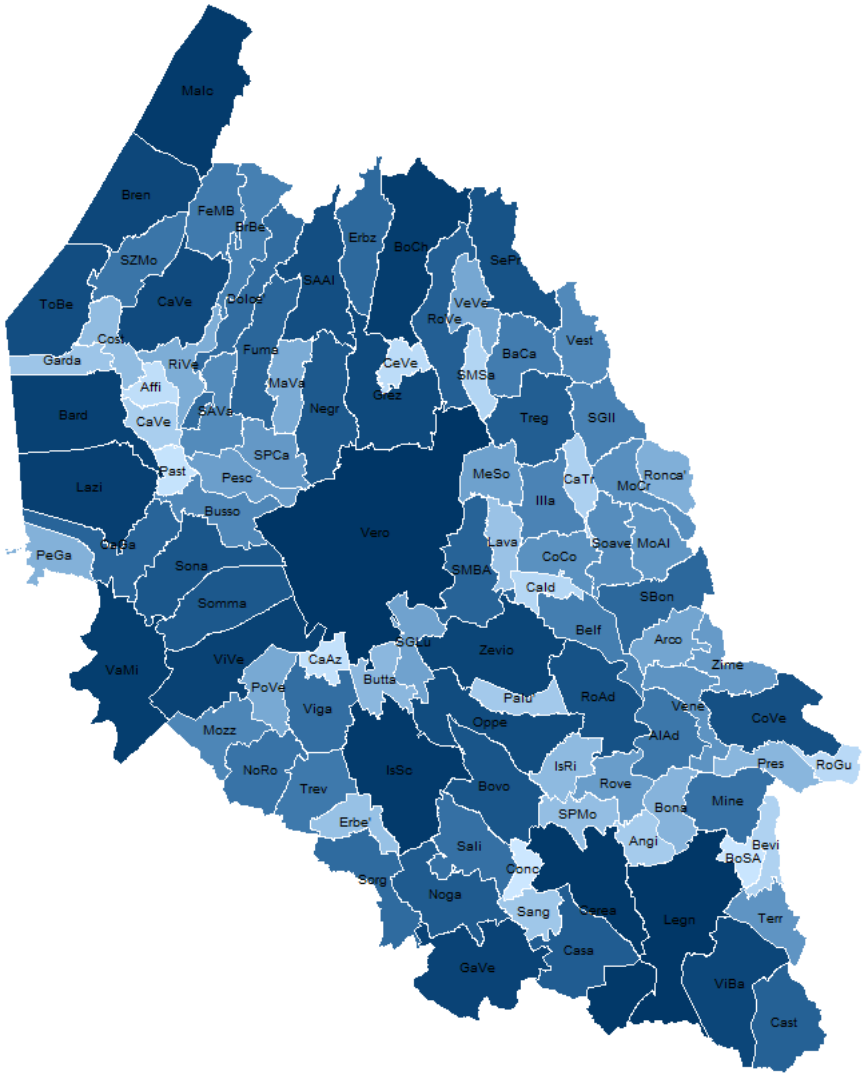


Figura 15.1: Mappa pre correzione

```
. replace x_c=x_c + 400 if cod_com==26;
(1 real change made)

. replace y_c=y_c - 1800 if cod_com==38;
(1 real change made)

. spmap sup using "`PR'_coord", id(_ID) fcolor(Blues2) clnumber(98) ocolor(white ..)
>  label(label(nom_com_abb) x(x_c) y(y_c) size(1.5) )
>  legenda(off) title("`sch'", size(*0.8));

. graph export map_post.png, replace;
(note: file map_post.png not found)
(file map_post.png written in PNG format)
```

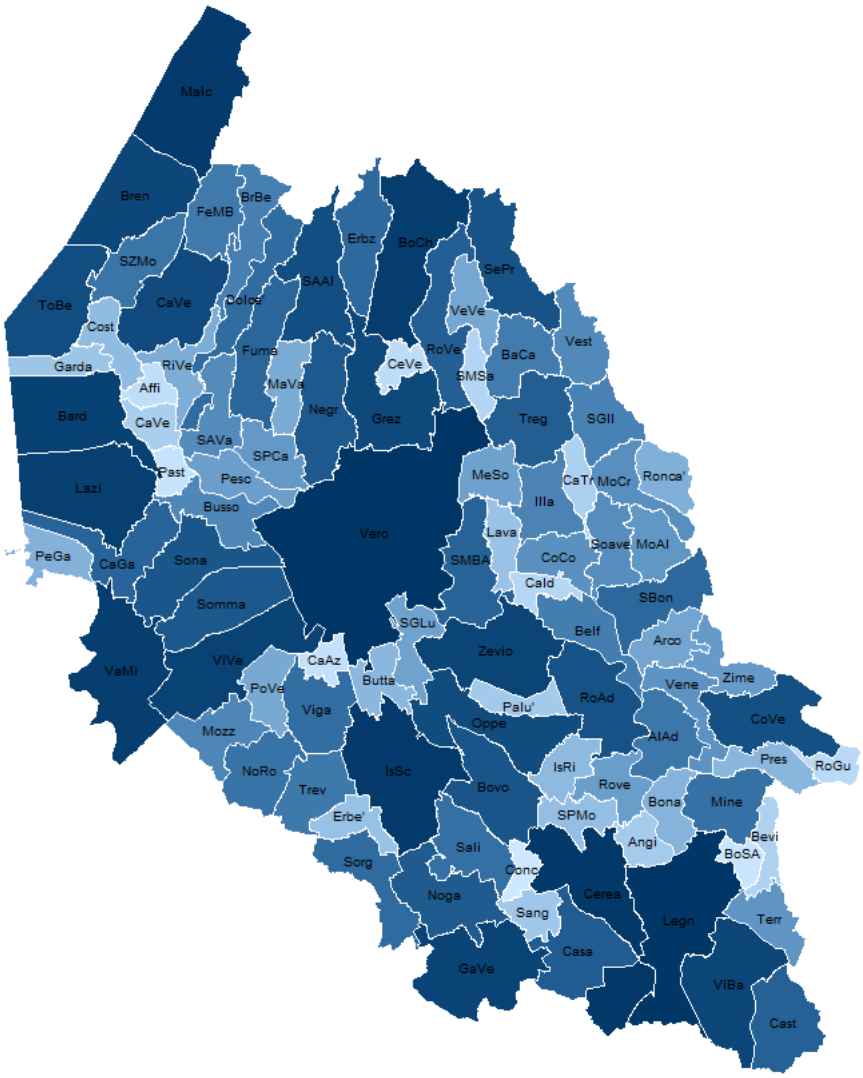


Figura 15.2: Mappa post correzione

Altro problema. Quando si rappresentano dati continui attraverso una choropleth map usando una delle combinazioni di colori previste dal programma, se c'è del testo da rappresentare ci può essere un problema di visualizzazione. Ovvero se il testo è di colore chiaro sarà difficilmente leggibile nelle aree più chiare, viceversa se il testo è di colore scuro sarà difficilmente leggibile nelle aree più scure. Potete apprezzare quanto appena detto nella figura prodotta da questo codice

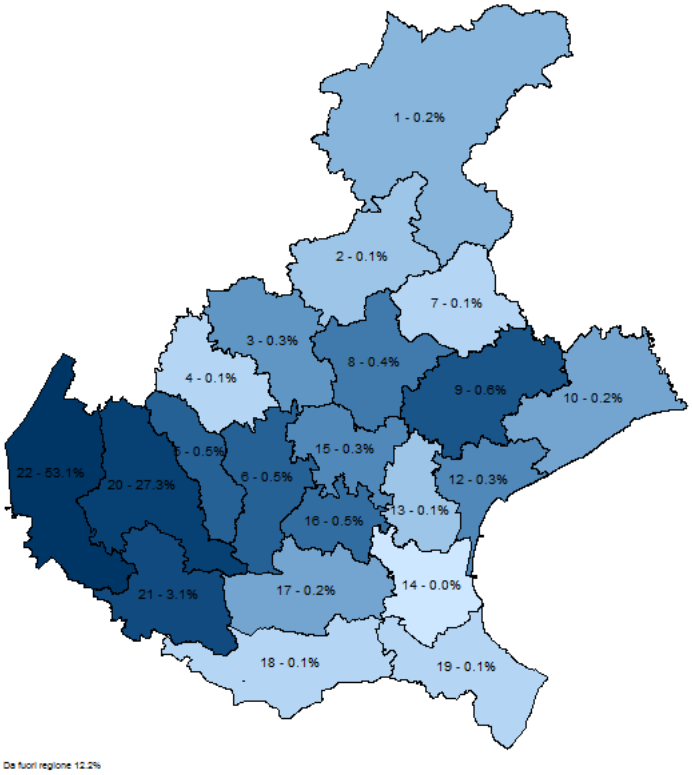


Figura 15.3: Mappa con colori predefiniti

```
. local tit : variable label pedia_od;
. spmap pedia_odp using coord_ulss.dta, id(_ID) fcolor(Blues2) ocolor(black ..)
>  clmethod(unique) label(label(pedia_odpstr) x(x_c) y(y_c) size(1.8) length(14))
>  legenda(off) note("Da fuori regione `pedia_odpFP'%", size(*0.50));
. graph export graph/ric_tot/pedia_od0.png, replace;

(file graph/ric_tot/pedia_od0.png written in PNG format)
```

le scritte in colore nero nelle aree più scure non si leggono molto bene usando la lista di colori `Blues2`. Questo accade perché il meccanismo di assegnazione dei colori attribuisce la tonalità più chiara ai valori minori e la tonalità più scura ai valori più elevati. Come ovviare? Ricorrendo ad un trucchetto che ci consenta di determinare le tonalità più chiara e più scura! Nel codice che segue determino quanti colori diversi mi servono. Per esempio sulle 22 aree da rappresentare ce ne sono 4 con valore assegnato pari a uno e che quindi avranno colore uguale. Scelgo come colore di base navy (`ocal rgb navy`) e poi

stabilisco che il colore più chiaro sarà di una intesità pari allo 0.01 di navy (`local inty =0.01`), mentre quello più scuro di 0.75 (`local INTY =0.75`). Entro questo intervallo determino le tonalità di colore necessarie per coprire gli altri valori attraverso uno passo pari a `local step = (`INTY'-`inty') / `ncl'`. Posso vedere la serie di tonalità nella local `colors`

```
. local tit : variable label pedia_od;
. tab pedia_od;

  Pediatria |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |          1        4.55        4.55
          1 |          4       18.18       22.73
          2 |          2        9.09       31.82
          3 |          1        4.55       36.36
          4 |          2        9.09       45.45
          5 |          1        4.55       50.00
          6 |          2        9.09       59.09
          7 |          1        4.55       63.64
          9 |          1        4.55       68.18
         10 |          2        9.09       77.27
         11 |          1        4.55       81.82
         59 |          1        4.55       86.36
        234 |          1        4.55       90.91
        526 |          1        4.55       95.45
       1022 |          1        4.55      100.00
------------+-----------------------------------
      Total |         22      100.00

. local colors = "";
. local rgb navy;
. local ncl = r(r) - 2;
. local INTY =0.75;
. local inty =0.01;
. local step = (`INTY'-`inty') / `ncl';
. local step = round(`step',0.01);
. forvalues c = 0(1)`ncl' ;
   local x = `inty' + `step'*`c';
   local x = round(`x',0.01);
   local colors = "`colors'" + "`rgb'*`x' ";
          ;
. di "`colors'";
navy*.01 navy*.07 navy*.13 navy*.19 navy*.25 navy*.31 navy*.37 navy*.43 navy*.49 navy*.55
>  navy*.61 navy*.67 navy*.73 navy*.79

.  spmap pedia_odp using coord_ulss.dta, id(_ID) fcolor(white `colors') ocolor(black ..) clmethod(unique)
>      label(label(pedia_odpstr) x(x_c) y(y_c) size(1.8) length(14)) legenda(off) split
>      note("Da fuori regione `pedia_odpFP'%", size(*0.50));

. graph export graph/ric_tot/pedia_od1.png, replace;
(file graph/ric_tot/pedia_od1.png written in PNG format)
```

e questo è il risultato:
In particolare le aree con numrosità pari a zero saranno bianche `fcolor(white ...`, mentre la successiva parte da un valore `navy*.01`, per passare ad un `navy*.07`, quindi a un `navy*.13` e così via.

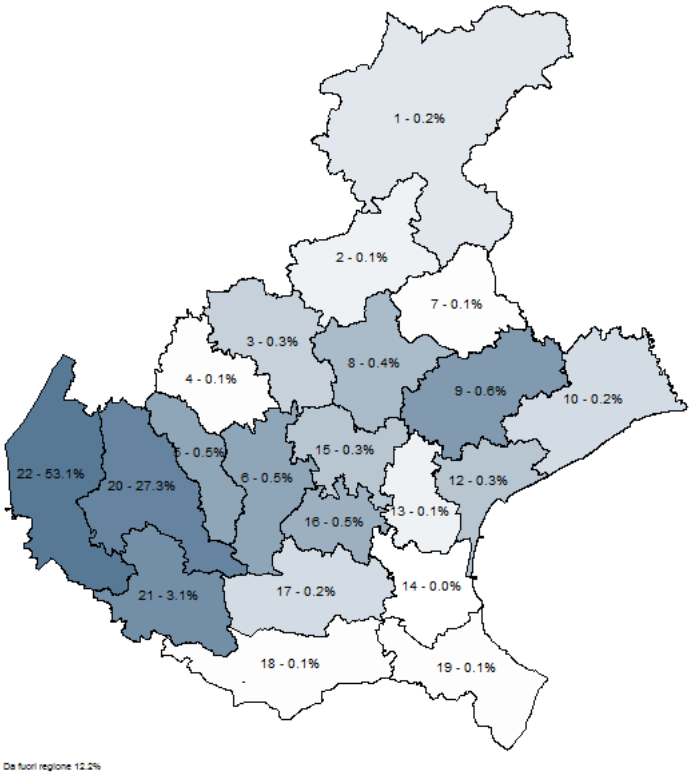


Figura 15.4: Mappa con colori assegnati

Parte II

Casi Applicati

Capitolo 16

Dataset di Grandi Dimensioni

Se la RAM a disposizione sul vostro computer è insufficiente a contenere una dataset di grandi dimensioni, la soluzione migliore è quella di spezzare il caricamento in n parti selezionando le sole variabili di interesse. Attraverso il comando `compress` e codificando per quanto possibile le variabili stringa in categoriche si riesce a recuperare ulteriori risorse.
Nel caso in esame abbiamo 2 file di dati, uno in formato testo e il suo corrispettivo in formato Stata:

```
. ls
  <dir>   8/30/07 14:17  .
  <dir>   8/30/07 14:17  ..
 770.8M   5/15/06 12:27  2002.asc
1370.2M   5/22/06 18:51  2002.dta
```

Per caricare il file 2002.asc servono circa 1400 MB di RAM. Questo file si compone di 256141 righe (una di intestazione, le altre di dati) e di 684 variabili. La strategia per bypassare il collo di bottiglia della RAM consiste nello spezzare il file in 2, mantenendo la prima linea di intestazione per entrambi. Purtroppo l'opzione *varlist* del comando `insheet` non funziona molto bene.
Oppure bisogna ricorrere al programma StatTransfer che converte i dati in maniera sequenziale senza problemi di RAM.

Per caricare invece il file 2002.dta abbiamo un maggior numero di possibilità. La prima è:

I. caricare la prima metà delle osservazioni e selezionare le variabili di interesse
II. salvare il file
III. caricare la seconda metà delle osservazioni e selezionare le variabili di interesse
IV. salvare il file
V. unire i due dataset

```
. set mem 740m

Current memory allocation
```

```
                  current                                      memory usage
    settable        value     description                    (1M = 1024k)
    ------------------------------------------------------------------------
    set maxvar       5000     max. variables allowed           1.909M
    set memory        740M    max. data space                740.000M
    set matsize       400     max. RHS vars in models          1.254M
                                                              -----------
                                                               743.163M

. use 2002.dta in 1/128070

. desc, short

Contains data from 2002.dta
  obs:       128,070
 vars:           669                          22 May 2006 18:49
 size:   718,856,910 (7.4% of memory free)
Sorted by:

. keep  h01 h03 h08 h12

. save tmp1, replace
(note: file tmp1.dta not found)
file tmp1.dta saved

. use 2002.dta in 128071/256140

. desc, short

Contains data from 2002.dta
  obs:       128,070
 vars:           669                          22 May 2006 18:49
 size:   718,856,910 (7.4% of memory free)
Sorted by:

. keep  h01 h03 h08 h12

. save tmp2, replace
(note: file tmp2.dta not found)
file tmp2.dta saved

. compress
h08 was str40 now str33
h12 was str15 now str14

. append using tmp1
h08 was str33 now str40
h12 was str14 now str15

. desc, short

Contains data from tmp2.dta
  obs:       256,140
 vars:             4                          30 Aug 2007 15:17
 size:    35,347,320 (95.4% of memory free)
Sorted by:
     Note:  dataset has changed since last saved
```

La seconda strategia invece consiste nel leggere direttamente tutte le osservazioni per le sole variabili di interesse:

```
. set mem 740m
```

```
Current memory allocation

                    current                                 memory usage
    settable          value     description                 (1M = 1024k)
    ------------------------------------------------------------------
    set maxvar         5000     max. variables allowed          1.909M
    set memory          740M    max. data space               740.000M
    set matsize         400     max. RHS vars in models         1.254M
                                                              ----------
                                                                743.163M

. use h01 h03 h08 h12 using 2002.dta, clear

. desc, short

Contains data from 2002.dta
  obs:       256,140
 vars:             4                          22 May 2006 18:49
 size:    35,347,320 (95.4% of memory free)
Sorted by:
```

Capitolo 17

Da Stringa a Numerica

17.1 Fondere variabili stringa con numeriche

Se ci si trova con due variabili che contengono la stessa informazione ma in una espressa in forma numerica e nell'altra espressa come stringa, possiamo ridurle in maniera semplice in una sola, utilizzando l'informazione della variabile stringa per costruire il `label define` per la variabile numerica. Se per esempio abbiamo due variabili con questi possibili valori:

```
cod_reg     regione
1           Piemonte
2           Valle d'Aosta
3           Lombardia
4           Trentino-Alto Adige
5           Veneto
6           Friuli-Venezia Giulia
7           Liguria
8           Emilia-Romagna
9           Toscana
10          Umbria
11          Marche
12          Lazio
13          Abruzzo
14          Molise
15          Campania
16          Puglia
17          Basilicata
18          Calabria
19          Sicilia
20          Sardegna
```

e vogliamo assegnare come label dei valori di `cod_reg` la descrizione contenuta nella variabile `regione`, possiamo, in maniera pedante, fare:

```
label define cod_reg 1 "Piemonte"
                     2 "Valle d´Aosta"
                     .....
                     20 "Sardegna";
label values cod_reg cod_reg;
```

oppure installare il comando `labutil`

```
ssc inst labutil
```

e poi

```
. tab1 regione cod_reg

-> tabulation of regione

               regione |      Freq.     Percent        Cum.
-----------------------+-----------------------------------
               Abruzzo |     61,610        3.76        3.76
            Basilicata |     26,462        1.62        5.38
              Calabria |     82,618        5.05       10.43
              Campania |    111,302        6.80       17.23
       Emilia-Romagna |     68,882        4.21       21.44
Friuli-Venezia Giulia |     44,238        2.70       24.14
                Lazio |     76,356        4.66       28.80
              Liguria |     47,470        2.90       31.70
            Lombardia |    312,696       19.10       50.81
               Marche |     49,692        3.04       53.84
              Molise |     27,472        1.68       55.52
             Piemonte |    243,612       14.88       70.41
               Puglia |     52,116        3.18       73.59
             Sardegna |     76,154        4.65       78.24
              Sicilia |     78,780        4.81       83.06
              Toscana |     57,974        3.54       86.60
  Trentino-Alto Adige |     68,478        4.18       90.78
              Umbria |     18,584        1.14       91.92
       Valle d´Aosta |     14,948        0.91       92.83
              Veneto |    117,362        7.17      100.00
-----------------------+-----------------------------------
               Total |  1,636,806      100.00

-> tabulation of cod_reg

    cod_reg |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |    243,612       14.88       14.88
          2 |     14,948        0.91       15.80
          3 |    312,696       19.10       34.90
          4 |     68,478        4.18       39.08
          5 |    117,362        7.17       46.25
          6 |     44,238        2.70       48.96
          7 |     47,470        2.90       51.86
          8 |     68,882        4.21       56.07
          9 |     57,974        3.54       59.61
         10 |     18,584        1.14       60.74
         11 |     49,692        3.04       63.78
         12 |     76,356        4.66       68.44
         13 |     61,610        3.76       72.21
         14 |     27,472        1.68       73.89
         15 |    111,302        6.80       80.69
         16 |     52,116        3.18       83.87
```

```
         17 |     26,462        1.62       85.49
         18 |     82,618        5.05       90.53
         19 |     78,780        4.81       95.35
         20 |     76,154        4.65      100.00
------------+-----------------------------------
      Total |  1,636,806      100.00

. labmask cod_reg, values(regione)

. tab cod_reg

              cod_reg |      Freq.     Percent        Cum.
----------------------+-----------------------------------
             Piemonte |    243,612       14.88       14.88
        Valle d´Aosta |     14,948        0.91       15.80
            Lombardia |    312,696       19.10       34.90
  Trentino-Alto Adige |     68,478        4.18       39.08
               Veneto |    117,362        7.17       46.25
Friuli-Venezia Giulia |     44,238        2.70       48.96
              Liguria |     47,470        2.90       51.86
       Emilia-Romagna |     68,882        4.21       56.07
              Toscana |     57,974        3.54       59.61
               Umbria |     18,584        1.14       60.74
               Marche |     49,692        3.04       63.78
                Lazio |     76,356        4.66       68.44
              Abruzzo |     61,610        3.76       72.21
               Molise |     27,472        1.68       73.89
             Campania |    111,302        6.80       80.69
               Puglia |     52,116        3.18       83.87
           Basilicata |     26,462        1.62       85.49
             Calabria |     82,618        5.05       90.53
              Sicilia |     78,780        4.81       95.35
             Sardegna |     76,154        4.65      100.00
----------------------+-----------------------------------
                Total |  1,636,806      100.00

. desc, short

Contains data from ita_82-06.dta
  obs:     1,709,390
 vars:            34                          4 Jul 2007 13:19
 size:   162,392,050 (38.1% of memory free)
Sorted by:
     Note:  dataset has changed since last saved

. drop regione

. desc, short

Contains data from ita_82-06.dta
  obs:     1,709,390
 vars:            33                          4 Jul 2007 13:19
 size:   126,494,860 (51.7% of memory free)
Sorted by:
     Note:  dataset has changed since last saved
```

semplice, veloce e pulito!. Inoltre eliminando la variabile stringa regione abbiamo ridotto il dataset di quasi 36Mb, senza perdere nessuna informazione dato che il contenuto della variabile `regione` è stato trasferito nella label di `cod_reg`.

In questo caso il vantaggio nell'utilizzo di `labmask` è relativo; costruire un `label define` per venti specificazioni non comporta un eccessivo spreco di tempo, ma pensate se dovete

fare la stessa cosa per il label delle provincie italiane (più di cento) o dei comuni italiani che sono più di ottomila!!! (io l'ho fatto e ottomila comuni sono tanti).

17.2 Da stringa a numerica categorica

Supponiamo di avere una variabile stringa (`basale`) che assume le seguenti specificazioni:

```
NO SHUNT
<10
>10 SINGLE SPIKES
>10 SHOWER O CURTAIN
```

e che vogliamo trasformarla in una variabile numerica categorica con queste assegnazioni:

```
0 per "NO SHUNT"
1 per "<10"
2 per ">10 SINGLE SPIKES"
3 per ">10 SHOWER O CURTAIN"
```

Invece di ricorrere alla costruzione

```
replace basale="0" if basale=="NO SHUNT";
replace basale="1" if basale=="<10";
replace basale="2" if basale==">10 SINGLE SPIKES";
replace basale="3" if basale==">10 SHOWER O CURTAIN";
destring basale valsalva, replace;
label define shunt 0 "no shunt"
                   1 "<10"
                   2 ">10 SINGLE SPIKES"
                   3 ">10 SHOWER O CURTAIN";
label values basale shunt;
```

possiamo fare:

```
label define shunt 0 "no shunt"
                   1 "<10"
                   2 ">10 SINGLE SPIKES"
                   3 ">10 SHOWER O CURTAIN";
encode basale, gen(basale_n) label(shunt);
```

dove in `gen()` si mette la nuova variabile numerica che verrà creata e i cui valori corrispondono a quelli definiti in `label define`. Notare che è obbligatorio creare una nuova variabile perché al momento il comando `encode` non prevede l'opzione `replace`.

Capitolo 18

Liste di Files e Directory

Il problema da risolvere è l'acquisizione e la riunione in un unico dataset delle informazioni contenute in un elevato numero di files. In questo caso abbiamo le venti regioni italiane

```
. dir, wide
  <dir> .                  <dir> ..                 <dir> abruzzo
  <dir> basilicata         <dir> calabria           <dir> campania
  <dir> emilia_romagna     <dir> friuli             <dir> lazio
  <dir> liguria            <dir> lombardia          <dir> marche
  <dir> molise             <dir> piemonte           <dir> puglia
  <dir> sardegna           <dir> sicilia            <dir> toscana
  <dir> trentino           <dir> umbria             <dir> vda
  <dir> veneto
```

All'interno di ciascuna regione abbiamo un cartella per ciascuna provincia di quella regione:

```
. cd abruzzo
G:\projects\popolazione\pop_res\com_82-01\abruzzo

. dir
  <dir>   7/04/07 10:36  .
  <dir>   7/04/07 10:36  ..
  <dir>   7/04/07 10:35  chieti
  <dir>   7/04/07 10:36  laquila
  <dir>   7/04/07 10:36  pescara
  <dir>   7/04/07 10:36  teramo
```

All'interno di ciascuna cartella delle provincie, una cartella `dati` che contiene 2 tipi di dati:

- una serie di files con estensione .TXT, con dati in formato testo delimitati da virgola. Per ogni comune della provincia c'è un file che contiene i dati inerenti alle femmine (*_F.TXT) e un file con i dati inerenti i maschi (*_M.TXT).
- una serie di files con estensione .csv, con dati in formato testo delimitati da virgola. In questo caso c'è un unico file per ciascun anno dal dal 1992 al 2001 con i dati sia dei maschi che delle femmine.

```
. cd chieti/dati
```

```
G:\projects\popolazione\pop_res\com_82-01\abruzzo\chieti\dati

. dir, wide
  <dir> .                  <dir> ..                 4.2k 069001_F.TXT
   4.2k 069001_M.TXT        4.2k 069002_F.TXT       4.2k 069002_M.TXT
   3.8k 069003_F.TXT        3.8k 069003_M.TXT       3.7k 069004_F.TXT
   3.7k 069004_M.TXT        4.3k 069005_F.TXT       4.3k 069005_M.TXT
   3.7k 069006_F.TXT        3.6k 069006_M.TXT       3.5k 069007_F.TXT
   3.5k 069007_M.TXT        4.3k 069008_F.TXT       4.3k 069008_M.TXT
   3.5k 069009_F.TXT        3.5k 069009_M.TXT       3.8k 069010_F.TXT
   3.8k 069010_M.TXT        3.5k 069011_F.TXT       3.5k 069011_M.TXT
   3.6k 069012_F.TXT        3.6k 069012_M.TXT       3.8k 069013_F.TXT
   3.7k 069013_M.TXT        3.7k 069014_F.TXT       3.7k 069014_M.TXT
   4.3k 069015_F.TXT        4.3k 069015_M.TXT       4.2k 069016_F.TXT
   4.2k 069016_M.TXT        4.3k 069017_F.TXT       4.3k 069017_M.TXT
   4.3k 069018_F.TXT        4.3k 069018_M.TXT       3.5k 069019_F.TXT
   3.5k 069019_M.TXT        4.2k 069020_F.TXT       4.2k 069020_M.TXT
   3.7k 069021_F.TXT        3.7k 069021_M.TXT       5.2k 069022_F.TXT
   5.1k 069022_M.TXT        3.5k 069023_F.TXT       3.5k 069023_M.TXT
   3.7k 069024_F.TXT        3.6k 069024_M.TXT       3.5k 069025_F.TXT
   3.5k 069025_M.TXT        3.5k 069026_F.TXT       3.5k 069026_M.TXT
   4.3k 069027_F.TXT        4.3k 069027_M.TXT       4.3k 069028_F.TXT
   4.3k 069028_M.TXT        3.5k 069029_F.TXT       3.5k 069029_M.TXT
   4.0k 069030_F.TXT        3.9k 069030_M.TXT       4.0k 069031_F.TXT
   3.9k 069031_M.TXT        3.7k 069032_F.TXT       3.6k 069032_M.TXT
   4.3k 069033_F.TXT        4.3k 069033_M.TXT       3.5k 069034_F.TXT
   3.5k 069034_M.TXT        4.9k 069035_F.TXT       4.9k 069035_M.TXT
   3.7k 069036_F.TXT        3.7k 069036_M.TXT       4.1k 069037_F.TXT
   4.0k 069037_M.TXT        3.8k 069038_F.TXT       3.8k 069038_M.TXT
   3.5k 069039_F.TXT        3.5k 069039_M.TXT       4.0k 069040_F.TXT
   4.0k 069040_M.TXT        4.3k 069041_F.TXT       4.3k 069041_M.TXT
   3.7k 069042_F.TXT        3.7k 069042_M.TXT       4.3k 069043_F.TXT
   4.3k 069043_M.TXT        3.5k 069044_F.TXT       3.5k 069044_M.TXT
   3.8k 069045_F.TXT        3.8k 069045_M.TXT       5.1k 069046_F.TXT
   5.1k 069046_M.TXT        3.5k 069047_F.TXT       3.5k 069047_M.TXT
   3.5k 069048_F.TXT        3.5k 069048_M.TXT       3.6k 069049_F.TXT
   3.6k 069049_M.TXT        4.3k 069050_F.TXT       4.3k 069050_M.TXT
   3.7k 069051_F.TXT        3.7k 069051_M.TXT       3.5k 069052_F.TXT
   3.5k 069052_M.TXT        3.5k 069053_F.TXT       3.5k 069053_M.TXT
   3.6k 069054_F.TXT        3.6k 069054_M.TXT       4.1k 069055_F.TXT
   4.1k 069055_M.TXT        4.0k 069056_F.TXT       4.0k 069056_M.TXT
   4.3k 069057_F.TXT        4.3k 069057_M.TXT       5.1k 069058_F.TXT
   5.0k 069058_M.TXT        4.3k 069059_F.TXT       4.3k 069059_M.TXT
   3.9k 069060_F.TXT        3.8k 069060_M.TXT       3.8k 069061_F.TXT
   3.8k 069061_M.TXT        3.7k 069062_F.TXT       3.7k 069062_M.TXT
   3.5k 069063_F.TXT        3.5k 069063_M.TXT       3.5k 069064_F.TXT
   3.5k 069064_M.TXT        3.9k 069065_F.TXT       3.9k 069065_M.TXT
   3.8k 069066_F.TXT        3.7k 069066_M.TXT       3.6k 069067_F.TXT
   3.6k 069067_M.TXT        4.2k 069068_F.TXT       4.2k 069068_M.TXT
   3.7k 069069_F.TXT        3.6k 069069_M.TXT       3.6k 069070_F.TXT
   3.6k 069070_M.TXT        3.9k 069071_F.TXT       3.9k 069071_M.TXT
   4.3k 069072_F.TXT        4.3k 069072_M.TXT       4.0k 069073_F.TXT
   4.0k 069073_M.TXT        4.2k 069074_F.TXT       4.2k 069074_M.TXT
   3.9k 069075_F.TXT        3.8k 069075_M.TXT       4.1k 069076_F.TXT
   4.1k 069076_M.TXT        3.5k 069077_F.TXT       3.5k 069077_M.TXT
   3.5k 069078_F.TXT        3.5k 069078_M.TXT       3.8k 069079_F.TXT
   3.7k 069079_M.TXT        3.5k 069080_F.TXT       3.5k 069080_M.TXT
   4.3k 069081_F.TXT        4.3k 069081_M.TXT       3.6k 069082_F.TXT
   3.5k 069082_M.TXT        4.7k 069083_F.TXT       4.7k 069083_M.TXT
   3.7k 069084_F.TXT        3.7k 069084_M.TXT       4.2k 069085_F.TXT
   4.2k 069085_M.TXT        4.3k 069086_F.TXT       4.3k 069086_M.TXT
   4.3k 069087_F.TXT        4.3k 069087_M.TXT       4.1k 069088_F.TXT
   4.0k 069088_M.TXT        3.5k 069089_F.TXT       3.5k 069089_M.TXT
   4.3k 069090_F.TXT        4.3k 069090_M.TXT       4.3k 069091_F.TXT
   4.3k 069091_M.TXT        4.1k 069092_F.TXT       4.0k 069092_M.TXT
```

```
  3.8k 069093_F.TXT         3.8k 069093_M.TXT         4.2k 069094_F.TXT
  4.2k 069094_M.TXT         4.0k 069095_F.TXT         3.9k 069095_M.TXT
  3.6k 069096_F.TXT         3.5k 069096_M.TXT         3.5k 069097_F.TXT
  3.5k 069097_M.TXT         3.9k 069098_F.TXT         3.9k 069098_M.TXT
  5.1k 069099_F.TXT         5.0k 069099_M.TXT         3.6k 069100_F.TXT
  3.6k 069100_M.TXT         4.2k 069101_F.TXT         4.1k 069101_M.TXT
  3.9k 069102_F.TXT         3.8k 069102_M.TXT         3.5k 069103_F.TXT
  3.5k 069103_M.TXT         3.5k 069104_F.TXT         3.5k 069104_M.TXT
 59.2k ch1992.csv          59.2k ch1993.csv          59.2k ch1994.csv
 59.2k ch1995.csv          59.2k ch1996.csv          59.2k ch1997.csv
 59.1k ch1998.csv          59.1k ch1999.csv          59.1k ch2000.csv
 59.0k ch2001.csv
```

per un totale di 16172 files .TXT e 1030 files .csv. Usare il comando `insheet` scrivendo il nome di tutti i files .TXT e .csv è la soluzione adottata da chi dispone di molto tempo ed è veloce nella digitazione. Io che non ho il primo e sono scarso nella seconda preferisco agire così.

Per prima cosa acquisisco all'interno di una `local` i nomi di tutti i files in formato .TXT:

```
. local files: dir . files "*.txt"

. di `"`files'"'
"069001_f.txt" "069001_m.txt" "069002_f.txt" "069002_m.txt" "069003_f.txt" "0690
> 03_m.txt" "069004_f.txt" "069004_m.txt" "069005_f.txt" "069005_m.txt" "069006_
> f.txt" "069006_m.txt" "069007_f.txt" "069007_m.txt" "069008_f.txt" "069008_m.t
> xt" "069009_f.txt" "069009_m.txt" "069010_f.txt" "069010_m.txt" "069011_f.txt"
>  "069011_m.txt" "069012_f.txt" "069012_m.txt" "069013_f.txt" "069013_m.txt" "0
> 69014_f.txt" "069014_m.txt" "069015_f.txt" "069015_m.txt" "069016_f.txt" "0690
> 16_m.txt" "069017_f.txt" "069017_m.txt" "069018_f.txt" "069018_m.txt" "069019_
> f.txt" "069019_m.txt" "069020_f.txt" "069020_m.txt" "069021_f.txt" "069021_m.t
> xt" "069022_f.txt" "069022_m.txt" "069023_f.txt" "069023_m.txt" "069024_f.txt"
>  "069024_m.txt" "069025_f.txt" "069025_m.txt" "069026_f.txt" "069026_m.txt" "0
> 69027_f.txt" "069027_m.txt" "069028_f.txt" "069028_m.txt" "069029_f.txt" "0690
> 29_m.txt" "069030_f.txt" "069030_m.txt" "069031_f.txt" "069031_m.txt" "069032_
> f.txt" "069032_m.txt" "069033_f.txt" "069033_m.txt" "069034_f.txt" "069034_m.t
> xt" "069035_f.txt" "069035_m.txt" "069036_f.txt" "069036_m.txt" "069037_f.txt"
>  "069037_m.txt" "069038_f.txt" "069038_m.txt" "069039_f.txt" "069039_m.txt" "0
> 69040_f.txt" "069040_m.txt" "069041_f.txt" "069041_m.txt" "069042_f.txt" "0690
> 42_m.txt" "069043_f.txt" "069043_m.txt" "069044_f.txt" "069044_m.txt" "069045_
> f.txt" "069045_m.txt" "069046_f.txt" "069046_m.txt" "069047_f.txt" "069047_m.t
> xt" "069048_f.txt" "069048_m.txt" "069049_f.txt" "069049_m.txt" "069050_f.txt"
>  "069050_m.txt" "069051_f.txt" "069051_m.txt" "069052_f.txt" "069052_m.txt" "0
> 69053_f.txt" "069053_m.txt" "069054_f.txt" "069054_m.txt" "069055_f.txt" "0690
> 55_m.txt" "069056_f.txt" "069056_m.txt" "069057_f.txt" "069057_m.txt" "069058_
> f.txt" "069058_m.txt" "069059_f.txt" "069059_m.txt" "069060_f.txt" "069060_m.t
> xt" "069061_f.txt" "069061_m.txt" "069062_f.txt" "069062_m.txt" "069063_f.txt"
>  "069063_m.txt" "069064_f.txt" "069064_m.txt" "069065_f.txt" "069065_m.txt" "0
> 69066_f.txt" "069066_m.txt" "069067_f.txt" "069067_m.txt" "069068_f.txt" "0690
> 68_m.txt" "069069_f.txt" "069069_m.txt" "069070_f.txt" "069070_m.txt" "069071_
> f.txt" "069071_m.txt" "069072_f.txt" "069072_m.txt" "069073_f.txt" "069073_m.t
> xt" "069074_f.txt" "069074_m.txt" "069075_f.txt" "069075_m.txt" "069076_f.txt"
>  "069076_m.txt" "069077_f.txt" "069077_m.txt" "069078_f.txt" "069078_m.txt" "0
> 69079_f.txt" "069079_m.txt" "069080_f.txt" "069080_m.txt" "069081_f.txt" "0690
> 81_m.txt" "069082_f.txt" "069082_m.txt" "069083_f.txt" "069083_m.txt" "069084_
> f.txt" "069084_m.txt" "069085_f.txt" "069085_m.txt" "069086_f.txt" "069086_m.t
> xt" "069087_f.txt" "069087_m.txt" "069088_f.txt" "069088_m.txt" "069089_f.txt"
>  "069089_m.txt" "069090_f.txt" "069090_m.txt" "069091_f.txt" "069091_m.txt" "0
> 69092_f.txt" "069092_m.txt" "069093_f.txt" "069093_m.txt" "069094_f.txt" "0690
> 94_m.txt" "069095_f.txt" "069095_m.txt" "069096_f.txt" "069096_m.txt" "069097_
> f.txt" "069097_m.txt" "069098_f.txt" "069098_m.txt" "069099_f.txt" "069099_m.t
> xt" "069100_f.txt" "069100_m.txt" "069101_f.txt" "069101_m.txt" "069102_f.txt"
>  "069102_m.txt" "069103_f.txt" "069103_m.txt" "069104_f.txt" "069104_m.txt"
```

La costruzione `local files: dir . files "*.txt"` rientra nelle funzioni estese delle macro di Stata ([P] **macro**)[1].
A questo punto per poter fare l'`append` dei dati devo partire con il primo file, salvarlo e poi fare l'`append` dei successivi file. Per fare ciò, estraggo dalla `local` che ho chiamato `files` il suo primo elemento, lo assegno alla `local` che chiamerò `primo` e contestualmente lo tolgo dalla `local files` per non leggere due volte lo stesso file di dati `` ` ' ``

```
. local primo : word 1 of `files'  /* primo elemento di `files' */

. di "`primo'"
069001_f.txt

. local files : list files - primo /* tolgo da `files' il suo primo elemento */
```

Anche la costruzione `local primo : word 1 of `files'` appartiene alle funzioni estese delle macro di Stata. A questo punto leggo i dati del primo file (quello indicato dalla `local primo`), genero la variabile `sex`, le assegno valore 2 e lo salvo in un file temporaneo:

```
. insheet using `primo', clear
(14 vars, 87 obs)

. gen sex=2

. save temp, replace
(note: file temp.dta not found)
file temp.dta saved
```

Ora posso leggere e fare l'`append` in sequenza di tutti gli altri file indicati nella `local files`. Devo però distinguere i files con dati riferiti alle femmine dai files con dati riferiti ai maschi. Come? Se nel nome del file c'è _m saranno dati riferiti a maschi e quindi assegnerò valore uno alla variabile `sex`, se c'è _f saranno dati riferiti alle femmine e quindi assegnerò valore due alla variabile `sex`. La costruzione `local meccia = strmatch("`f'","*_m*")` mi permette di distinguere tra le 2 possibilità e quindi di agire di conseguenza sul valore da assegnare alla variabile `sex`.

```
foreach f in `files'  {;
   insheet using dati/`f', clear;
   local meccia = strmatch("`f'","*_m*");
   if `meccia'==1 {;  /* se trova _m nel nome del file */
      gen sex=1;    /* assegna a sex il valore 1 */
   };
   else if `meccia'==0 {;  /* altrimenti */
      gen sex=2;   /* assegna a sex il valore 2 */
   };

   append using temp;
   save temp, replace;
};

. summ

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
          v1 |         0
          v2 |     18096     69052.5    30.02166      69001      69104
          v3 |         0
```

[1] Notare la particolare sintassi usata per fare il display della `local files`!!

```
          v4 |     18096    43.14943    25.40616          0         99
          v5 |     18096    40.94828    437.5124          0      29005
-------------+---------------------------------------------------------
          v6 |     18096    41.16534    441.2414          0      29067
          v7 |     18096    41.30836    444.0427          0      29125
          v8 |     18096    41.46795    446.7315          0      29183
          v9 |     18096    41.58709     449.089          0      29241
         v10 |     18096    41.66225    450.6736          0      29273
-------------+---------------------------------------------------------
         v11 |     18096    41.75696    452.6105          0      29311
         v12 |     18096    41.91943    455.5657          0      29404
         v13 |     18096    42.05449    458.2125          0      29479
         v14 |     18096     42.1408    460.1403          0      29603
         sex |     18096         1.5    .5000138          1          2
```

Bene, risolto il problema dei files; ma vorrei potermi muovere anche tra le venti cartelle delle regioni e tra le cartelle delle provincie sfruttando lo stesso meccanismo. No problem, basta sfruttare la funzione `dir . dirs`:

```
. local regioni :  dir . dirs "*";

. di `"`regioni'"';
"abruzzo" "basilicata" "calabria" "campania" "emilia_romagna" "friuli" "lazio"
"liguria" "lombardia" "marche" "molise" "piemonte" "puglia" "sardegna" "sicilia"
"toscana" "trentino" "umbria" "vda" "veneto"
```

Adesso facciamo in modo di entrare in ciascuna cartella delle regione e lanciare il file `prov.do` in questo modo:

```
foreach regio in `regioni' {;
cd `regio';
do prov.do;
cd  ..;
};
```

Il file `prov.do` a sua volta raccoglie l'elenco delle cartelle delle provincie e lancia il file `read.do` che si occupa di leggere i dati dai files .TXT e .csv.

```
local diry :  dir . dirs "*";
di `"`diry'"';
"chieti" "laquila" "pescara" "teramo"

foreach prv in  `diry' {;
cd `prv';
do read.do;
cd  ..;
};
```

Capitolo 19

Previsioni Multiple

19.1 Introduzione

Spiego brevemente il contesto e il problema che si vuole risolvere. Si vuole stimare attraverso un modello probit la probabilità di separazione di un gruppo di coppie in base ad un ridotto numero di variabili esplicative. In realtà le variabili esplicative sarebbero molto più numerose ma in questo esempio utilizziamo solo quelle necessarie per produrre un insieme di probabilità di separazione data da tutte le possibili combinazioni delle variabili esplicative. Per fare ciò ricorreremo al comando `contract`, ad un ciclo `foreach`, a matrici e al comando aggiuntivo `prchange` implementato da J. Scott Long e Jeremy Freese.

19.2 La stima del modello

Come già detto stimiamo un versione ridotta del modello. Vediamo una descrizione delle variabili utilizzate e il risultato della stima probit:

```
. summ separazione higher_edu_M_l higher_edu_F_l
>      du_dip_M du_self_M du_nw_F du_dip_F;

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
 separazione |      4865    .0341213    .1815593          0          1
higher_e~M_l |      4810    .1039501    .3052275          0          1
higher_e~F_l |      4830    .0917184    .2886579          0          1
    du_dip_M |      4845     .580805    .4934783          0          1
   du_self_M |      4845    .2637771    .4407253          0          1
-------------+--------------------------------------------------------
     du_nw_F |      4859    .5330315    .4989591          0          1
    du_dip_F |      4859     .370035    .4828634          0          1

. probit separazione sons_M_l higher_edu_M_l higher_edu_F_l du_dip_M
>        du_self_M du_nw_F du_dip_F;

Iteration 0:   log likelihood = -721.19595
Iteration 1:   log likelihood = -702.70504
Iteration 2:   log likelihood =  -702.3606
Iteration 3:   log likelihood = -702.36044
```

```
Probit regression                                 Number of obs   =       4789
                                                  LR chi2(7)      =      37.67
                                                  Prob > chi2     =     0.0000
Log likelihood = -702.36044                       Pseudo R2       =     0.0261

------------------------------------------------------------------------------
 separazione |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
    sons_M_l |  -.1462057   .0385635    -3.79   0.000    -.2217886   -.0706227
higher_e~M_l |   .3278412   .1107032     2.96   0.003     .1108669    .5448156
higher_e~F_l |  -.1637689   .1314967    -1.25   0.213    -.4214978    .0939599
    du_dip_M |   .0943842   .1120967     0.84   0.400    -.1253214    .3140898
   du_self_M |   .2067016   .1204811     1.72   0.086    -.0294371    .4428402
     du_nw_F |  -.1692472   .1189374    -1.42   0.155    -.4023601    .0638657
    du_dip_F |   .0184736   .1186803     0.16   0.876    -.2141355    .2510828
       _cons |  -1.693473   .1491936   -11.35   0.000    -1.985887   -1.401059
------------------------------------------------------------------------------
```

La combinazione delle dummy `du_dip_M` e `du_self_M` individua questi tre profili occupazionali per l'uomo:

1. No Working (NW) se `du_dip_M==0 & du_self_M==0`
2. Employed (E) se `du_dip_M==1 & du_self_M==0`
3. Self-Employed (SE) se `du_dip_M==0 & du_self_M==1`

La combinazione delle dummy `du_nw_F` e `du_dip_F` individua questi tre profili occupazionali per la donna:

1. No Working (NW) se `du_nw_F==1 & du_dip_F==0`
2. Employed (E) se `du_nw_F==0 & du_dip_F==1`
3. Self-Employed (SE) se `du_nw_F==0 & du_dip_F==0`

Ora modifichiamo il dataset in modo da ottenere una osservazione per ciascuna possibile combinazione dei valori delle nostre variabili esplicative. Dato che

- `sons_M_l` assume 4 valori diversi (0,1,2,3)
- `higher_edu_M_l` assume 2 valori (0,1)
- `higher_edu_F_l` assume 2 valori (0,1)
- `du_dip_M` assume 2 valori (0,1)
- `du_self_M` assume 2 valori (0,1)
- `du_nw_F` assume 2 valori (0,1)
- `du_dip_F` assume 2 valori (0,1)

Alla fine devo ottenere un dataset con $4 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 256$ osservazioni. Si tenga presente anche che nel dataset non esistono tutte queste combinazioni (`_freq` = 0). Posso ottenere quanto mi prefiggo tramite il comando `contract` come segue

```
. contract sons_M_l higher_edu_M_l higher_edu_F_l du_dip_M
>          du_self_M du_nw_F du_dip_F, nomiss zero;

. clist
```

	sons_M_l	highe~M_l	highe~F_l	du_dip_M	du_sel~M	du_dip_F	du_nw_F	_freq
1.	0	0	0	0	0	0	0	8
2.	0	0	0	0	0	1	0	26
3.	0	0	0	0	0	0	1	105
4.	0	0	0	0	0	1	1	0
5.	0	0	0	0	1	0	0	39
6.	0	0	0	0	1	1	0	77
7.	0	0	0	0	1	0	1	60
8.	0	0	0	0	1	1	1	0
9.	0	0	0	1	0	0	0	31
10.	0	0	0	1	0	1	0	222
11.	0	0	0	1	0	0	1	162
12.	0	0	0	1	0	1	1	0
13.	0	0	0	1	1	0	0	0
14.	0	0	0	1	1	1	0	0
15.	0	0	0	1	1	0	1	0
16.	0	0	0	1	1	1	1	0
17.	0	0	1	0	0	0	0	1
18.	0	0	1	0	0	1	0	1
19.	0	0	1	0	0	0	1	0
20.	0	0	1	0	0	1	1	0
21.	0	0	1	0	1	0	0	7
22.	0	0	1	0	1	1	0	7
23.	0	0	1	0	1	0	1	3
24.	0	0	1	0	1	1	1	0
25.	0	0	1	1	0	0	0	4
26.	0	0	1	1	0	1	0	22
27.	0	0	1	1	0	0	1	10
28.	0	0	1	1	0	1	1	0
29.	0	0	1	1	1	0	0	0
30.	0	0	1	1	1	1	0	0
31.	0	0	1	1	1	0	1	0
32.	0	0	1	1	1	1	1	0
33.	0	1	0	0	0	0	0	1
34.	0	1	0	0	0	1	0	2
35.	0	1	0	0	0	0	1	1
36.	0	1	0	0	0	1	1	0
37.	0	1	0	0	1	0	0	4
38.	0	1	0	0	1	1	0	3
39.	0	1	0	0	1	0	1	8
40.	0	1	0	0	1	1	1	0
41.	0	1	0	1	0	0	0	2
42.	0	1	0	1	0	1	0	15
43.	0	1	0	1	0	0	1	8
44.	0	1	0	1	0	1	1	0
45.	0	1	0	1	1	0	0	0
46.	0	1	0	1	1	1	0	0
47.	0	1	0	1	1	0	1	0
48.	0	1	0	1	1	1	1	0
49.	0	1	1	0	0	0	0	0
50.	0	1	1	0	0	1	0	0
51.	0	1	1	0	0	0	1	0
52.	0	1	1	0	0	1	1	0
53.	0	1	1	0	1	0	0	1
54.	0	1	1	0	1	1	0	8
55.	0	1	1	0	1	0	1	1
56.	0	1	1	0	1	1	1	0
57.	0	1	1	1	0	0	0	6
58.	0	1	1	1	0	1	0	13
59.	0	1	1	1	0	0	1	8
60.	0	1	1	1	0	1	1	0
61.	0	1	1	1	1	0	0	0
62.	0	1	1	1	1	1	0	0
63.	0	1	1	1	1	0	1	0
64.	0	1	1	1	1	1	1	0

65.	1	0	0	0	0	0	0	20
66.	1	0	0	0	0	1	0	57
67.	1	0	0	0	0	0	1	180
68.	1	0	0	0	0	1	1	0
69.	1	0	0	0	1	0	0	73
70.	1	0	0	0	1	1	0	89
71.	1	0	0	0	1	0	1	167
72.	1	0	0	0	1	1	1	0
73.	1	0	0	1	0	0	0	33
74.	1	0	0	1	0	1	0	335
75.	1	0	0	1	0	0	1	405
76.	1	0	0	1	0	1	1	0
77.	1	0	0	1	1	0	0	0
78.	1	0	0	1	1	1	0	0
79.	1	0	0	1	1	0	1	0
80.	1	0	0	1	1	1	1	0
81.	1	0	1	0	0	0	0	0
82.	1	0	1	0	0	1	0	1
83.	1	0	1	0	0	0	1	3
84.	1	0	1	0	0	1	1	0
85.	1	0	1	0	1	0	0	6
86.	1	0	1	0	1	1	0	12
87.	1	0	1	0	1	0	1	7
88.	1	0	1	0	1	1	1	0
89.	1	0	1	1	0	0	0	2
90.	1	0	1	1	0	1	0	33
91.	1	0	1	1	0	0	1	10
92.	1	0	1	1	0	1	1	0
93.	1	0	1	1	1	0	0	0
94.	1	0	1	1	1	1	0	0
95.	1	0	1	1	1	0	1	0
96.	1	0	1	1	1	1	1	0
97.	1	1	0	0	0	0	0	0
98.	1	1	0	0	0	1	0	1
99.	1	1	0	0	0	0	1	0
100.	1	1	0	0	0	1	1	0
101.	1	1	0	0	1	0	0	3
102.	1	1	0	0	1	1	0	14
103.	1	1	0	0	1	0	1	11
104.	1	1	0	0	1	1	1	0
105.	1	1	0	1	0	0	0	4
106.	1	1	0	1	0	1	0	31
107.	1	1	0	1	0	0	1	20
108.	1	1	0	1	0	1	1	0
109.	1	1	0	1	1	0	0	0
110.	1	1	0	1	1	1	0	0
111.	1	1	0	1	1	0	1	0
112.	1	1	0	1	1	1	1	0
113.	1	1	1	0	0	0	0	1
114.	1	1	1	0	0	1	0	0
115.	1	1	1	0	0	0	1	1
116.	1	1	1	0	0	1	1	0
117.	1	1	1	0	1	0	0	6
118.	1	1	1	0	1	1	0	9
119.	1	1	1	0	1	0	1	8
120.	1	1	1	0	1	1	1	0
121.	1	1	1	1	0	0	0	4
122.	1	1	1	1	0	1	0	40
123.	1	1	1	1	0	0	1	17
124.	1	1	1	1	0	1	1	0
125.	1	1	1	1	1	0	0	0
126.	1	1	1	1	1	1	0	0
127.	1	1	1	1	1	0	1	0
128.	1	1	1	1	1	1	1	0
129.	2	0	0	0	0	0	0	16

130.	2	0	0	0	0	1	0	54
131.	2	0	0	0	0	0	1	174
132.	2	0	0	0	0	1	1	0
133.	2	0	0	0	1	0	0	72
134.	2	0	0	0	1	1	0	84
135.	2	0	0	0	1	0	1	232
136.	2	0	0	0	1	1	1	0
137.	2	0	0	1	0	0	0	39
138.	2	0	0	1	0	1	0	302
139.	2	0	0	1	0	0	1	521
140.	2	0	0	1	0	1	1	0
141.	2	0	0	1	1	0	0	0
142.	2	0	0	1	1	1	0	0
143.	2	0	0	1	1	0	1	0
144.	2	0	0	1	1	1	1	0
145.	2	0	1	0	0	0	0	0
146.	2	0	1	0	0	1	0	2
147.	2	0	1	0	0	0	1	1
148.	2	0	1	0	0	1	1	0
149.	2	0	1	0	1	0	0	4
150.	2	0	1	0	1	1	0	14
151.	2	0	1	0	1	0	1	4
152.	2	0	1	0	1	1	1	0
153.	2	0	1	1	0	0	0	1
154.	2	0	1	1	0	1	0	33
155.	2	0	1	1	0	0	1	8
156.	2	0	1	1	0	1	1	0
157.	2	0	1	1	1	0	0	0
158.	2	0	1	1	1	1	0	0
159.	2	0	1	1	1	0	1	0
160.	2	0	1	1	1	1	1	0
161.	2	1	0	0	0	0	0	0
162.	2	1	0	0	0	1	0	6
163.	2	1	0	0	0	0	1	3
164.	2	1	0	0	0	1	1	0
165.	2	1	0	0	1	0	0	6
166.	2	1	0	0	1	1	0	13
167.	2	1	0	0	1	0	1	7
168.	2	1	0	0	1	1	1	0
169.	2	1	0	1	0	0	0	3
170.	2	1	0	1	0	1	0	44
171.	2	1	0	1	0	0	1	29
172.	2	1	0	1	0	1	1	0
173.	2	1	0	1	1	0	0	0
174.	2	1	0	1	1	1	0	0
175.	2	1	0	1	1	0	1	0
176.	2	1	0	1	1	1	1	0
177.	2	1	1	0	0	0	0	0
178.	2	1	1	0	0	1	0	0
179.	2	1	1	0	0	0	1	1
180.	2	1	1	0	0	1	1	0
181.	2	1	1	0	1	0	0	7
182.	2	1	1	0	1	1	0	11
183.	2	1	1	0	1	0	1	7
184.	2	1	1	0	1	1	1	0
185.	2	1	1	1	0	0	0	5
186.	2	1	1	1	0	1	0	39
187.	2	1	1	1	0	0	1	5
188.	2	1	1	1	0	1	1	0
189.	2	1	1	1	1	0	0	0
190.	2	1	1	1	1	1	0	0
191.	2	1	1	1	1	0	1	0
192.	2	1	1	1	1	1	1	0
193.	3	0	0	0	0	0	0	6
194.	3	0	0	0	0	1	0	17

195.	3	0	0	0	0	0	1	50
196.	3	0	0	0	0	1	1	0
197.	3	0	0	0	1	0	0	38
198.	3	0	0	0	1	1	0	24
199.	3	0	0	0	1	0	1	102
200.	3	0	0	0	1	1	1	0
201.	3	0	0	1	0	0	0	5
202.	3	0	0	1	0	1	0	71
203.	3	0	0	1	0	0	1	186
204.	3	0	0	1	0	1	1	0
205.	3	0	0	1	1	0	0	0
206.	3	0	0	1	1	1	0	0
207.	3	0	0	1	1	0	1	0
208.	3	0	0	1	1	1	1	0
209.	3	0	1	0	0	0	0	0
210.	3	0	1	0	0	1	0	1
211.	3	0	1	0	0	0	1	0
212.	3	0	1	0	0	1	1	0
213.	3	0	1	0	1	0	0	1
214.	3	0	1	0	1	1	0	1
215.	3	0	1	0	1	0	1	3
216.	3	0	1	0	1	1	1	0
217.	3	0	1	1	0	0	0	0
218.	3	0	1	1	0	1	0	6
219.	3	0	1	1	0	0	1	1
220.	3	0	1	1	0	1	1	0
221.	3	0	1	1	1	0	0	0
222.	3	0	1	1	1	1	0	0
223.	3	0	1	1	1	0	1	0
224.	3	0	1	1	1	1	1	0
225.	3	1	0	0	0	0	0	0
226.	3	1	0	0	0	1	0	2
227.	3	1	0	0	0	0	1	2
228.	3	1	0	0	0	1	1	0
229.	3	1	0	0	1	0	0	0
230.	3	1	0	0	1	1	0	2
231.	3	1	0	0	1	0	1	4
232.	3	1	0	0	1	1	1	0
233.	3	1	0	1	0	0	0	0
234.	3	1	0	1	0	1	0	9
235.	3	1	0	1	0	0	1	11
236.	3	1	0	1	0	1	1	0
237.	3	1	0	1	1	0	0	0
238.	3	1	0	1	1	1	0	0
239.	3	1	0	1	1	0	1	0
240.	3	1	0	1	1	1	1	0
241.	3	1	1	0	0	0	0	0
242.	3	1	1	0	0	1	0	0
243.	3	1	1	0	0	0	1	0
244.	3	1	1	0	0	1	1	0
245.	3	1	1	0	1	0	0	3
246.	3	1	1	0	1	1	0	3
247.	3	1	1	0	1	0	1	0
248.	3	1	1	0	1	1	1	0
249.	3	1	1	1	0	0	0	3
250.	3	1	1	1	0	1	0	17
251.	3	1	1	1	0	0	1	5
252.	3	1	1	1	0	1	1	0
253.	3	1	1	1	1	0	0	0
254.	3	1	1	1	1	1	0	0
255.	3	1	1	1	1	0	1	0
256.	3	1	1	1	1	1	1	0

OK, il dataset è nella forma adatta ai nostri scopi. Adesso voglio calcolare la probabilità associata a ciascuna delle 256 combinazioni ed aggiungerla come nuova variabile. Pos-

siamo fare questo con il comando `prchange`, passandogli come parametri i valori in base ai quali calcolare le probabilità. In pratica si dovrebbe applicare il comando per le 256 combinazioni, partendo dalla prima combinazione:

```
prchange, x(sons_M_l         = 0
            higher_edu_M_l = 0
            higher_edu_F_l = 0
            du_dip_M       = 0
            du_self_M      = 0
            du_nw_F        = 0
            du_dip_F       = 0);
```

per finire con l'ultima combinazione

```
prchange, x(sons_M_l         = 3
            higher_edu_M_l = 1
            higher_edu_F_l = 1
            du_dip_M       = 1
            du_self_M      = 1
            du_nw_F        = 1
            du_dip_F       = 1);
```

nella `return list` del comando troviamo il vettore `r(predval)` che come secondo elemento contiene la probabilità stimata che ci interessa. A questo punto diventa più semplice usare le matrici. Per prima cosa trasformiamo il dataset in una matrice che chiameremo `comb`:

```
mkmat sons_M_l higher_edu_M_l higher_edu_F_l du_dip_M du_self_M du_nw_F du_dip_F,
      matrix(comb);
```

Determiniamo il numero di righe della matrice `comb` e costruiamo un ciclo `foreach` per passare i parametri al comando. Questi parametri vengono passati a `prchange` attraverso delle `local` (`local sons_M_l` ...`local du_dip_F`) i cui valori vengono prelevati dalla matrice `comb`:

```
local r_comb = rowsof(comb);

foreach riga of numlist 1/`r_comb' ;
**set trace on;
local sons_M_l       = comb[`riga',1];
local higher_edu_M_l = comb[`riga',2];
local higher_edu_F_l = comb[`riga',3];
local du_dip_M       = comb[`riga',4];
local du_self_M      = comb[`riga',5];
local du_nw_F        = comb[`riga',6];
local du_dip_F       = comb[`riga',7];


prchange, x(sons_M_l   =`sons_M_l'
            higher_edu_M_l =`higher_edu_M_l'
            higher_edu_F_l =`higher_edu_F_l'
            du_dip_M       =`du_dip_M'
            du_self_M      =`du_self_M'
            du_nw_F        =`du_nw_F'
            du_dip_F       =`du_dip_F');
matrix pred = r(predval);
matrix comb_pr`riga' = comb[`riga',1...] , pred[1,2];
matrix drop pred;
qui mat comb_pr= (nullmat(comb_pr)  comb_pr`riga');

;
```

al termine di ogni esecuzione di `prchange` il valore di `pred[1,2]`, ovvero il secondo valore del vettore `r(predval)` viene aggiunto come nuovo elemento al vettore riga dei valori delle variabili `matrix comb_pr`riga' = comb[`riga',1...] , pred[1,2];`. Infine tutte le righe vengono combinate assieme per ottenere la matrice finale `comb_pr`;

```
. matrix list comb_pr;

comb_pr[256,8]
      sons_M_l  high~_M_l  high~_F_l   du_dip_M  du_self_M   du_nw_F  du_dip_F         c8
  r1         0          0          0          0          0         1         0   .03125083
  r2         1          0          0          0          0         1         0   .02227251
  r3         2          0          0          0          0         1         0   .01557578
  r4         3          0          0          0          0         1         0    .0106863
  r5         0          0          0          0          0         0         1    .0469672
  r6         1          0          0          0          0         0         1   .03428787
  r7         2          0          0          0          0         0         1   .02456796
  r8         3          0          0          0          0         0         1   .01727403
  r9         0          0          0          0          0         0         0   .04518275
 r10         1          0          0          0          0         0         0   .03290772
 r11         2          0          0          0          0         0         0   .02352309
 r12         3          0          0          0          0         0         0   .01649973
 r13         0          0          1          0          0         1         0   .02135735
 r14         1          0          1          0          0         1         0   .01490165
 r15         2          0          1          0          0         1         0   .01020022
 r16         3          0          1          0          0         1         0   .00684864
 r17         0          0          1          0          0         0         1   .03297465
 r18         1          0          1          0          0         0         1   .02357369
 r19         2          0          1          0          0         0         1   .01653719
 r20         3          0          1          0          0         0         1   .01138162
 r21         0          0          1          0          0         0         0   .03163838
 r22         1          0          1          0          0         0         0   .02256464
 r23         2          0          1          0          0         0         0   .01579133
 r24         3          0          1          0          0         0         0   .01084198
 r25         0          0          0          1          0         1         0   .03850239
 r26         1          0          0          1          0         1         0    .0277755
 r27         2          0          0          1          0         1         0   .01966359
 r28         3          0          0          1          0         1         0   .01365868
 r29         0          0          0          1          0         0         1   .05698306
 r30         1          0          0          1          0         0         1   .04209995
 r31         2          0          0          1          0         0         1   .03053238
 r32         3          0          0          1          0         0         1   .02173155
 r33         0          0          0          1          0         0         0   .05490045
 r34         1          0          0          1          0         0         0   .04046682
 r35         2          0          0          1          0         0         0   .02927882
 r36         3          0          0          1          0         0         0   .02078969
 r37         0          0          1          1          0         1         0   .02667329
 r38         1          0          1          1          0         1         0   .01884039
 r39         2          0          1          1          0         1         0   .01305687
 r40         3          0          1          1          0         1         0   .00887669
 r41         0          0          1          1          0         0         1   .04054608
 r42         1          0          1          1          0         0         1   .02933957
 r43         2          0          1          1          0         0         1   .02083529
 r44         3          0          1          1          0         0         1   .01451788
 r45         0          0          1          1          0         0         0   .03896226
 r46         1          0          1          1          0         0         0   .02812697
 r47         2          0          1          1          0         0         0   .01992651
 r48         3          0          1          1          0         0         0   .01385122
 r49         0          0          0          0          1         1         0   .04885904
 r50         1          0          0          0          1         1         0   .03575507
 r51         2          0          0          0          1         1         0   .02568178
 r52         3          0          0          0          1         1         0    .0181017
 r53         0          0          0          0          1         0         1    .0710117
 r54         1          0          0          0          1         0         1   .05320916
```

r55	2	0	0	0	1	0	1	.03914384
r56	3	0	0	0	1	0	1	.02826582
r57	0	0	0	0	1	0	0	.06853762
r58	1	0	0	0	1	0	0	.05123693
r59	2	0	0	0	1	0	0	.03760492
r60	3	0	0	0	1	0	0	.0270904
r61	0	0	1	0	1	1	0	.03439569
r62	1	0	1	0	1	1	0	.02464971
r63	2	0	1	0	1	1	0	.0173347
r64	3	0	1	0	1	1	0	.01196023
r65	0	0	1	0	1	0	1	.05133273
r66	1	0	1	0	1	0	1	.03767958
r67	2	0	1	0	1	0	1	.02714735
r68	3	0	1	0	1	0	1	.01919419
r69	0	0	1	0	1	0	0	.04941624
r70	1	0	1	0	1	0	0	.03618798
r71	2	0	1	0	1	0	0	.02601101
r72	3	0	1	0	1	0	0	.01834679
r73	0	1	0	0	0	1	0	.06240681
r74	1	1	0	0	0	1	0	.04637326
r75	2	1	0	0	0	1	0	.03382808
r76	3	1	0	0	0	1	0	.02421956
r77	0	1	0	0	0	0	1	.08896469
r78	1	1	0	0	0	0	1	.06767102
r79	2	1	0	0	0	0	1	.05054738
r80	3	1	0	0	0	0	1	.03706785
r81	0	1	0	0	0	0	0	.08602732
r82	1	1	0	0	0	0	0	.06528763
r83	2	1	0	0	0	0	0	.0486544
r84	3	1	0	0	0	0	0	.03559617
r85	0	1	1	0	0	1	0	.04469279
r86	1	1	1	0	0	1	0	.03252943
r87	2	1	1	0	0	1	0	.0232372
r88	3	1	1	0	0	1	0	.01628823
r89	0	1	1	0	0	0	1	.06540354
r90	1	1	1	0	0	0	1	.04874634
r91	2	1	1	0	0	0	1	.03566756
r92	3	1	1	0	0	0	1	.02561526
r93	0	1	1	0	0	0	0	.0630826
r94	1	1	1	0	0	0	0	.04690768
r95	2	1	1	0	0	0	0	.03424177
r96	3	1	1	0	0	0	0	.02453301
r97	0	1	0	1	0	1	0	.07486377
r98	1	1	0	1	0	1	0	.05629028
r99	2	1	0	1	0	1	0	.0415562
r100	3	1	0	1	0	1	0	.03011463
r101	0	1	0	1	0	0	1	.10514402
r102	1	1	0	1	0	0	1	.08090958
r103	2	1	0	1	0	0	1	.06115075
r104	3	1	0	1	0	0	1	.0453811
r105	0	1	0	1	0	0	0	.10182033
r106	1	1	0	1	0	0	0	.07817525
r107	2	1	0	1	0	0	0	.05894887
r108	3	1	0	1	0	0	0	.04364548
r109	0	1	1	1	0	1	0	.05432799
r110	1	1	1	1	0	1	0	.04001869
r111	2	1	1	1	0	1	0	.02893543
r112	3	1	1	1	0	1	0	.02053213
r113	0	1	1	1	0	0	1	.07830834
r114	1	1	1	1	0	0	1	.0590559
r115	2	1	1	1	0	0	1	.04372974
r116	3	1	1	1	0	0	1	.03178672
r117	0	1	1	1	0	0	0	.07564123
r118	1	1	1	1	0	0	0	.05691365
r119	2	1	1	1	0	0	0	.04204545

r120	3	1	1	1	0	0	0	.0304905
r121	0	1	0	0	1	1	0	.09205978
r122	1	1	0	0	1	1	0	.07018927
r123	2	1	0	0	1	1	0	.05255296
r124	3	1	0	0	1	1	0	.03863135
r125	0	1	0	0	1	0	1	.12704809
r126	1	1	0	0	1	0	1	.09910604
r127	2	1	0	0	1	0	1	.07594781
r128	3	1	0	0	1	0	1	.05715961
r129	0	1	0	0	1	0	0	.12324235
r130	1	1	0	0	1	0	0	.09592332
r131	2	1	0	0	1	0	0	.07334241
r132	3	1	0	0	1	0	0	.05507191
r133	0	1	1	0	1	1	0	.06785662
r134	1	1	1	0	1	1	0	.05069501
r135	2	1	1	0	1	1	0	.03718279
r136	3	1	1	0	1	1	0	.02676854
r137	0	1	1	0	1	0	1	.09607838
r138	1	1	1	0	1	0	1	.07346918
r139	2	1	1	0	1	0	1	.05517336
r140	3	1	1	0	1	0	1	.04068058
r141	0	1	1	0	1	0	0	.09296776
r142	1	1	1	0	1	0	0	.07092933
r143	2	1	1	0	1	0	0	.05314341
r144	3	1	1	0	1	0	0	.03909247
r145	0	1	1	1	1	1	1	.08393377
r146	0	0	0	1	1	0	1	.08473428
r147	0	0	1	1	0	1	1	.02783366
r148	0	0	0	0	0	1	1	.03257362
r149	0	1	0	0	0	1	1	.06470847
r150	0	1	1	0	1	1	1	.07031184
r151	0	0	1	1	1	1	0	.04222742
r152	0	1	1	1	1	0	0	.10966442
r153	0	1	0	1	1	1	0	.10864002
r154	0	1	1	1	0	1	1	.05639349
r155	0	0	1	1	1	1	1	.04391757
r156	0	1	0	1	0	1	1	.07751009
r157	0	1	1	1	1	1	0	.08112232
r158	0	1	0	1	1	1	1	.11212215
r159	0	0	0	1	0	1	1	.04007105
r160	0	0	0	1	1	1	1	.0613959
r161	0	0	1	0	1	1	1	.03582672
r162	0	0	0	0	1	1	1	.05075834
r163	0	1	0	1	1	0	1	.14776383
r164	0	1	0	0	1	1	1	.09514806
r165	0	1	1	1	1	0	1	.11316993
r166	0	0	0	1	1	1	0	.05918708
r167	0	0	0	1	1	0	0	.08190262
r168	0	1	0	1	1	0	0	.14354074
r169	0	0	1	1	1	0	1	.06206314
r170	0	0	1	1	1	0	0	.0598355
r171	0	0	1	0	0	1	1	.02232083
r172	0	1	1	0	0	1	1	.04646172
r173	1	0	0	1	1	1	0	.04383302
r174	1	0	1	1	1	1	0	.03063037
r175	1	0	0	0	1	1	1	.0372321
r176	1	0	0	1	1	0	1	.06424052
r177	1	0	0	1	1	1	1	.04557462
r178	1	0	1	1	1	1	1	.03193149
r179	1	1	0	1	1	1	1	.08667391
r180	1	1	1	1	1	1	0	.06132231
r181	1	1	1	0	0	1	1	.03389654
r182	1	1	0	1	0	1	1	.05841413
r183	1	0	1	0	1	1	1	.02573625
r184	1	1	0	0	1	1	1	.07270885

r185	1	0	1	1	1	0	1	.04610165
r186	1	0	0	0	0	1	1	.02327058
r187	1	0	1	1	1	0	0	.04434379
r188	1	1	0	1	1	0	1	.11657615
r189	1	0	1	1	0	1	1	.01970707
r190	1	0	1	0	0	1	1	.01561142
r191	1	0	0	1	0	1	1	.02897554
r192	1	1	0	1	1	0	0	.11299532
r193	1	1	1	1	0	1	1	.04163717
r194	1	1	1	1	1	0	1	.08754214
r195	1	1	0	0	0	1	1	.04819519
r196	1	1	0	1	1	1	0	.08379353
r197	1	0	0	1	1	0	0	.06195188
r198	1	1	1	1	1	1	1	.0635929
r199	1	1	1	0	1	1	1	.05265072
r200	1	1	1	1	1	0	0	.08464009
r201	2	1	0	0	1	1	1	.0545651
r202	2	0	0	1	1	0	0	.04601374
r203	2	1	1	1	1	0	1	.06651596
r204	2	1	1	0	1	1	1	.03870767
r205	2	0	1	0	0	1	1	.01071203
r206	2	0	0	1	1	1	1	.03321049
r207	2	1	1	0	0	1	1	.02427141
r208	2	1	1	1	1	0	0	.0641643
r209	2	0	1	1	0	1	1	.01369051
r210	2	0	1	0	1	1	1	.01814223
r211	2	1	1	1	1	1	0	.04551652
r212	2	1	1	1	1	1	1	.04731152
r213	2	0	1	1	1	1	1	.02278574
r214	2	0	0	0	0	1	1	.01631292
r215	2	0	1	1	1	1	0	.02180528
r216	2	1	0	1	0	1	1	.04322468
r217	2	0	1	1	1	0	0	.03226015
r218	2	1	0	1	1	1	1	.06581173
r219	2	1	0	1	1	1	0	.06347949
r220	2	1	0	0	0	1	1	.03523976
r221	2	0	0	0	1	1	1	.02680613
r222	2	1	0	1	1	0	0	.0873974
r223	2	1	0	1	1	0	1	.09036943
r224	2	1	1	1	0	1	1	.03017682
r225	2	0	0	1	1	0	1	.04782438
r226	2	0	0	1	1	1	0	.03186632
r227	2	0	0	1	0	1	1	.0205622
r228	2	0	1	1	1	0	1	.03361796
r229	3	0	1	1	1	1	0	.0152314
r230	3	1	1	1	1	0	0	.04776399
r231	3	0	0	1	1	0	0	.03354997
r232	3	1	1	0	1	1	1	.02793236
r233	3	0	1	1	1	0	0	.02303382
r234	3	1	0	0	0	1	1	.02529023
r235	3	1	0	1	1	0	1	.06881309
r236	3	1	1	1	1	1	0	.03316559
r237	3	0	1	0	1	1	1	.01254771
r238	3	1	1	1	0	1	1	.02146415
r239	3	0	0	0	0	1	1	.01121921
r240	3	1	1	0	0	1	1	.01705406
r241	3	1	0	1	1	1	0	.04722175
r242	3	0	0	1	1	1	0	.02273657
r243	3	1	0	0	1	1	1	.04020426
r244	3	0	0	1	1	1	1	.02375207
r245	3	0	1	1	1	0	1	.02406044
r246	3	1	0	1	1	0	0	.06639852
r247	3	1	1	1	1	0	1	.04962935
r248	3	0	0	1	0	1	1	.01431736
r249	3	0	1	1	1	1	1	.01595458

r250	3	1	0	1	0	1	1	.03139766
r251	3	0	0	0	1	1	1	.01893949
r252	3	1	0	1	1	1	1	.04907022
r253	3	0	0	1	1	0	1	.03495214
r254	3	0	1	0	0	1	1	.00720989
r255	3	0	1	1	0	1	1	.00933016
r256	3	1	1	1	1	1	1	.0345546

Capitolo 20

`reshape` su molte Variabili

Per l'esecuzione dei comandi **`reshape long`** e **`reshape wide`** è necessario specificare tutte le variabili che abbiamo definito del gruppo $\texttt{X}_{ij}$ (vedi capitolo 10.4 pagina 127). Purtroppo per specificare questa lista di variabili non possiamo fare ricorso ai caratteri jolly e alle sequenze di variabili (vedi capitolo 5.7 pagina 34). Come fare nel caso in cui le variabili del gruppo $\texttt{X}_{ij}$ siano molto numerose? Cerchiamo di spiegarlo tramite un caso applicato.
Abbiamo il seguente dataset:

```
. desc, short

Contains data from dta/ts.dta
  obs:           800
 vars:         3,511                          24 May 2008 10:27
 size:    22,450,400 (85.7% of memory free)
Sorted by:
     Note:  dataset has changed since last saved
```

di cui dobbiamo fare un **`reshape long`** sul gruppo di 3500 variabili $\texttt{X}_{ij}$. Le variabili riguardano indici relativi ad una serie di titoli bond e si presentano in gruppi di 8, dove per ciascun titolo si riportano 8 indici:

```
. desc DE000WLB8FC4_DM-DE000WLB8FC4_MV, fullnames

                  storage  display     value
variable name       type   format      label      variable label
------------------------------------------------------------------------------
DE000WLB8FC4_DM    double  %10.0g
DE000WLB8FC4_IBOX  double  %10.0g
DE000WLB8FC4_SP    double  %10.0g
DE000WLB8FC4_SWSP  double  %10.0g
DE000WLB8FC4_RI    double  %10.0g
DE000WLB8FC4_AC    double  %10.0g
DE000WLB8FC4_RYIB  double  %10.0g
DE000WLB8FC4_MV    double  %10.0g
```

La prima parte del nome della variabile è l'identificativo del titolo, la seconda parte, quella dopo l'"_", è l'indice relativo al titolo. In particolare

DM Duration - Modified
IBOX iBoxx End-Of-Day Mid Price

SP Spread Over Benchmark Curve
SWSP Spread Over Swap Curve
RI Total Return Index
AC Accrued Interest
RYIB IBoxx - Yield
MV Market Value (Capital)

e rappresentano l'elemento `j`. Teoricamente si dovrebbe fare il `reshape` scrivendo circa 440 variabili!. Con questo pezzo di codice però riesco ad ottenere la lista che mi serve:

```
preserve;
keep *_AC;

renvars , postsub(_AC _);

ds *_;
di "`r(varlist)'";
restore;
```

Analizziamolo. Con `keep *_AC` prendo una sola occorrenza tra le 8 di ciascun titolo, quindi elimino la parte terminale `AC` del nome delle variabili. Con il comando `ds` ottengo la lista delle variabili che vengono salvare nella local `r(varlist)`

```
. di "`r(varlist)'";
DE000WLB8FC4_ XS0308936037_ DE000A0NUMG1_ XS0222695008_ XS0188
> 568751_ DE0009168062_ DE0007009482_ DE000A0DXH13_ DE000A0DAL

  (output omitted)

> 0185415139_ XS0173128983_ XS0328609580_ XS0312464158_
```

Ok ci siamo; la variabile `data` è l'elemento `i`, indichiamo con `tipo` l'elemento `j` e aggiungiamo l'opzione `string` dato che l'identificativo $\mathtt{X}_{ij}$ non è numerico:

```
. reshape long  "`r(varlist)'" , i(data) j(tipo) string;
(note: j = AC DM IBOX MV RI RYIB SP SWSP)
(note: ES0312342001_DM not found)
(note: ES0312298005_DM not found)
(note: DE000LBW6062_DM not found)
(note: ES0312342001_IBOX not found)
(note: ES0312298005_IBOX not found)
(note: ES0370148001_IBOX not found)
(note: DE000LBW6062_IBOX not found)
(note: DE000LBW3DR8_IBOX not found)
(note: DE000HBE0D55_IBOX not found)
(note: DE000WLB6EG2_IBOX not found)
(note: XS0282598167_IBOX not found)
(note: DE000NWB2846_IBOX not found)
(note: DE000DHY1149_IBOX not found)
(note: DE000MHB3018_IBOX not found)
(note: DE000LBW6062_MV not found)
(note: DE000LBW6062_RI not found)
(note: ES0312342001_RYIB not found)
(note: ES0312298005_RYIB not found)
(note: ES0370148001_RYIB not found)
(note: DE000LBW6062_RYIB not found)
(note: DE000LBW3DR8_RYIB not found)
(note: DE000HBE0D55_RYIB not found)
(note: DE000WLB6EG2_RYIB not found)
(note: XS0282598167_RYIB not found)
(note: DE000NWB2846_RYIB not found)
```

```
(note: DE000DHY1149_RYIB not found)
(note: DE000MHB3018_RYIB not found)
(note: DE000LBW6062_SP not found)
(note: DE000LBW6062_SWSP not found)

Data                               wide   ->   long
-----------------------------------------------------------------------------
Number of obs.                      800   ->    6400
Number of variables                3511   ->     447
j variable (8 values)                     ->   tipo
xij variables:
DE000WLB8FC4_AC DE000WLB8FC4_DM ... DE000WLB8FC4_SWSP->DE000WLB8FC4_
XS0308936037_AC XS0308936037_DM ... XS0308936037_SWSP->XS0308936037_

  (output omitted)

XS0241183804_AC XS0241183804_DM ... XS0241183804_SWSP->XS0241183804_
XS0169374443_AC XS0169374443_DM ... XS0169374443_SWSP->XS0169374443_
-----------------------------------------------------------------------------
```

Capitolo 21

Importare dati .xml dal GME

Il GME (Gestore Mercati Elettrici) mette a disposizione nella sezione download del suo sito (http://www.mercatoelettrico.org/), diversi database. Nel nostro caso ci interessano i database relativi ai prices, scaricabili in formato .xml con un file di dati per ciascun giorno dell'anno. Sempre sul sito indicano una procedura che passando da Access consente di importare i dati in Excel. Il nostro obiettivo è creare un database in Stata che comprenda tutti i singoli database giornalieri. I problemi da risolvere sono molteplici:

- Creare una procedura che automatizzi l'accesso a files contenuti in 365 (o 366) archivi in formato .zip
- Caricare in Stata un file di dati in formato .xml
- Riunire i 365 (o 366) file giornalieri in un unico file
- Ripetere i punti precedenti per n anni (2004-2011 nel nostro caso)

21.1 Accesso ai files .zip

Per ogni anno di interesse (dal 2004 al 2011) è stata creata una cartella. All'interno di questa cartella sono stati scaricati i files .zip giornalieri che hanno una struttura del tipo <aaaammgg>OfferteFree_Pubbliche.zip dove <> rappresenta la data in formato anno a quattro cifre, mese a due e giorno a 2. Questo archivio contiene a sua volta altri files .zip (da tre a cinque) anche questi con una nomenclatura definita. Per esempio il file 20100101OfferteFree_Pubbliche.zip contiene i seguenti files:

20100101MGPOffertePubbliche.zip
20100101MI1OffertePubbliche.zip
20100101MI2OffertePubbliche.zip
20100101MSDOffertePubbliche.zip

Tutti iniziano con l'indicazione della data e ciò che cambia è la parte finale del nome. Il file .xml da caricare è contenuto nel file <aaaammgg>MGPOffertePubbliche.zip ed ha lo stesso nome dello zipfile che lo contiene (<aaaammgg>MGPOffertePubbliche.xml). Quindi la prima procedura dovrebbe:

1. entrare nella cartella di ciascun anno
2. creare l'elenco dei file. zip da processare
3. estrarre da ciascun file .zip gli n file .zip contenuti
4. estrarre dal <aaaammgg>MGPOffertePubbliche.zip il file <aaaammgg>MGPOffertePubbliche.xml

Vediamo come è possibile fare ciò, premettendo che questo codice si inserisce in una procedura che automatizza il processo per tutti gli anni presi in considerazione. Qui per semplicità espositiva ci si limita al solo anno 2010 per cui si definisce la local `Y`:

```
local Y = 2010;
```

Tramite `local list :` si inseriscono nella local `list` tutti i files .zip contenuti nella cartella data/2010, si determina il loro numero (`local quanti :`) e per sicurezza vengono ordinati in senso alfabetico crescente (`local list : list sort list`)[1];

```
local list : dir "data/`Y´" files "*.zip";
local quanti : list sizeof list;
local list : list sort list;
```

il risultato di queste operazioni è il seguente:

```
di `list´;

20100101offertefree_pubbliche.zip20100102offertefree_pubbliche.zip...
> zip20100105offertefree_pubbliche.zip20100106offertefree_pubbliche.zip...
> iche.zip20100109offertefree_pubbliche.zip20100110offertefree_pubbliche.zip...
> pubbliche.zip20100113offertefree_pubbliche.zip20100114offertefree_pubbliche.zip...
> free_pubbliche.zip20100117offertefree_pubbliche.zip20100118offertefree_pubbliche.zip...
> fertefree_pubbliche.zip20100121offertefree_pubbliche.zip...
> 124offertefree_pubbliche.zip20100125offertefree_pubbliche.zip...
> 20100128offertefree_pubbliche.zip20100129offertefree_pubbliche.zip...
> e.zip20100201offertefree_pubbliche.zip20100202offertefree_pubbliche.zip...
> bliche.zip20100205offertefree_pubbliche.zip20100206offertefree_pubbliche.zip...
> e_pubbliche.zip20100209offertefree_pubbliche.zip20100210offertefree_pubbliche.zip...
...
```

Nella local `list` ci sono tutti i files .zip da aprire i quali a loro volta conterranno il file <aaaammgg>MGPOffertePubbliche.zip da aprire ulteriormente per accedere a <aaaammgg>MGPOffertePubbliche.xml. Tutto ciò viene eseguito all'interno di un ciclo `foreach`

[1] Questo ordinamento viene fatto per avere lo stesso comportamento in ambiente Linux e Windows. Non ne ho capito il motivo ma, lo stesso comando, genera un lista ordinata in senso alfabetico crescente in Windows ma non in Linux.

```
1. local id_file = 1;
2. foreach file of local list {;
3.     local id_time : piece 1 8 of "`file´";
4.     qui unzipfile data/`Y´/`file´ , replace;
5.     qui unzipfile `id_time´mgpoffertepubbliche.zip , replace;
6.     shell erase *ubbliche.zip;
7.     global xml_file "`id_time´MGPOffertePubbliche";
8.     local xml_file = "$xml_file";
9.     qui include read_xml2.do;
10.    if `id_file´==1 qui save temp_xml, replace;
11.    else  {;
12.       qui append using temp_xml;
13.       qui save temp_xml, replace;
14.    };
15. capture erase `xml_file´.xml;
16. local id_file `++id_file´;
17. };
```

Vediamo in dettaglio riga per riga

```
1. local id_file = 1;
```

Si inizializza la local `id_file` al valore 1. Questo servirà in seguito (righe 10.-14.) per sapere se i dati caricati sono il primo giorno dell'anno e quindi da salvare subito o se devono essere aggiunti a un database già esistente tramite un `append`.

```
2. foreach file of local list {;
```

Si inizia il ciclo per ciascun file dell'anno contenuto in `list`.

```
3.     local id_time : piece 1 8 of "`file´";
```

Dal nome del file si estraggono le prime otto lettere corrispondenti alla data e vengono assegnate alla local `id_time`

```
4.     qui unzipfile data/`Y´/`file´ , replace;
```

Si estrae il contenuto del file .zip indicato nella local `'file'`

```
5.     qui unzipfile `id_time´mgpoffertepubbliche.zip , replace;
```

Si estrae il contenuto del file 'id_time'mgpoffertepubbliche.zip, dove `'id_time'` indica la data di riferimento ricavata alla riga 3.

```
6.     shell erase *ubbliche.zip;
7.     global xml_file "`id_time´MGPOffertePubbliche";
8.     local xml_file = "$xml_file";
```

Vengono cancellati i files .zip estratti da `'file'` e il cui nome termina in *ubbliche.zip (6.) e si assegna alla global e alla local `xml_file` il nome del file .xml da caricare in Stata (righe 7. e 8.).

```
9.     qui include read_xml2.do;
```

Viene mandato il esecuzione il do-file read_xml2 che si occupa di caricare in Stata il contenuto del file .xml. Vedremo tra poco il suo funzionamento.

```
10.    if `id_file´==1 qui save temp_xml, replace;
11.    else  {;
12.       qui append using temp_xml;
```

```
13.       qui save temp_xml, replace;
14.    };
```

Al termine dell'esecuzione di read_xml2.do viene verificato se questo è il primo file di dati caricato. In caso affermativo viene salvato il file temp_xml.dta (riga 10.), altrimenti viene fatto un `append` con il file temp_xml.dta (che quindi esiste) e poi risalvato con lo stesso nome (righe 11. - 13.).

```
15. capture erase `xml_file´.xml;
16. local id_file `++id_file´;
```

Prima di iniziare il ciclo successivo viene cancellato il file .xml appena letto (15.) e incrementato il contatore `id_file` utilizzato per discriminare quale azione debba essere eseguita nelle righe 10.-14.

21.2 Leggere dati in formato .xml

Stata ha la possibilità di caricare dati in formato .xml tramite il comando `xmluse`. Sono previsti due tipi di doctype xml leggibili: dta ed excel. In questo caso il comando non funziona non essendo gli .xml in nessuno dei due formati. Usando `xmluse` si ottiene questo messaggio di errore:

```
. xmluse 20100222MGPOffertePubbliche.xml, clear
unrecognizable XML doctype
r(198);
```

I file .xml sono in sostanza dei file di dati in formato testo con in più tutta un serie di informazioni racchiusa in tag. Se si riesce a ripulire il file da tutti i tag si ottiene un semplice file di testo e con un po' di lavoro si può ricostruire il database.
Analizziamo i file .xml del nostro caso. Iniziano con un preambolo in cui viene descritto il file, le variabili che contiene e il loro tipo. Ne riporto una parte:

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
<xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
         xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:UseCurrentLocale="true">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="OfferteOperatori" msdata:CaseSensitive="False" msdata:Locale="it-IT">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="PURPOSE_CD" type="xs:string" minOccurs="0" />
              <xs:element name="TYPE_CD" type="xs:string" minOccurs="0" />
              <xs:element name="STATUS_CD" type="xs:string" minOccurs="0" />
              <xs:element name="MARKET_CD" type="xs:string" minOccurs="0" />
              <xs:element name="UNIT_REFERENCE_NO" type="xs:string" minOccurs="0" />
              <xs:element name="MARKET_PARTECIPANT_XREF_NO" type="xs:string" minOccurs="0" />
              <xs:element name="INTERVAL_NO" type="xs:decimal" minOccurs="0" />
...
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Tutta questa parte non serve e verrà eliminata. Inizia poi l'informazione che si vuole caricare come nell'esempio che segue:

```
<OfferteOperatori>
    <PURPOSE_CD>BID</PURPOSE_CD>
    <TYPE_CD>REG</TYPE_CD>
    <STATUS_CD>ACC</STATUS_CD>
    <MARKET_CD>MGP</MARKET_CD>
    <UNIT_REFERENCE_NO>UC_DP0012_CNOR</UNIT_REFERENCE_NO>
    <INTERVAL_NO>1</INTERVAL_NO>
    <BID_OFFER_DATE_DT>20100101</BID_OFFER_DATE_DT>
    <TRANSACTION_REFERENCE_NO>93650573761067</TRANSACTION_REFERENCE_NO>
    <QUANTITY_NO>0.157</QUANTITY_NO>
    <AWARDED_QUANTITY_NO>0.157</AWARDED_QUANTITY_NO>
    <ENERGY_PRICE_NO>0.00</ENERGY_PRICE_NO>
    <MERIT_ORDER_NO>294</MERIT_ORDER_NO>
    <PARTIAL_QTY_ACCEPTED_IN>N</PARTIAL_QTY_ACCEPTED_IN>
    <ADJ_QUANTITY_NO>0.157</ADJ_QUANTITY_NO>
    <GRID_SUPPLY_POINT_NO>PSR_CNOR</GRID_SUPPLY_POINT_NO>
    <ZONE_CD>CNOR</ZONE_CD>
    <AWARDED_PRICE_NO>53.57</AWARDED_PRICE_NO>
    <OPERATORE>Bilateralista</OPERATORE>
    <SUBMITTED_DT>20091231091204624</SUBMITTED_DT>
    <BILATERAL_IN>true</BILATERAL_IN>
  </OfferteOperatori>
```

Tutto ciò che è racchiuso tra i tag < e > non serve come dato e va cancellato. Ci serve però come informazione per identificare le variabili. Infatti quella precedente è una singola osservazione ripartita però su più righe, per cui uno dei problemi da risolvere sarà anche come riportare tutte queste variabili riferite ad una sola osservazione su una sola riga. Inoltre il numero di variabili per singola osservazione non è costante, ovvero può accadere che per esempio in una certa osservazione manchi il dato relativo a <ZONE_CD> per cui questa riga manca all'interno dei tag <OfferteOperatori> e </OfferteOperatori>. Questo fatto, considerando anche che il numero di righe del preambolo non è costante nei diversi file .xml, comporta l'impossibilità di usare il comando `infile` per la lettura dei dati nella forma

```
infile dictionary using `xml_file´.xml {

_first(40) * Begin reading on line 2
_lines(21)  * Each observation takes 21 lines.

_line(1)   str3 purpose_cd
_line(2)   str3 type_cd
....

_line(21)  str4 bilateral_in
}
```

come abbiamo visto nel capitolo precedente è il file read_xml2.do che si occupa di leggere i file .xml. Con il comando

```
infix str var1 1-150 using `xml_file´.xml, clear;
```

si carica l'intero contenuto del file .xml nella variabile stringa `var1`. Ora questa andrà pulita dalle informazioni non pertinenti. Per prima cosa si eliminano tutte le righe relative al preambolo:

```
drop if  strmatch(var1,"<?xml*") | strmatch(var1,"<*NewDataSet>*") | strmatch(var1,"<*xs:*");
```

Si considera il tag <OfferteOperatori> come inizio osservazione e gli viene assegnato un codice 0 per poi poter ricostruire le osservazioni nel database finale mettendo tutta

l'informazione contenuta nei singoli <OfferteOperatori> - </OfferteOperatori> sulla medesima riga

```
drop if inlist(var1,"</OfferteOperatori>");
replace var_id=0 if strmatch(var1,"<OfferteOperatori>*")
```

Ora viene creata una variabile che numera tutte le righe che sono rimaste dopo l'opera di pulizia (`id`) e una variabile stringa che verrà popolata con l'informazione relativa al nome della variabile desunto dai singoli tag:

```
gen id=_n;
gen var_id_str = "";
     replace var_id_str="purpose_cd" if strmatch(var1,"<PURPOSE_CD>*");
     replace var_id_str="type_cd" if strmatch(var1,"<TYPE_CD>*");
   ...
```

Si assegna un numero identificativo crescente a tutte le occorrenze pari a 0 della variabile `var_id`. Questo sarà l'identificativo delle osservazioni del database finale. Tutte le righe che appartenevano ad uno stesso gruppo di righe contenute nei tag <OfferteOperatori> prendono lo stesso valore (ciclo `forvalues`)

```
egen id_group = group(id var_id) if var_id==0;
qui count;
forvalues i=1(1)`r(N)´ {;
if id_group[`i´]==. qui replace id_group = id_group[`i´-1] in `i´;
};
```

il risultato dei comandi precedenti sarà il seguente:

var1	id	var_id_str	id_group
<OfferteOperatori>	1	inizio_obs	1
<PURPOSE_CD>BID</PURPOSE_CD>	2	purpose_cd	1
<STATUS_CD>ACC</STATUS_CD>	3	status_cd	1
<UNIT_REFERENCE_NO>UC_DP0012_CNOR</UNIT_REFERENCE_NO>	4	unit_reference_no	1
<INTERVAL_NO>1</INTERVAL_NO>	5	interval_no	1
<BID_OFFER_DATE_DT>20100101</BID_OFFER_DATE_DT>	6	bid_offer_date_dt	1
<QUANTITY_NO>0.157</QUANTITY_NO>	7	quantity_no	1
<AWARDED_QUANTITY_NO>0.157</AWARDED_QUANTITY_NO>	8	awarded_quantity_no	1
<ENERGY_PRICE_NO>0.00</ENERGY_PRICE_NO>	9	energy_price_no	1
<MERIT_ORDER_NO>294</MERIT_ORDER_NO>	10	merit_order_no	1
<PARTIAL_QTY_ACCEPTED_IN>N</PARTIAL_QTY_ACCEPTED_IN>	11	partial_qty_accepted_in	1
<ZONE_CD>CNOR</ZONE_CD>	12	zone_cd	1
<AWARDED_PRICE_NO>53.57</AWARDED_PRICE_NO>	13	awarded_price_no	1
<OPERATORE>Bilateralista</OPERATORE>	14	operatore	1
<OfferteOperatori>	15	inizio_obs	2
<PURPOSE_CD>BID</PURPOSE_CD>	16	purpose_cd	2
<STATUS_CD>ACC</STATUS_CD>	17	status_cd	2
<UNIT_REFERENCE_NO>UC_DP0012_CNOR</UNIT_REFERENCE_NO>	18	unit_reference_no	2
<INTERVAL_NO>2</INTERVAL_NO>	19	interval_no	2
<BID_OFFER_DATE_DT>20100101</BID_OFFER_DATE_DT>	20	bid_offer_date_dt	2
<QUANTITY_NO>0.144</QUANTITY_NO>	21	quantity_no	2
<AWARDED_QUANTITY_NO>0.144</AWARDED_QUANTITY_NO>	22	awarded_quantity_no	2
<ENERGY_PRICE_NO>0.00</ENERGY_PRICE_NO>	23	energy_price_no	2
<MERIT_ORDER_NO>294</MERIT_ORDER_NO>	24	merit_order_no	2
<PARTIAL_QTY_ACCEPTED_IN>N</PARTIAL_QTY_ACCEPTED_IN>	25	partial_qty_accepted_in	2
<ZONE_CD>CNOR</ZONE_CD>	26	zone_cd	2
<AWARDED_PRICE_NO>48.00</AWARDED_PRICE_NO>	27	awarded_price_no	2
<OPERATORE>Bilateralista</OPERATORE>	28	operatore	2
<OfferteOperatori>	29	inizio_obs	3

```
                          <PURPOSE_CD>BID</PURPOSE_CD>   30              purpose_cd         3
                            <STATUS_CD>ACC</STATUS_CD>   31                status_cd         3
<UNIT_REFERENCE_NO>UC_DP0012_CNOR</UNIT_REFERENCE_NO>   32         unit_reference_no         3
                          <INTERVAL_NO>3</INTERVAL_NO>   33               interval_no         3
```

Di nuovo si impone un po' di pulitura. Si elimina la riga che identifica l'inizio dell'osservazione e sopratutto si crea la variabile `var2` pulita da tutto quello racchiuso tra i simboli $<$ e $>$.

```
drop if var_id==0;
gen var2 = regexr(var1,"<[^<>]+>","");
replace var2 = regexr(var2,"<[^<>]+>","");
```

Finalmente con un `reshape wide` si riportano le righe che appartengono alla stessa osservazione su una sola riga e si pulisce il nome delle variabili che risulta dopo il `reshape` (`renvars`):

```
drop var1 id var_id;
reshape wide var2, i(id_group) j(var_id_str) string;
renvars, presub(var2);
rename id_group id;
```

E questo sarà il risultato finale:

```
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
          id |     45375       22688    13098.78           1      45375
awarded_pr~o |     45375    20.35459    21.63082           0        115
awarded_qu~o |     45375    27.71044    137.3443           0   5817.424
bid_offer_~t |     45375       18263           0       18263      18263
energy_pri~o |     45375    37.18588    177.5324           0       3000
-------------+--------------------------------------------------------
 interval_no |     45375    12.66969    6.874062           1         24
merit_orde~o |     45375    395.2345    350.7231           0       1224
   operatore |     45375    78.35423    49.34221           2        210
partial_qt~n |     45375    .0039229    .0625105           0          1
  purpose_cd |     45375    1.705543    .4558034           1          2
-------------+--------------------------------------------------------
 quantity_no |     45375    79.44589    580.1227        .001      10000
   status_cd |     45375    1.705499    1.125442           1          5
unit_refer~o |         0
     zone_cd |     37727     9.52037    4.232601           1         17
```

Nota finale: caricando per intero un singolo file .xml e trasformando in numeriche tutte le variabili possibili, si ottiene un dataset in Stata della dimensione di circa 4-5mb. Moltiplicando questo valore per 365 si ha un database annuale di circa 1.6 GB. Quindi l'uso di Stata a 64bit è si rende di fatto necessario.

Parte III

Appendici

Appendice A

spmap: Visualization of spatial data

Autore: Maurizio Pisati
Department of Sociology and Social Research
University of Milano Bicocca - Italy
maurizio.pisati@unimib.it

A.1 Syntax

(Version 1.1.0)

`spmap` [`attribute`] [*if*] [*in*] `using` *basemap* [`,`
basemap_options
`polygon(`*polygon_suboptions*`)`
`line(`*line_suboptions*`)`
`point(`*point_suboptions*`)`
`diagram(`*diagram_suboptions*`)`
`arrow(`*arrow_suboptions*`)`
`label(`*label_suboptions*`)`
`scalebar(`*scalebar_suboptions*`)`
graph_options]

A.1.1 basemap_options

Main

`id(`*idvar*`)` base map polygon identifier[1]

Cartogram

[1] Required option

area(*areavar*) draw base map polygons with area proportional to variable *areavar*
split split multipart base map polygons
map(*backgroundmap*) draw background map defined in Stata dataset *backgroundmap*
mfcolor(*colorstyle*) fill color of the background map
mocolor(*colorstyle*) outline color of the background map
mosize(*linewidthstyle*) outline thickness of the background map
mopattern(*linepatternstyle*) outline pattern of the background map

Choropleth map

clmethod(*method*) *attribute* classification method, where *method* is one of the following: quantile, boxplot, eqint, stdev, kmeans, custom, unique
clnumber(#) number of classes
clbreaks(*numlist*) custom class breaks
eirange(*min, max*) *attribute* range for eqint classification method
kmiter(#) number of iterations for kmeans classification method
ndfcolor(*colorstyle*) fill color of empty (no data) base map polygons
ndocolor(*colorstyle*) outline color of empty (no data) base map polygons
ndsize(*linewidthstyle*) outline thickness of empty (no data) base map polygons
ndlabel(*string*) legend label of empty (no data) base map polygons

Format

fcolor(*colorlist*) fill color of base map polygons
ocolor(*colorlist*) outline color of base map polygons
osize(*linewidthstyle_list*) outline thickness of base map polygons

Legend

legenda(on|off) display/hide base map legend
legtitle(*string*) base map legend title
leglabel(*string*) single-key base map legend label
legorder(hilo|lohi) base map legend order
legstyle(0|1|2|3) base map legend style
legjunction(*string*) string connecting lower and upper class limits in base map legend labels when legstyle(2)
legcount display number of base map polygons belonging to each class

A.1.2 polygon_suboptions

Main

data(*polygon*) Stata dataset defining one or more supplementary polygons to be superimposed onto the base map[2]
select(*command*) keep/drop specified records of dataset *polygon*

[2] Required when option polygon() is specified

by(*byvar_pl*) group supplementary polygons by variable *byvar_pl*

Format

fcolor(*colorlist*) fill color of supplementary polygons
ocolor(*colorlist*) outline color of supplementary polygons
osize(*linewidthstyle_list*) outline thickness of supplementary polygons

Legend

legenda(on|off) display/hide supplementary-polygon legend
legtitle(*string*) supplementary-polygon legend title
leglabel(*string*) single-key supplementary-polygon legend label
legshow(*numlist*) display only selected keys of supplementary-polygon legend
legcount display number of supplementary polygons belonging to each group

A.1.3 line_suboptions

Main

data(*line*) Stata dataset defining one or more polylines to be superimposed onto the base map[3]
select(*command*) keep/drop specified records of dataset *line*
by(*byvar_ln*) group polylines by variable *byvar_ln*

Format

color(*colorlist*) polyline color
size(*linewidthstyle_list*) polyline thickness
pattern(*linepatternstyle_list*) polyline pattern

Legend

legenda(on|off) display/hide polyline legend
legtitle(*string*) polyline legend title
leglabel(*string*) single-key polyline legend label
legshow(*numlist*) display only selected keys of polyline legend
legcount display number of polylines belonging to each group

A.1.4 point_suboptions

Main

data(*point*) Stata dataset defining one or more points to be superimposed onto the base map
select(*command*) keep/drop specified records of dataset *point*

[3] Required when option line() is specified

by(*byvar_pn*) group points by variable *byvar_pn*
xcoord(*xvar_pn*) variable containing the x-coordinate of points[4]
ycoord(*yvar_pn*) variable containing the y-coordinate of points[5]

Proportional size

proportional(*propvar_pn*) draw point markers with size proportional to variable *propvar_pn*
prange(*min, max*) normalization range of variable *propvar_pn*
psize(relative|absolute) reference system for drawing point markers

Deviation

deviation(*devvar_pn*) draw point markers as deviations from given reference value of variable *devvar_pn*
refval(mean|median|#) reference value of variable *devvar_pn*
refweight(*weightvar_pn*) compute reference value of variable *devvar_pn* weighting observations by variable *weightvar_pn*
dmax(#) absolute value of maximum deviation

Format

size(*markersizestyle_list*) size of point markers
shape(*symbolstyle_list*) shape of point markers
fcolor(*colorlist*) fill color of point markers
ocolor(*colorlist*) outline color of point markers
osize(*linewidthstyle_list*) outline thickness of point markers

Legend

legenda(on|off) display/hide point legend
legtitle(*string*) point legend title
leglabel(*string*) single-key point legend label
legshow(*numlist*) display only selected keys of point legend
legcount display number of points belonging to each group

A.1.5 diagram_suboptions

Main

data(*diagram*) Stata dataset defining one or more diagrams to be superimposed onto the base map at given reference points
select(*command*) keep/drop specified records of dataset *diagram*
by(*byvar_dg*) group diagrams by variable *byvar_dg*

[4] Required when option point() is specified
[5] Required when option point() is specified

xcoord(*xvar_dg*) variable containing the x-coordinate of diagram reference points[6]
ycoord(*yvar_dg*) variable containing the y-coordinate of diagram reference points[7]
variables(*diagvar_dg*) variable or variables to be represented by diagrams[8]
type(frect|pie) diagram type

Proportional size

proportional(*propvar_dg*) draw diagrams with area proportional to variable *propvar_dg*
prange(*min, max*) reference range of variable *propvar_dg*

Framed-rectangle chart

range(*min, max*) reference range of variable *diagvar_dg*
refval(mean|median|#) reference value of variable *diagvar_dg*
refweight(*weightvar_dg*) compute the reference value of variable *diagvar_dg* weighting observations by variable *weightvar_dg*
refcolor(*colorstyle*) color of the line representing the reference value of variable *diagvar_dg*
refsize(*linewidthstyle*) thickness of the line representing the reference value of variable *diagvar_dg*

Format

size(#) diagram size
fcolor(*colorlist*) fill color of the diagrams
ocolor(*colorlist*) outline color of the diagrams
osize(*linewidthstyle_list*) outline thickness of the diagrams

Legend

legenda(on|off) display/hide diagram legend
legtitle(*string*) diagram legend title
legshow(*numlist*) display only selected keys of diagram legend
legcount display number of diagrams belonging to each group

A.1.6 arrow_suboptions

Main

data(*arrow*) Stata dataset defining one or more arrows to be superimposed onto the base map[9]
select(*command*) keep/drop specified records of dataset *arrow*

[6] Required when option diagram() is specified
[7] Required when option diagram() is specified
[8] Required when option diagram() is specified
[9] Required when option arrow() is specified

by(*byvar_ar*) group arrows by variable *byvar_ar*

Format

direction(*directionstyle_list*) arrow direction, where *directionstyle* is one of the following: 1 (monodirectional arrow), 2 (bidirectional arrow)
hsize(*markersizestyle_list*) arrowhead size
hangle(*anglestyle_list*) arrowhead angle
hbarbsize(*markersizestyle_list*) size of filled portion of arrowhead
hfcolor(*colorlist*) arrowhead fill color
hocolor(*colorlist*) arrowhead outline color
hosize(*linewidthstyle_list*) arrowhead outline thickness
lcolor(*colorlist*) arrow shaft line color
lsize(*linewidthstyle_list*) arrow shaft line thickness
lpattern(*linepatternstyle_list*) arrow shaft line pattern

Legend

legenda(on|off) display/hide arrow legend
legtitle(*string*) arrow legend title
leglabel(*string*) single-key arrow legend label
legshow(*numlist*) display only selected keys of arrow legend
legcount display number of arrows belonging to each group

A.1.7 label_suboptions

Main

data(*label*) Stata dataset defining one or more labels to be superimposed onto the base map at given reference points
select(*command*) keep/drop specified records of dataset *label*
by(*byvar_lb*) group labels by variable *byvar_lb*
xcoord(*xvar_lb*) variable containing the x-coordinate of label reference points[10]
ycoord(*yvar_lb*) variable containing the y-coordinate of label reference points[11]
label(*labvar_lb*) variable containing the labels[12]

Format

length(*lengthstyle_list*) maximum number of label characters, where *lengthstyle* is any integer>0
size(*textsizestyle_list*) label size
color(*colorlist*) label color
position(*clockpos_list*) position of labels relative to their reference point
gap(*relativesize_list*) gap between labels and their reference point
angle(*anglestyle_list*) label angle

[10] Required when option label() is specified
[11] Required when option label() is specified
[12] Required when option label() is specified

A.1.8 scalebar_suboptions

Main

units(#) scale bar extent[13]
scale(#) ratio of scale bar units to map units
xpos(#) scale bar horizontal position relative to plot region center
ypos(#) scale bar vertical position relative to plot region center

Format

size(#) scale bar height multiplier
fcolor(*colorstyle*) fill color of scale bar
ocolor(*colorstyle*) outline color of scale bar
osize(*linewidthstyle*) outline thickness of scale bar
label(*string*) scale bar label
tcolor(*colorstyle*) color of scale bar text
tsize(*textsizestyle*) size of scale bar text

A.1.9 graph_options

Main

polyfirst draw supplementary polygons before the base map
gsize(#) length of shortest side of *available area* (in inches)
freestyle ignore built-in graph formatting presets and restrictions
twoway_options any options documented in [**G**] *twoway_options, except for axis_options, aspect_option, scheme_option, by_option,* and *advanced_options*

A.2 description

spmap is aimed at visualizing several kinds of spatial data, and is particularly suited for drawing thematic maps and displaying the results of spatial data analyses.

spmap functioning rests on three basic principles:

- First, a base map representing a given region of interest R made up of N polygons is drawn.
- Second, at the user's choice, one or more kinds of additional spatial objects may be superimposed onto the base map. In the current version of spmap, six different kinds of spatial objects can be superimposed onto the base map: polygons (via option polygon()), polylines (via option line()), points (via option point()), diagrams (via option diagram()), arrows (via option arrow()), and labels (via option label()).

[13] Required when option scalebar() is specified

- Third, at the user's choice, one or more additional map elements may be added, such as a scale bar (via option `scalebar()`), a title, a subtitle, a note, and a caption (via *title_options*).

Proper specification of `spmap` options and suboptions, combined with the availability of properly formatted spatial data, allows the user to draw several kinds of maps, including choropleth maps, proportional symbol maps, pin maps, pie chart maps, and noncontiguous area cartograms.

While providing sensible defaults for most options and supoptions, `spmap` gives the user full control over the formatting of almost every map element, thus allowing the production of highly customized maps.

A.3 Spatial data format

`spmap` requires that the spatial data to be visualized be arranged into properly formatted Stata datasets. Such datasets can be classified into nine categories: *master*, *basemap*, *backgroundmap*, *polygon*, *line*, *point*, *diagram*, *arrow*, *label*.

The master dataset is the dataset that resides in memory when `spmap` is invoked. At the minimum, it must contain variable idvar, a numeric variable that uniquely identifies the polygon or polygons making up the base map. If a choropleth map is to be drawn, then the master dataset should contain also variable attribute, a numeric variable expressing the values of the feature to be represented. Additionally, if a noncontiguous area cartogram is to be drawn - i.e., if the polygons making up the base map are to be drawn with area proportional to the values of a given numeric variable areavar - then the master dataset should contain also variable areavar.
A basemap dataset is a Stata dataset that contains the definition of the polygon or polygons making up the base map. A basemap dataset is required to have the following structure:

_ID	_X	_Y	_EMBEDDED
1	.	.	0
1	10	30	0
1	10	50	0
1	30	50	0
1	30	30	0
1	10	30	0
2	.	.	0
2	10	10	0
2	10	30	0
2	18	30	0
2	18	10	0

```
  2        10        10         0
  2         .         .         0
  2        22        10         0
  2        22        30         0
  2        30        30         0
  2        30        10         0
  2        22        10         0
  3         .         .         1
  3        15        35         1
  3        15        45         1
  3        25        45         1
  3        25        35         1
  3        15        35         1
-----------------------------------
```

_ID is required and is a numeric variable that uniquely identifies the polygons making up the base map. _X is required and is a numeric variable that contains the x-coordinate of the nodes of the base map polygons. _Y is required and is a numeric variable that contains the y-coordinate of the nodes of the base map polygons. Finally, _EMBEDDED is optional and is an indicator variable taking value 1 if the corresponding polygon is completely enclosed in another polygon, and value 0 otherwise. The following should be noticed:

- Both simple and multipart polygons are allowed. In the example above, polygons 1 and 3 are simple (i.e., they consist of a single area), while polygon 2 is multipart (i.e., it consists of two distinct areas).
- The first record of each simple polygon or of each part of a multipart polygon must contain missing x- and y-coordinates.
- The non-missing coordinates of each simple polygon or of each part of a multipart polygon must be ordered so as to correspond to consecutive nodes.
- Each simple polygon or each part of a multipart polygon must be "closed", i.e., the last pair of non-missing coordinates must be equal to the first pair.
- A *basemap* dataset is always required to be sorted by variable _ID.

A *backgroundmap* dataset is a Stata dataset that contains the definition of the polygon or polygons making up the background map (a map that can be optionally drawn as background of a noncontiguous area cartogram). A backgroundmap dataset has exactly the same structure as a *basemap* dataset, except for variable _EMBEDDED that is never used.

A *polygon* dataset is a Stata dataset that contains the definition of one or more supplementary polygons to be superimposed onto the base map. A *polygon* dataset is required to have the following structure:

_ID	_X	_Y	byvar_pl
1	.	.	1
1	20	40	1
1	20	42	1
1	25	42	1
1	25	40	1
1	20	40	1
2	.	.	1
2	11	20	1
2	11	25	1
2	13	25	1
2	13	20	1
2	11	20	1
3	.	.	2
3	25	25	2
3	25	35	2
3	30	35	2
3	30	25	2
3	25	25	2

Variables _ID, _X, and _Y are defined exactly in the same way as in a *basemap* dataset, with the sole exception that only simple polygons are allowed. In turn, *byvar_pl* is a placeholder denoting an optional variable that can be specified to distinguish different kinds of supplementary polygons.

A *line* dataset is a Stata dataset that contains the definition of one or more polylines to be superimposed onto the base map. A *line* dataset is required to have the following structure:

_ID	_X	_Y	byvar_ln
1	.	.	1
1	11	30	1
1	12	33	1
1	15	33	1
1	16	35	1
1	18	40	1
1	25	38	1
1	25	42	1
2	.	.	2
2	12	20	2
2	18	15	2

```
    3         .         .         2
    3        27        28         2
    3        27        25         2
    3        28        27         2
    3        29        25         2
-------------------------------------
```

_ID is required and is a numeric variable that uniquely identifies the polylines. _X is required and is a numeric variable that contains the x-coordinate of the nodes of the polylines. _Y is required and is a numeric variable that contains the y-coordinate of the nodes of the polylines. Finally, *byvar_ln* is a placeholder denoting an optional variable that can be specified to distinguish different kinds of polylines. The following should be noticed:

- The first record of each polyline must contain missing x- and y-coordinates.
- The non-missing coordinates of each polyline must be ordered so as to correspond to consecutive nodes.

A *point* dataset is a Stata dataset that contains the definition of one or more points to be superimposed onto the base map. A *point* dataset is required to have the following structure:

```
 xvar_pn  yvar_pn  byvar_pn  propvar_pn  devvar_pn  weightvar_pn
-------------------------------------------------------------------
      11       30         1         100         30          1000
      20       34         1         110         25          1500
      25       40         1          90         40          1230
      25       45         2         200         10           950
      15       20         2          50         70           600
-------------------------------------------------------------------
```

xvar_pn is a placeholder denoting a required numeric variable that contains the x-coordinate of the points. *yvar_pn* is a placeholder denoting a required numeric variable that contains the y-coordinate of the points. *byvar_pn* is a placeholder denoting an optional variable that can be specified to distinguish different kinds of points. *propvar_pn* is a placeholder denoting an optional variable that, when specified, requests that the point markers be drawn with size proportional to *propvar_pn*. *devvar_pn* is a placeholder denoting an optional variable that, when specified, requests that the point markers be drawn as deviations from a given reference value of *devvar_pn*. Finally, *weightvar_pn* is a placeholder denoting an optional variable that, when specified, requests that the reference value of *devvar_pn* be computed weighting observations by variable *weightvar_pn*. It is important to note that the required and optional variables making up a point dataset can either reside in an external dataset or be part of the master dataset.

A *diagram* dataset is a Stata dataset that contains the definition of one or more diagrams to be superimposed onto the base map at given reference points. A *diagram* dataset is required to have the following structure:

xvar_dg	yvar_dg	byvar_dg	diagvar_dg	propvar_dg	weightvar_dg
15	30	1	...	30	1000
18	40	1	...	25	1500
20	45	1	...	40	1230
25	45	2	...	10	950
15	20	2	...	70	600

xvar_dg is a placeholder denoting a required numeric variable that contains the x-coordinate of the diagram reference points. *yvar_dg* is a placeholder denoting a required numeric variable that contains the y-coordinate of the diagram reference points. *byvar_dg* is a placeholder denoting an optional variable that can be specified to distinguish different groups of diagrams. *diagvar_dg* is a placeholder denoting one or more variables to be represented by the diagrams. *propvar_dg* is a placeholder denoting an optional variable that, when specified, requests that the diagrams be drawn with area proportional to *propvar_dg*. Finally, *weightvar_dg* is a placeholder denoting an optional variable that, when specified, requests that the reference value of the diagrams be computed weighting observations by variable *weightvar_dg* (this applies only to framed-rectangle charts). It is important to note that the required and optional variables making up a *diagram* dataset can either reside in an external dataset or be part of the *master* dataset.

An *arrow* dataset is a Stata dataset that contains the definition of one or more arrows to be superimposed onto the base map. An *arrow* dataset is required to have the following structure:

_ID	_X1	_Y1	_X2	_Y2	byvar_ar
1	11	30	18	30	1
2	15	40	15	45	1
3	15	40	25	40	1
4	20	35	28	45	2
5	17	20	20	11	2

_ID is required and is a numeric variable that uniquely identifies the arrows. _X1 is required and is a numeric variable that contains the x-coordinate of the starting point of the arrows. _Y1 is required and is a numeric variable that contains the y-coordinate of the starting point of the arrows. _X2 is required and is a numeric variable that

contains the x-coordinate of the ending point of the arrows. _Y2 is required and is a numeric variable that contains the y-coordinate of the ending point of the arrows. Finally, byvar_ar is a placeholder denoting an optional variable that can be specified to distinguish different kinds of arrows.

A *label* dataset is a Stata dataset that contains the definition of one or more labels to be superimposed onto the base map at given reference points. A *label* dataset is required to have the following structure:

xvar_lb	yvar_lb	byvar_lb	labvar_lb
11	33	1	Abcde
20	37	1	Fgh
25	43	1	IJKL
25	48	2	Mnopqr
15	22	2	stu

xvar_lb is a placeholder denoting a required numeric variable that contains the x-coordinate of the label reference points. *yvar_lb* is a placeholder denoting a required numeric variable that contains the y-coordinate of the label reference points. *byvar_lb* is a placeholder denoting an optional variable that can be specified to distinguish different kinds of labels. Finally, *labvar_lb* is a placeholder denoting the variable that contains the labels. It is important to note that the required and optional variables making up a *label* dataset can either reside in an external dataset or be part of the *master* dataset.

A.4 Color lists

Some `spmap` options and suboptions request the user to specify a list of one or more colors. When the list includes only one color, the user is required to specify a standard *colorstyle*. On the other hand, when the list includes two or more colors, the user can either specify a standard *colorstyle list*, or specify the name of a predefined color scheme.

The following table lists the predefined color schemes available in the current version of `spmap`, indicating the name of each scheme, the maximum number of different colors it allows, its type, and its source.

NAME	MAXCOL	TYPE	SOURCE
Blues	9	Sequential	Brewer
Blues2	99	Sequential	Custom
BuGn	9	Sequential	Brewer
BuPu	9	Sequential	Brewer
GnBu	9	Sequential	Brewer

```
      Greens          9   Sequential     Brewer
     Greens2         99   Sequential     Custom
       Greys          9   Sequential     Brewer
      Greys2         99   Sequential     Brewer
        Heat         16   Sequential     Custom
        OrRd          9   Sequential     Brewer
     Oranges          9   Sequential     Brewer
        PuBu          9   Sequential     Brewer
      PuBuGn          9   Sequential     Brewer
        PuRd          9   Sequential     Brewer
     Purples          9   Sequential     Brewer
     Rainbow         99   Sequential     Custom
        RdPu          9   Sequential     Brewer
        Reds          9   Sequential     Brewer
       Reds2         99   Sequential     Custom
     Terrain         16   Sequential     Custom
 Topological         16   Sequential     Custom
        YlGn          9   Sequential     Brewer
      YlGnBu          9   Sequential     Brewer
      YlOrBr          9   Sequential     Brewer
      YlOrRd          9   Sequential     Brewer
        BrBG         11    Diverging     Brewer
        BuRd         11    Diverging     Custom
      BuYlRd         11    Diverging     Custom
        PRGn         11    Diverging     Brewer
        PiYG         11    Diverging     Brewer
        PuOr         11    Diverging     Brewer
        RdBu         11    Diverging     Brewer
        RdGy         11    Diverging     Brewer
      RdYlBu         11    Diverging     Brewer
      RdYlGn         11    Diverging     Brewer
    Spectral         11    Diverging     Brewer
      Accent          8  Qualitative     Brewer
       Dark2          8  Qualitative     Brewer
      Paired         12  Qualitative     Brewer
     Pastel1          9  Qualitative     Brewer
     Pastel2          8  Qualitative     Brewer
        Set1          9  Qualitative     Brewer
        Set2          8  Qualitative     Brewer
        Set3         12  Qualitative     Brewer
------------------------------------------------
```

Following Brewer (1999), *sequential schemes* are typically used to represent ordered data, so that higher data values are represented by darker colors; in turn, *diverging schemes*

are used when there is a meaningful midpoint in the data, to emphasize progressive divergence from this midpoint in the two opposite directions; finally, *qualitative schemes* are generally used to represent unordered, categorical data.
The color schemes whose source is indicated as "Brewer" were designed by Dr. Cynthia A. Brewer, Department of Geography, The Pennsylvania State University, University Park, Pennsylvania, USA (Brewer et al. 2003). These color schemes are used with Dr. Brewer's permission and are taken from the ColorBrewer map design tool available at ColorBrewer.org.

A.5 Choropleth maps

A choropleth map can be defined as a map in which each subarea (e.g., each census tract) of a given region of interest (e.g., a city) is colored or shaded with an intensity proportional to the value taken on by a given quantitative variable in that subarea (Slocum et al. 2005). Since choropleth maps are one of the most popular means for representing the spatial distribution of quantitative variables, it is worth noting the way `spmap` can be used to draw this kind of map.
In `spmap`, a choropleth map is a base map whose constituent polygons are colored according to the values taken on by attribute, a numeric variable that must be contained in the *master* dataset and specified immediately after the main command (see syntax diagram above). To draw the desired choropleth map, `spmap` first groups the values taken on by variable attribute into k classes defined by a given set of class breaks, and then assigns a different color to each class. The current version of `spmap` offers six methods for determining class breaks:

- *Quantiles*: class breaks correspond to quantiles of the distribution of variable *attribute*, so that each class includes approximately the same number of polygons.

- *Boxplot*: the distribution of variable *attribute* is divided into 6 classes defined as follows: [min, p25 - 1.5*iqr], (p25 - 1.5*iqr, p25], (p25, p50], (p50, p75], (p75, p75 + 1.5*iqr] and (p75 + 1.5*iqr, max], where iqr = interquartile range.

- *Equal intervals*: class breaks correspond to values that divide the distribution of variable *attribute* into k equal-width intervals.

- *Standard deviates*: the distribution of variable *attribute* is divided into k classes (2 <= k <= 9) whose width is defined as a fraction p of its standard deviation sd. Following the suggestions of Evans (1977), this proportion p varies with k as follows:

```
   k       p
------------
   2     inf
   3     1.2
```

```
  4     1.0
  5     0.8
  6     0.8
  7     0.8
  8     0.6
  9     0.6
------------
```

Class intervals are centered on the arithmetic mean m, which is a class midpoint if k is odd and a class boundary if k is even; the lowest and highest classes are open-ended (Evans 1977).

- *k-means*: the distribution of variable *attribute* is divided into k classes using k-means partition cluster analysis. The clustering procedure is applied several times to variable attribute, and the solution that maximizes the goodness-of-variance fit (Armstrong et al. 2003) is used.

- *Custom*: class breaks are specified by the user.

Alternatively, `spmap` allows the user to leave the values of variable attribute ungrouped. In this case, attribute is treated as a categorical variable and a different color is assigned to each of its values.

A.6 Options for drawing the base map

Main

`id(`*idvar*`)` specifies the name of a numeric variable that uniquely identifies the polygon or polygons making up the base map. *idvar* must be contained in the *master* dataset, and its values must correspond to the values taken on by variable _ID contained in the *basemap* dataset.

Cartogram

`area(`*areavar*`)` requests that the polygons making up the base map be drawn with area proportional to the values taken on by numeric variable *areavar*, so that a non-contiguous area cartogram (Olson 1976) is obtained. *areavar* must be contained in the *master* dataset.

`split` requests that, before drawing a noncontiguous area cartogram, all multipart base map polygons be split into their constituent parts, each of which will then be treated as a distinct simple polygon.

`map(`*backgroundmap*`)` requests that, when drawing a noncontiguous area cartogram, the polygons making up the base map be superimposed onto a background map defined in Stata dataset *backgroundmap*.

`mfcolor(`*colorstyle*`)` specifies the fill color of the background map. The default is `mfcolor(none)`.

`mocolor(`*colorstyle*`)` specifies the outline color of the background map. The default is `mocolor(black)`.

`mosize(`*linewidthstyle*`)` specifies the outline thickness of the background map. The default is `mosize(thin)`.

`mopattern(`*linepatternstyle*`)` specifies the outline pattern of the background map. The default is `mopattern(solid)`.

Choropleth map

`clmethod(`*method*`)` specifies the method to be used for classifying variable *attribute* and representing its spatial distribution as a choropleth map.

`clmethod(quantile)` is the default and requests that the quantiles method be used.

`clmethod(boxplot)` requests that the boxplot method be used.

`clmethod(eqint)` requests that the equal intervals method be used.

`clmethod(stdev)` requests that the standard deviates method be used.

`clmethod(kmeans)` requests that the k-means method be used.

`clmethod(custom)` requests that class breaks be specified by the user with option `clbreaks(`*numlist*`)`.

`clmethod(unique)` requests that each value of variable *attribute* be treated as a distinct class.

`clnumber(#)` specifies the number of classes k in which variable *attribute* is to be divided. When the quantiles, equal intervals, standard deviates, or k-means classification method is chosen, the default is `clnumber(4)`. When the boxplot classification method is chosen, this option is inactive and k=6. When the custom classification method is chosen, this option is inactive and k equals the number of elements of numlist specified in option `clbreaks(`*numlist*`)` minus 1. When the unique classification method is chosen, this option is inactive and k equals the number of different values taken on by variable *attribute*.

`clbreaks(`*numlist*`)` is required when option `clmethod(custom)` is specified. It defines the custom class breaks to be used for classifying variable *attribute*. *numlist* should be specified so that the first element is the minimum value of variable *attribute* to be considered; the second to kth elements are the class breaks; and the last element is the maximum value of variable *attribute* to be considered. For example, suppose we want to group the values of variable attribute into the following four classes: [10,15], (15,20], (20,25] and (25,50]; for this we must specify `clbreaks(10 15 20 25 50)`.

`eirange(`*min, max*`)` specifies the range of values (minimum and maximum) to be considered in the calculation of class breaks when option `clmethod(eqint)` is specified. This option overrides the default range [min(*attribute*), max(*attribute*)].

`kmiter(#)` specifies the number of times the clustering procedure is applied when option `clmethod(kmeans)` is specified. The default is `kmiter(20)`.

`ndfcolor(`*colorstyle*`)` specifies the fill color of the empty (no data) polygons of the choropleth map. The default is `ndfcolor(white)`.

`ndocolor(`*colorstyle*`)` specifies the outline color of the empty (no data) polygons of the choropleth map. The default is `ndocolor(black)`.

`ndsize(`*linewidthstyle*`)` specifies the outline thickness of the empty (no data) polygons of the choropleth map. The default is `ndsize(thin)`.

`ndlabel(`*string*`)` specifies the legend label to be attached to the empty (no data) polygons of the choropleth map. The default is `ndlabel(No data)`.

Format

`fcolor(`*colorlist*`)` specifies the list of fill colors of the base map polygons. When no choropleth map is drawn, the list should include only one element. On the other hand, when a choropleth map is drawn, the list should be either composed of k elements, or represented by the name of a predefined color scheme. The default fill color is `none`. When a choropleth map is drawn, the default argument is a color scheme that depends on the classification method specified in option `clmethod(method)`:

Classification method	Default color scheme
quantile	Greys
boxplot	BuRd
eqint	Greys
stdev	BuRd
kmeans	Greys
custom	Greys
unique	Paired

`ocolor(`*colorlist*`)` specifies the list of outline colors of the base map polygons. When no choropleth map is drawn, the list should include only one element. On the other hand, when a choropleth map is drawn, the list should be either composed of k elements, or represented by the name of a predefined color scheme. The default outline color is `black`, the default specification is `ocolor(black ...)`.

`osize(`*linewidthstyle_list*`)` specifies the list of outline thicknesses of the base map polygons. When no choropleth map is drawn, the list should include only one element. On the other hand, when a choropleth map is drawn, the list should be composed of k elements. The default outline thickness is `thin`, the default specification is `osize(thin ...)`.

Legend

`legenda(on|off)` specifies whether the base map legend should be displayed or hidden.

`legenda(on)` requests that the base map legend be displayed. This is the default when a choropleth map is drawn.

`legenda(off)` requests that the base map legend be hidden. This is the default when no choropleth map is drawn.

`legtitle(`*string*`)` specifies the title of the base map legend. When a choropleth map is drawn, option `legtitle(varlab)` requests that the label of variable *attribute* be used as the legend title.

`leglabel(`*string*`)` specifies the label to be attached to the single key of the base map legend when no choropleth map is drawn. This option is required when option `legenda(on)` is specified and no choropleth map is drawn.

`legorder(hilo|lohi)` specifies the display order of the keys of the base map legend when a choropleth map is drawn.

`legorder(hilo)` is the default and requests that the keys of the base map legend be displayed in descending order of variable *attribute*.

`legorder(lohi)` requests that the keys of the base map legend be displayed in ascending order of variable *attribute*. This is the default when option `clmethod(unique)` is specified.

`legstyle(0|1|2|3)` specifies the way the keys of the base map legend are labelled when a choropleth map is drawn.

`legstyle(0)` requests that the keys of the base map legend not be labelled.

`legstyle(1)` is the default and requests that the keys of the base map legend be labelled using the standard mathematical notation for value intervals (e.g.: (20,35]).

`legstyle(2)` requests that the keys of the base map legend be labelled using the notation ll&ul, where ll denotes the lower limit of the class interval, ul denotes the upper limit of the class interval, and & denotes a string that separates the two values. For example, if ll=20, ul=35, and &=" - ", then the resulting label will be "20 - 35".

`legstyle(3)` requests that only the first and last keys of the base map legend be labelled; the first key is labelled with the lower limit of the corresponding class interval, the last key is labelled with the upper limit of the corresponding class interval.

`legjunction(`*string*`)` specifies the string to be used as separator when option `legstyle(2)` is specified. The default is `legjunction(" - ")`.

`legcount` requests that, when a choropleth map is drawn, the number of base map polygons belonging to each class of variable *attribute* be displayed in the legend.

A.7 Option polygon() suboptions

Main

`data(`*polygon*`)` requests that one or more supplementary polygons defined in Stata dataset polygon be superimposed onto the base map.

`select(`*command*`)` requests that a given subset of records of dataset *polygon* be selected using Stata commands keep or drop.

`by(`*byvar_pl*`)` indicates that the supplementary polygons defined in dataset *polygon* belong to kpl different groups specified by variable *byvar_pl*.

Format

`fcolor(`*colorlist*`)` specifies the list of fill colors of the supplementary polygons. When suboption `by(`*byvar_pl*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_pl*`)` is specified, the list should be either composed of kpl elements, or represented by the name of a predefined color scheme. The default fill color is `none`, the default specification is `fcolor(none ...)`.

`ocolor(`*colorlist*`)` specifies the list of outline colors of the supplementary polygons. When suboption `by(`*byvar_pl*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_pl*`)` is specified, the list should be either composed of kpl elements, or represented by the name of a predefined color scheme. The default outline color is `black`, the default specification is `ocolor(black ...)`.

`osize(`*linewidthstyle_list*`)` specifies the list of outline thicknesses of the supplementary polygons. When suboption `by(`*byvar_pl*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_pl*`)` is specified, the list should be composed of kpl elements. The default outline thickness is `thin`, the default specification is `osize(thin ...)`.

Legend

`legenda(on|off)` specifies whether the supplementary-polygon legend should be displayed or hidden.

`legenda(on)` requests that the supplementary-polygon legend be displayed.

`legenda(off)` is the default and requests that the supplementary-polygon legend be hidden.

`legtitle(`*string*`)` specifies the title of the supplementary-polygon legend. When suboption `by(`*byvar_pl*`)` is specified, suboption `legtitle(varlab)` requests that the label of variable *byvar_pl* be used as the legend title.

`leglabel(`*string*`)` specifies the label to be attached to the single key of the supplementary-polygon legend when suboption `by(`*byvar_pl*`)` is not specified. This suboption is required when suboption `legenda(on)` is specified and suboption `by(`*byvar_pl*`)` is not specified.

`legshow(`*numlist*`)` requests that, when suboption `by(`*byvar_pl*`)` is specified, only the keys included in *numlist* be displayed in the supplementary-polygon legend.

`legcount` requests that the number of supplementary polygons be displayed in the legend.

A.8 Option line() suboptions

Main

`data(`*line*`)` requests that one or more polylines defined in Stata dataset *line* be superimposed onto the base map.

`select(`*command*`)` requests that a given subset of records of dataset *line* be selected using Stata commands `keep` or `drop`.

`by(`*byvar_ln*`)` indicates that the polylines defined in dataset *line* belong to kln different groups specified by variable *byvar_ln*.

Format

`color(`*colorlist*`)` specifies the list of polyline colors. When suboption `by(`*byvar_ln*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_ln*`)` is specified, the list should be either composed of kln elements, or represented by the name of a predefined color scheme. The default color is `black`, the default specification is `color(black ...)`.

`size(`*linewidthstyle_list*`)` specifies the list of polyline thicknesses. When suboption `by(`*byvar_ln*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_ln*`)` is specified, the list should be composed of kln elements. The default thickness is `thin`, the default specification is `size(thin ...)`.

`pattern(`*linepatternstyle_list*`)` specifies the list of polyline patterns. When suboption `by(`*byvar_ln*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_ln*`)` is specified, the list should be composed of kln elements. The default pattern is `solid`, the default specification is `pattern(solid ...)`.

Legend

`legenda(on|off)` specifies whether the polyline legend should be displayed or hidden.

`legenda(on)` requests that the polyline legend be displayed.
`legenda(off)` is the default and requests that the polyline legend be hidden.

`legtitle(`*string*`)` specifies the title of the polyline legend. When suboption `by(`*byvar_ln*`)` is specified, suboption `legtitle(varlab)` requests that the label of variable (*byvar_ln*) be used as the legend title.

`leglabel(`*string*`)` specifies the label to be attached to the single key of the polyline legend when suboption `by(`*byvar_ln*`)` is not specified. This suboption is required when suboption `legenda(on)` is specified and suboption `by(`*byvar_ln*`)` is not specified.

`legshow(`*numlist*`)` requests that, when suboption `by(`*byvar_ln*`)` is specified, only the keys included in *numlist* be displayed in the polyline legend.

`legcount` requests that the number of polylines be displayed in the legend.

A.9 Option point() suboptions

Main

`data(`*point*`)` requests that one or more points defined in Stata dataset *point* be superimposed onto the base map.

`select(`*command*`)` requests that a given subset of records of dataset *point* be selected using Stata commands `keep` or `drop`.

`by(`*byvar_pn*`)` indicates that the points defined in dataset *point* belong to kpn different groups specified by variable *byvar_pn*.

`xcoord(`*xvar_pn*`)` specifies the name of the variable containing the x-coordinate of each point.

`ycoord(`*yvar_pn*`)` specifies the name of the variable containing the y-coordinate of each point.

Proportional size

`proportional(`*propvar_pn*`)` requests that the point markers be drawn with size proportional to the values taken on by numeric variable *propvar_pn*.

`prange(`*min, max*`)` requests that variable *propvar_pn* specified in suboption `proportional(`*propvar_pn*`)` be normalized based on range [*min, max*]. This suboption overrides the default normalization based on range [0, max(*propvar_pn*)].

`psize(relative|absolute)` specifies the reference system for drawing the point markers.

`psize(relative)` is the default and requests that the point markers be drawn using relative minimum and maximum reference values. This is the best choice when there is no need to compare the map at hand with other maps of the same kind.

`psize(absolute)` requests that the point markers be drawn using absolute minimum and maximum reference values. This is the best choice when the map at hand is to be compared with other maps of the same kind.

Deviation

`deviation(`*devvar_pn*`)` requests that the point markers be drawn as deviations from a reference value of numeric variable *devvar_pn* specified in option `refval()`. When this suboption is specified, in the first place the values of variable *devvar_pn* are re-expressed as deviations from the chosen reference value. Then, points associated with positive deviations are represented by solid markers, whereas points associated with negative deviations are represented by hollow markers of the same shape; in both cases, markers are drawn with size proportional to the absolute value of the deviation. This suboption is incompatible with suboption `proportional(`*propvar_pn*`)`.

`refval(mean|median|#)` specifies the reference value of variable *devvar_pn* for computing deviations.

`refval(mean)` is the default and requests that the arithmetic mean of variable *devvar_pn* be taken as the reference value.

`refval(median)` requests that the median of variable *devvar_pn* be taken as the reference value.

`refval(#)` requests that an arbitrary real value #be taken as the reference value.

`refweight(`*weightvar_pn*`)` requests that the reference value of variable *devvar_pn* be computed weighting observations by values of variable *weightvar_pn*.

`dmax(#)` requests that the point markers be drawn using value #as the maximum absolute deviation of reference.

Format

`size(`*markersizestyle_list*`)` specifies the list of point marker sizes. When suboption `by(`*byvar_pn*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_pn*`)` is specified, the list should be composed of kpn elements. The default size is `*1`, the default specification is `size(*1 ...)`.

`shape(`*symbolstyle_list*`)` specifies the list of point marker shapes. When suboption `by(`*byvar_pn*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_pn*`)` is specified, the list should be composed of kpn elements. The default shape is `o`, the default specification is `shape(o ...)`. When suboption `deviation(`*devvar_pn*`)` is specified, this suboption accepts only solid *symbolstyles* written in short form: `O D T S o d t s`.

`fcolor(`*colorlist*`)` specifies the list of fill colors of the point markers. When suboption `by(`*byvar_pn*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_pn*`)` is specified, the list should be either composed of kpn elements, or represented by the name of a predefined color scheme. The default fill color is `black`, the default specification is `fcolor(black ...)`.

`ocolor(`*colorlist*`)` specifies the list of outline colors of the point markers. When suboption `by(`*byvar_pn*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_pn*`)` is specified, the list should be either composed of kpn elements, or represented by the name of a predefined color scheme. The default outline color is `none`, the default specification is `ocolor(none ...)`.

`osize(`*linewidthstyle_list*`)` specifies the list of outline thicknesses of the point markers. When suboption `by(`*byvar_pn*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_pn*`)` is specified, the list should be composed of kpl elements. The default outline thickness is `thin`, the default specification is `osize(thin ...)`.

Legend

`legenda(on|off)` specifies whether the point legend should be displayed or hidden.

`legenda(on)` requests that the point legend be displayed.

`legenda(off)` is the default and requests that the point legend be hidden.

`legtitle(`*string*`)` specifies the title of the point legend. When suboption `by(`*byvar_pn*`)` is specified, suboption `legtitle(varlab)` requests that the label of variable *byvar_pn* be used as the legend title.

`leglabel(`*string*`)` specifies the label to be attached to the single key of the point legend when suboption `by(`*byvar_pn*`)` is not specified. This suboption is required when suboption `legenda(on)` is specified and suboption `by(`*byvar_pn*`)` is not specified.

`legshow(`*numlist*`)` requests that, when suboption `by(`*byvar_pn*`)` is specified, only the keys included in *numlist* be displayed in the point legend.

`legcount` requests that the number of points be displayed in the legend.

A.10 Option diagram() suboptions

Main

`data(`*diagram*`)` requests that one or more diagrams defined in Stata dataset *diagram* be superimposed onto the base map at given reference points.

`select(`*command*`)` requests that a given subset of records of dataset *diagram* be selected using Stata commands `keep` or `drop`.

`by(`*byvar_dg*`)` indicates that the diagrams defined in dataset *diagram* belong to kdg different groups specified by variable *byvar_dg*. This option is active only when just one variable is specified in suboption `variables(`*diagvar_dg*`)`.

`xcoord(`*xvar_dg*`)` specifies the name of the variable containing the x-coordinate of each diagram reference point.

`ycoord(`*yvar_dg*`)` specifies the name of the variable containing the y-coordinate of each diagram reference point.

`variables(`*diagvar_dg*`)` specifies the list of variables to be represented by the diagrams.

`type(frect|pie)` specifies the type of diagram to be used.

`type(frect)` is the default when only one variable is specified in suboption `variables(`*diagvar_dg*
and requests that framed-rectangle charts (Cleveland and McGill 1984; Cleveland 1994) be used.

`type(pie)` is the default (and the only possibility) when two or more variables are specified in suboption `variables(`*diagvar_dg*`)` and requests that pie charts be used. When option `type(pie)` is specified, the variables specified in suboption `variables(`*diagvar_dg*`)` are rescaled so that they sum to 1 within each observation.

Proportional size

`proportional(`*propvar_dg*`)` requests that the diagrams be drawn with size proportional to the values taken on by numeric variable *propvar_dg*.

`prange(`*min, max*`)` requests that variable *propvar_dg* specified in suboption `proportional(`*propvar_dg*
be normalized based on range [*min, max*]. This suboption overrides the default normalization based on range [0, max(*propvar_dg*)].

Framed-rectangle chart

`range(`*min, max*`)` requests that variable *diagvar_dg* specified in suboption `variables(`*diagvar_dg*`)` be normalized based on range [*min, max*]. This suboption overrides the default normalization based on range [0, max(*diagvar_dg*)].

`refval(mean|median|#)` specifies the reference value of variable *diagvar_dg* for drawing the reference line.

`refval(mean)` is the default and requests that the arithmetic mean of variable *diagvar_dg* be taken as the reference value.

`refval(median)` requests that the median of variable *diagvar_dg* be taken as the reference value.

`refval(#)` requests that an arbitrary real value #be taken as the reference value.

`refweight(`*weightvar_dg*`)` requests that the reference value of variable *diagvar_dg* be computed weighting observations by values of variable *weightvar_dg*.

`refcolor(`*colorstyle*`)` specifies the color of the reference line. The default is `refcolor(black)`.

`refsize(`*linewidthstyle*`)` specifies the thickness of the reference line. The default is `refsize(medium)`.

Format

`size(#)` specifies a multiplier that affects the size of the diagrams. For example, `size(1.5)` requests that the default size of all the diagrams be increased by 50%. The default is `size(1)`.

`fcolor(`*colorlist*`)` specifies the list of fill colors of the diagrams. When just one variable is specified in suboption `variables(`*diagvar_dg*`)` and suboption `by(`*byvar_dg*`)` is not specified, the list should include only one element. When just one variable is specified in suboption `variables(`*diagvar_dg*`)` and suboption `by(`*byvar_dg*`)` is specified, the list should be either composed of kdg elements, or represented by the name of a predefined color scheme. Finally, when J>1 variables are specified in suboption `variables(`*diagvar_dg*`)`, the list should be either composed of J elements, or represented by the name of a predefined color scheme. The default fill color is `black`, the default specification when J=1 is `fcolor(black ...)`, and the default specification when J>1 is `fcolor(red blue orange green lime navy sienna ltblue cranberry emerald eggshell magenta olive brown yellow dkgreen)`.

`ocolor(`*colorlist*`)` specifies the list of outline colors of the diagrams. When just one variable is specified in suboption `variables(`*diagvar_dg*`)` and suboption `by(`*byvar_dg*`)` is not specified, the list should include only one element. When just one variable is specified in suboption `variables(`*diagvar_dg*`)` and suboption `by(`*byvar_dg*`)` is specified, the list should be either composed of kdg elements, or represented by the name of a predefined color scheme. Finally, when J>1 variables are specified in suboption `variables(`*diagvar_dg*`)`, the list should be either composed of J elements, or represented by the name of a predefined color scheme. The default fill color is `black`, the default specification is `ocolor(black ...)`.

`osize(`*linewidthstyle_list*`)` specifies the list of outline thicknesses of the diagrams. When just one variable is specified in suboption `variables(`*diagvar_dg*`)` and suboption `by(`*byvar_dg*`)` is not specified, the list should include only one element. When

just one variable is specified in suboption `variables(`*diagvar_dg*`)` and suboption `by(`*byvar_dg*`)` is specified, the list should be composed of kdg elements. Finally, when J>1 variables are specified in suboption `variables(`*diagvar_dg*`)`, the list should be composed of J elements. The default outline thickness is `thin`, the default specification is `osize(thin ...)`.

Legend

`legenda(on|off)` specifies whether the diagram legend should be displayed or hidden.

`legenda(on)` requests that the diagram legend be displayed.
`legenda(off)` is the default and requests that the point diagram be hidden.

`legtitle(`*string*`)` specifies the title of the diagram legend. When just one variable is specified in suboption `variables(`*diagvar_dg*`)`, suboption `legtitle(varlab)` requests that the label of variable *diagvar_dg* be used as the legend title.

`legshow(`*numlist*`)` requests that only the keys included in *numlist* be displayed in the diagram legend.

`legcount` requests that the number of diagrams be displayed in the legend.

A.11 Option arrow() suboptions

Main

`data(`*arrow*`)` requests that one or more arrows defined in Stata dataset *arrow* be superimposed onto the base map.

`select(`*command*`)` requests that a given subset of records of dataset *arrow* be selected using Stata commands `keep` or `drop`.

`by(`*byvar_ar*`)` indicates that the arrows defined in dataset *arrow* belong to kar different groups specified by variable *byvar_ar*.

Format

`direction(`*directionstyle_list*`)` specifies the list of arrow directions, where *directionstyle* is one of the following: `1` (monodirectional arrow), `2` (bidirectional arrow). When suboption `by(`*byvar_ar*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_ar*`)` is specified, the list should be composed of kar elements. The default direction is `1`, the default specification is `direction(1 ...)`.

`hsize(`*markersizestyle_list*`)` specifies the list of arrowhead sizes. When suboption `by(`*byvar_ar*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_ar*`)` is specified, the list should be composed of kar elements. The default size is `1.5`, the default specification is `hsize(1.5 ...)`.

`hangle(`*anglestyle_list*`)` specifies the list of arrowhead angles. When suboption `by(`*byvar_ar*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_ar*`)` is specified, the list should be composed of kar elements. The default angle is `28.64`, the default specification is `hangle(28.64 ...)`.

`hbarbsize(`*markersizestyle_list*`)` specifies the list of sizes of the filled portion of arrowheads. When suboption `by(`*byvar_ar*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_ar*`)` is specified, the list should be composed of kar elements. The default size is `1.5`, the default specification is `hbarbsize(1.5 ...)`.

`hfcolor(`*colorlist*`)` specifies the list of arrowhead fill colors. When suboption `by(`*byvar_ar*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_ar*`)` is specified, the list should be either composed of kar elements, or represented by the name of a predefined color scheme. The default fill color is `black`, the default specification is `hfcolor(black ...)`.

`hocolor(`*colorlist*`)` specifies the list of arrowhead outline colors. When suboption `by(`*byvar_ar*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_ar*`)` is specified, the list should be either composed of kar elements, or represented by the name of a predefined color scheme. The default outline color is `black`, the default specification is `hocolor(black ...)`.

`hosize(`*linewidthstyle_list*`)` specifies the list of arrowhead outline thicknesses. When suboption `by(`*byvar_ar*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_ar*`)` is specified, the list should be composed of kar elements. The default outline thickness is `thin`, the default specification is `hosize(thin ...)`.

`lcolor(`*colorlist*`)` specifies the list of arrow shaft line colors. When suboption `by(`*byvar_ar*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_ar*`)` is specified, the list should be either composed of kar elements, or represented by the name of a predefined color scheme. The default color is `black`, the default specification is `lcolor(black ...)`.

`lsize(`*linewidthstyle_list*`)` specifies the list of arrow shaft line thicknesses. When suboption `by(`*byvar_ar*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_ar*`)` is specified, the list should be composed of kar elements. The default thickness is `thin`, the default specification is `lsize(thin ...)`.

`lpattern(`*linepatternstyle_list*`)` specifies the list of arrow shaft line patterns. When suboption `by(`*byvar_ar*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_ar*`)` is specified, the list should be composed of kar elements. The default pattern is `solid`, the default specification is `lpattern(solid ...)`.

Legend

`legenda(on|off)` specifies whether the arrow legend should be displayed or hidden.

`legenda(on)` requests that the arrow legend be displayed.
`legenda(off)` is the default and requests that the arrow legend be hidden.

`legtitle(`*string*`)` specifies the title of the arrow legend. When suboption `by(`*byvar_ar*`)` is specified, suboption `legtitle(varlab)` requests that the label of variable *byvar_ar* be used as the legend title.

`leglabel(`*string*`)` specifies the label to be attached to the single key of the arrow legend when suboption `by(`*byvar_ar*`)` is not specified. This suboption is required when suboption `legenda(on)` is specified and suboption `by(`*byvar_ar*`)` is not specified.

`legshow(`*numlist*`)` requests that, when suboption `by(`*byvar_ar*`)` is specified, only the keys included in *numlist* be displayed in the arrow legend.

`legcount` requests that the number of arrows be displayed in the legend.

A.12 Option label() suboptions

Main

`data(`*label*`)` requests that one or more labels defined in Stata dataset *label* be superimposed onto the base map at given reference points.

`select(`*command*`)` requests that a given subset of records of dataset *label* be selected using Stata commands `keep` or `drop`.

`by(`*byvar_lb*`)` indicates that the labels defined in dataset *label* belong to klb different groups specified by variable *byvar_lb*.

`xcoord(`*xvar_lb*`)` specifies the name of the variable containing the x-coordinate of each label reference point.

`ycoord(`*yvar_lb*`)` specifies the name of the variable containing the y-coordinate of each label reference point.

`label(`*labvar_lb*`)` specifies the name of the variable containing the labels.

Format

`length(`*lengthstyle_list*`)` specifies the list of label lengths, where *lengthstyle* is any integer greater than 0 indicating the maximum number of characters of the labels. When suboption `by(`*byvar_lb*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_lb*`)` is specified, the list should be composed of klb elements. The default label lenght is `12`, the default specification is `length(12 ...)`.

`size(`*textsizestyle_list*`)` specifies the list of label sizes. When suboption `by(`*byvar_lb*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_lb*`)` is specified, the list should be composed of klb elements. The default label size is `*1`, the default specification is `size(*1 ...)`.

`color(`*colorlist*`)` specifies the list of label colors. When suboption `by(`*byvar_lb*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_lb*`)` is specified, the list should be either composed of klb elements, or represented by the name of a predefined color scheme. The default label color is `black`, the default specification is `color(black ...)`.

`position(`*clockpos_list*`)` specifies the list of label positions relative to their reference point. When suboption `by(`*byvar_lb*`)` is not specified, the list should include only one element. On the other hand, when suboption `by(`*byvar_lb*`)` is specified, the list should be composed of klb elements. The default label position is `0`, the default specification is `position(0 ...)`.

gap(*relativesize_list*) specifies the list of gaps between labels and their reference point. When suboption by(*byvar_lb*) is not specified, the list should include only one element. On the other hand, when suboption by(*byvar_lb*) is specified, the list should be composed of klb elements. The default label gap is *1, the default specification is gap(*1 ...).

angle(*anglestyle_list*) specifies the list of label angles. When suboption by(*byvar_lb*) is not specified, the list should include only one element. On the other hand, when suboption by(*byvar_lb*) is specified, the list should be composed of klb elements. The default label angle is horizontal, the default specification is angle(horizontal ...).

A.13 Option scalebar() suboptions

Main

units(#) specifies the length of the scale bar expressed in arbitrary units.

scale(#) specifies the ratio of scale bar units to map units. For example, suppose map coordinates are expressed in meters: if the scale bar length is to be expressed in meters too, then the ratio of scale bar units to map units will be 1; if, on the other hand, the scale bar length is to be expressed in kilometers, then the ratio of scale bar units to map units will be 1/1000. The default is scale(1).

xpos(#) specifies the distance of the scale bar from the center of the plot region on the horizontal axis, expressed as percentage of half the total width of the plot region. Positive values request that the distance be computed from the center to the right, whereas negative values request that the distance be computed from the center to the left. The default is xpos(0).

ypos(#) specifies the distance of the scale bar from the center of the plot region on the vertical axis, expressed as percentage of half the total height of the plot region. Positive values request that the distance be computed from the center to the top, whereas negative values request that the distance be computed from the center to the bottom. The default is ypos(-110).

Format

size(#) specifies a multiplier that affects the height of the scale bar. For example, size(1.5) requests that the default height of the scale bar be increased by 50%. The default is size(1).

fcolor(*colorstyle*) specifies the fill color of the scale bar. The default is fcolor(black).

ocolor(*colorstyle*) specifies the outline color of the scale bar. The default is ocolor(black).

osize(*linewidthstyle*) specifies the outline thickness of the scale bar. The default is osize(vthin).

label(*string*) specifies the descriptive label of the scale bar. The default is label(Units).

tcolor(*colorstyle*) specifies the color of the scale bar text. The default is tcolor(black).

tsize(*textsizestyle*) specifies the size of the scale bar text. The default is tsize(*1).

A.14 Graph options

Main

`polyfirst` requests that the supplementary polygons specified in option `polygon()` be drawn before the base map. By default, the base map is drawn before any other spatial object.

`gsize(#)` specifies the length (in inches) of the shortest side of the graph available area (the lenght of the longest side is set internally by `spmap` to minimize the amount of blank space around the map). The default ranges from 1 to 4, depending on the aspect ratio of the map. Alternatively, the height and width of the graph available area can be set using the standard `xsize()` and `ysize()` options.

`freestyle` requests that, when drawing the graph, all the formatting presets and restrictions built in `spmap` be ignored. By default, `spmap` presets the values of some graph options and restricts the use of some others, so as to produce a "nice" graph automatically. By specifying option `freestyle`, the user loses this feature but gains full control over most of the graph formatting options.

twoway_options include all the options documented in [**G**] *twoway_options*, except for *axis_options*, *aspect_option*, *scheme_option*, *by_option*, and *advanced_options*. These include *added_line_options*, *added_text_options*, *title_options*, *legend_options*, and *region_options*, as well as options `nodraw`, `name()`, and `saving()`.

```
spmap relig1 using "Italy-RegionsCoordinates.dta", id(id);
```

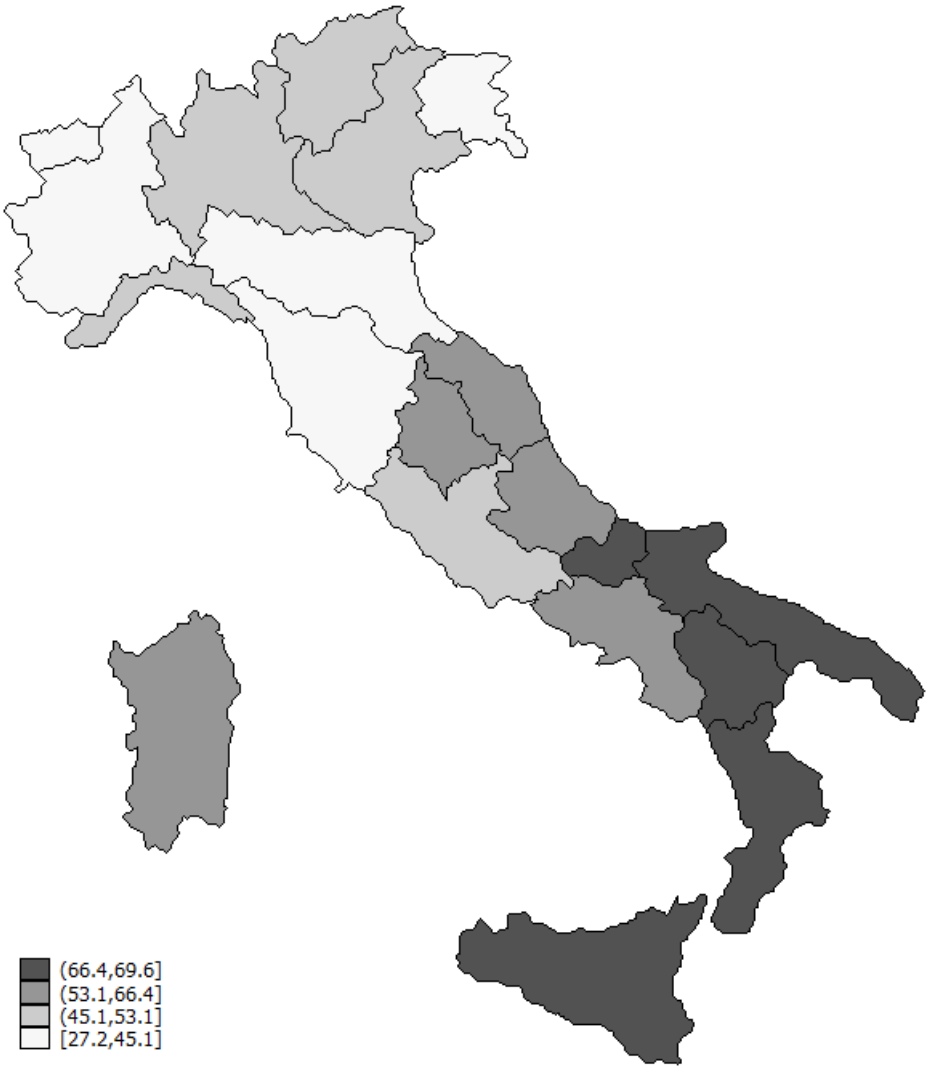


Figura A.1: Choropleth maps

```
spmap relig1 using "Italy-RegionsCoordinates.dta", id(id)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8));
```

Figura A.2: Choropleth maps

```
spmap relig1 using "Italy-RegionsCoordinates.dta", id(id)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8))
      legstyle(2) legend(region(lcolor(black)));
```

Figura A.3: Choropleth maps

```
spmap relig1m using "Italy-RegionsCoordinates.dta", id(id)
      ndfcolor(red)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8))
      legstyle(2) legend(region(lcolor(black)));
```

Figura A.4: Choropleth maps

```
spmap relig1 using "Italy-RegionsCoordinates.dta", id(id)
      clmethod(eqint) clnumber(5) eirange(20 70)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8))
      legstyle(2) legend(region(lcolor(black)));
```

Figura A.5: Choropleth maps

```
spmap relig1 using "Italy-RegionsCoordinates.dta", id(id)
      clnumber(20) fcolor(Reds2) ocolor(none ..)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8))
      legstyle(3);
```

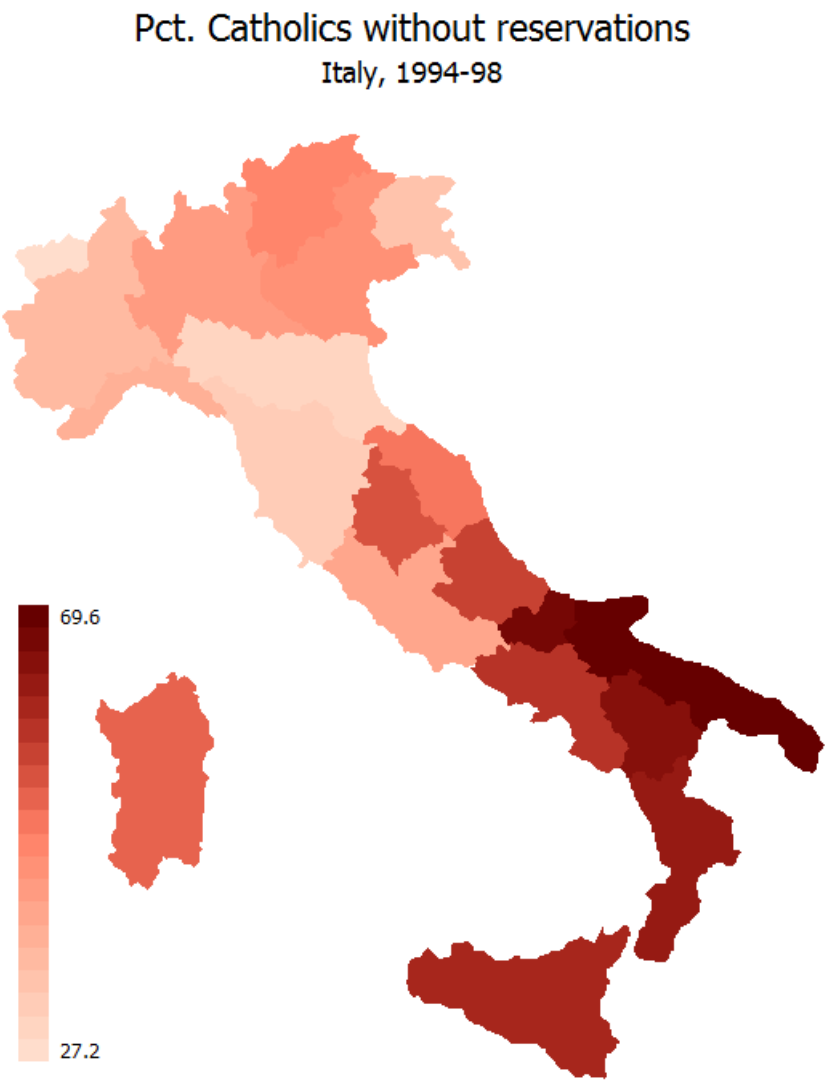


Figura A.6: Choropleth maps

```
spmap relig1 using "Italy-RegionsCoordinates.dta", id(id)
      clnumber(20) fcolor(Reds2) ocolor(none ..)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8))
      legstyle(3) legend(ring(1) position(3));
```

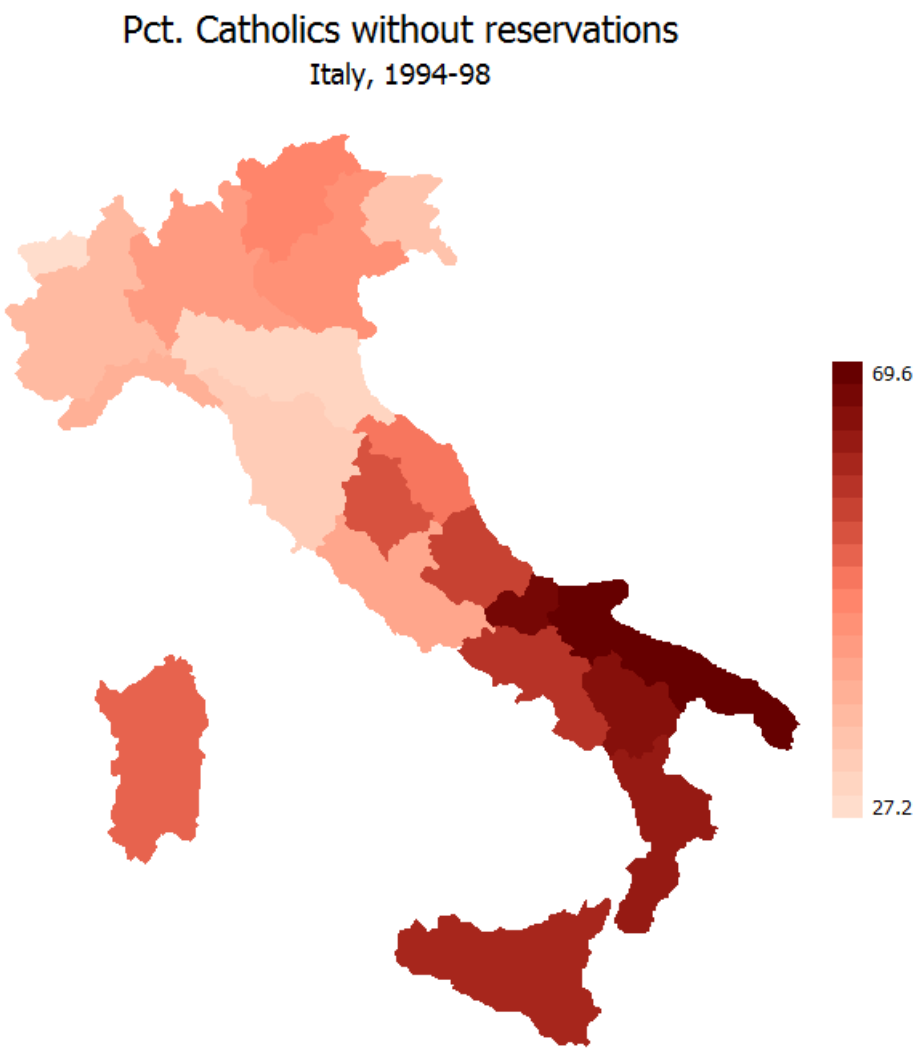


Figura A.7: Choropleth maps

```
spmap relig1 using "Italy-RegionsCoordinates.dta", id(id)
      clnumber(20) fcolor(Reds2) ocolor(none ..)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8))
      legstyle(3) legend(ring(1) position(3))
      plotregion(margin(vlarge));
```

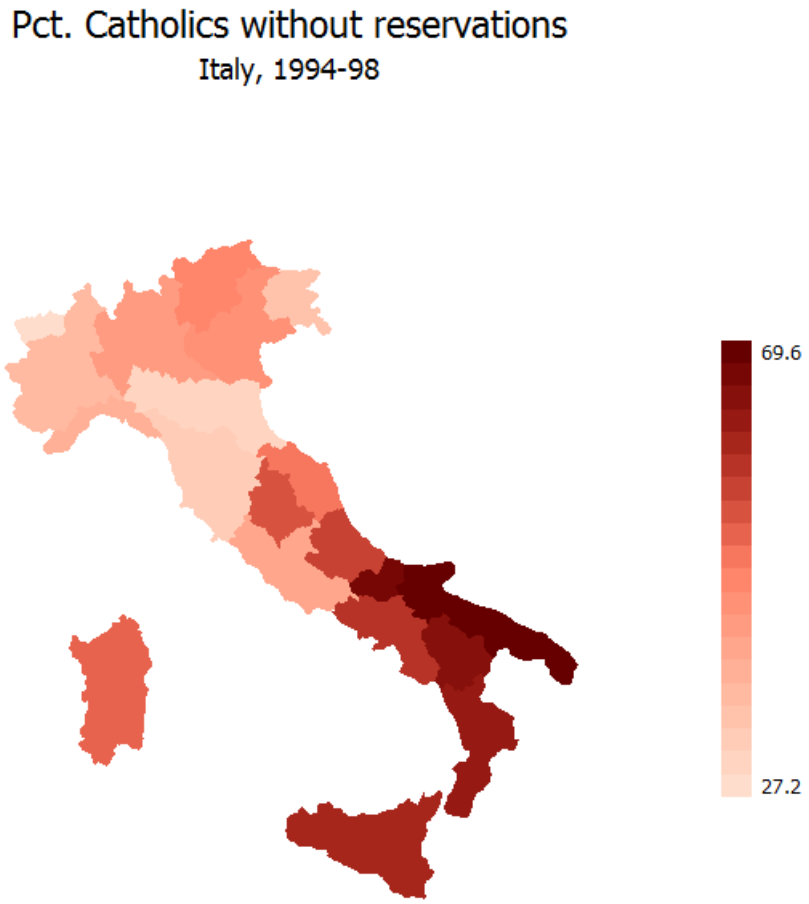


Figura A.8: Choropleth maps

```
spmap relig1 using "Italy-RegionsCoordinates.dta", id(id)
      clnumber(20) fcolor(Reds2) ocolor(none ..)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8))
      legstyle(3) legend(ring(1) position(3))
      plotregion(icolor(stone)) graphregion(icolor(stone));
```

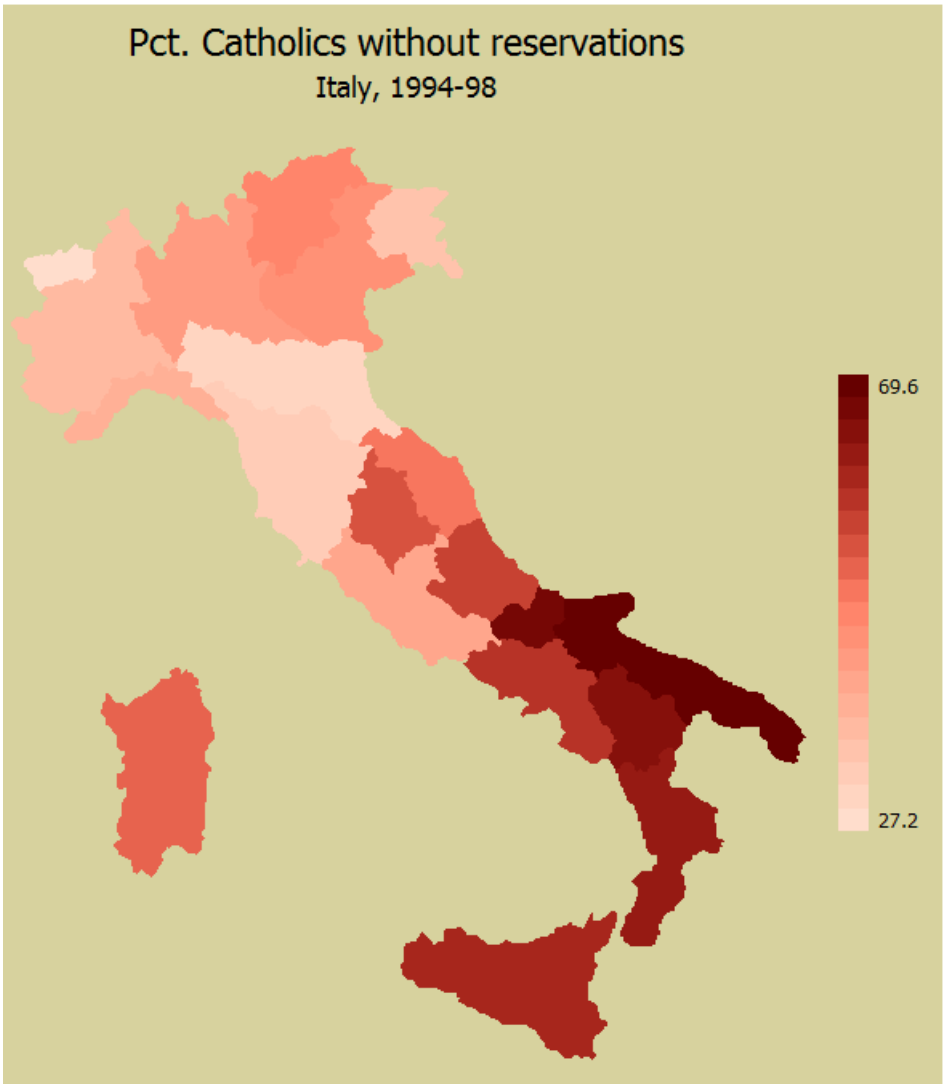


Figura A.9: Choropleth maps

```
spmap relig1 using "Italy-RegionsCoordinates.dta", id(id)
      clnumber(20) fcolor(Greens2) ocolor(white ..) osize(medthin ..)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8))
      legstyle(3) legend(ring(1) position(3))
      plotregion(icolor(stone)) graphregion(icolor(stone));
```

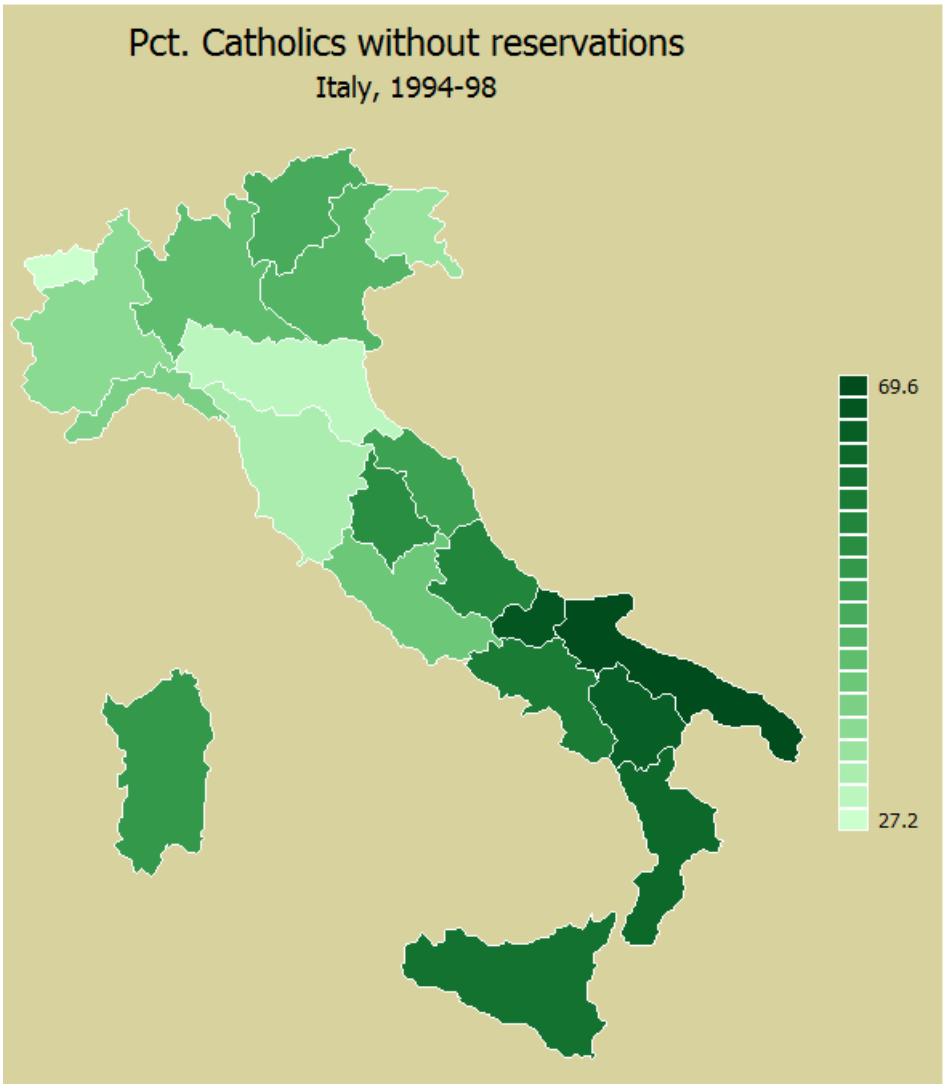


Figura A.10: Choropleth maps

```
spmap relig1 using "Italy-RegionsCoordinates.dta", id(id)
      clnumber(20) fcolor(Greens2) ocolor(white ..) osize(thin ..)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8))
      legstyle(3) legend(ring(1) position(3))
      plotregion(icolor(stone)) graphregion(icolor(stone))
      polygon(data("Italy-Highlights.dta") ocolor(white)
      osize(medthick));
```

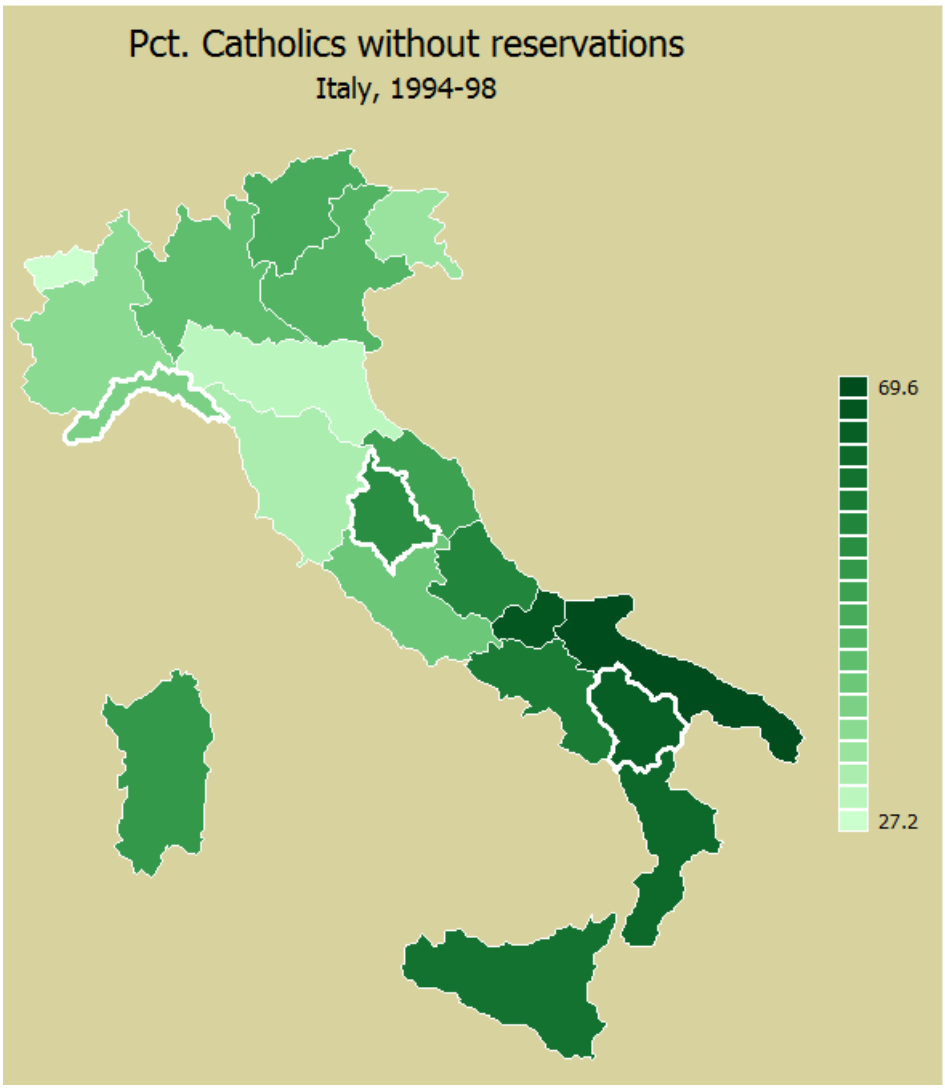


Figura A.11: Choropleth maps

```
spmap relig1 using "Italy-RegionsCoordinates.dta", id(id)
      clnumber(20) fcolor(Greens2) ocolor(white ..) osize(medthin ..)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8))
      legstyle(3) legend(ring(1) position(3))
      plotregion(icolor(stone)) graphregion(icolor(stone))
      scalebar(units(500) scale(1/1000) xpos(-100) label(Kilometers));
```

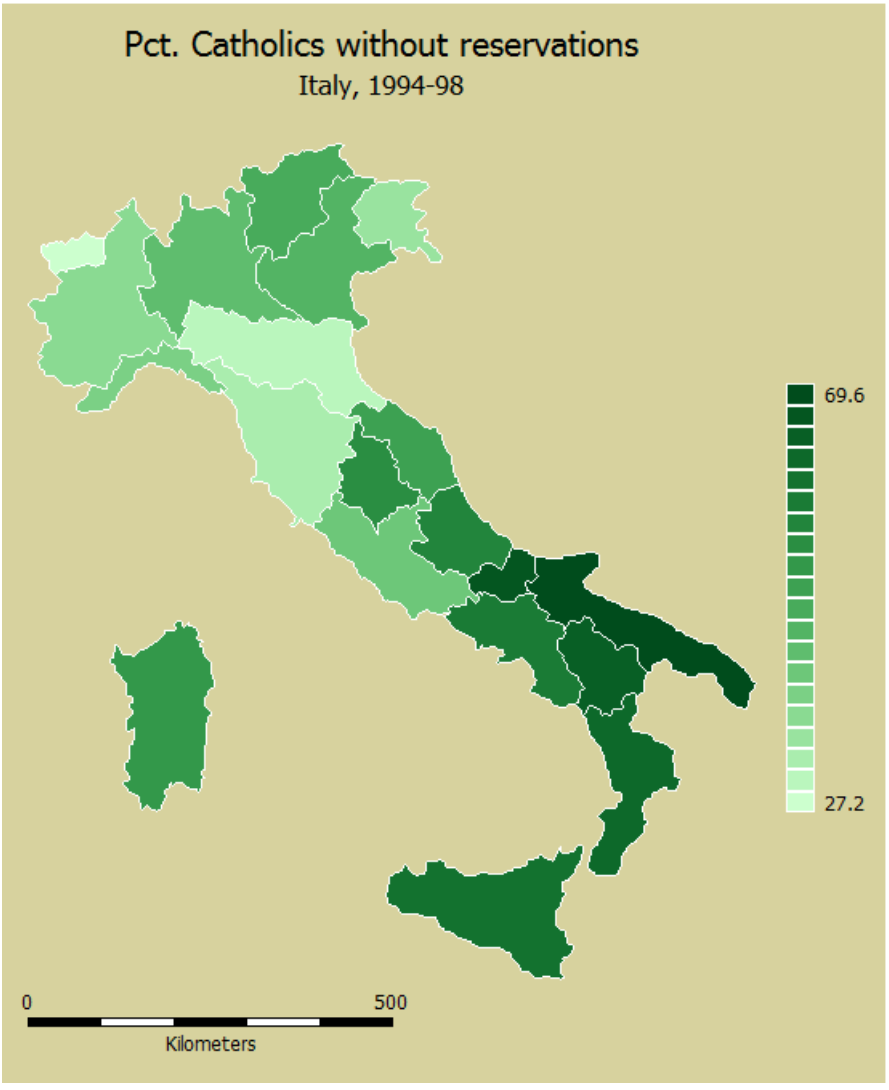


Figura A.12: Choropleth maps

```
spmap using "Italy-OutlineCoordinates.dta", id(id)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8))
      point(data("Italy-RegionsData.dta") xcoord(xcoord)
      ycoord(ycoord) proportional(relig1) fcolor(red) size(*1.5));
```

Figura A.13: Proportional symbol maps

```
spmap using "Italy-OutlineCoordinates.dta", id(id)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8))
      point(data("Italy-RegionsData.dta") xcoord(xcoord)
      ycoord(ycoord) proportional(relig1) fcolor(red) size(*1.5)
      shape(s));
```

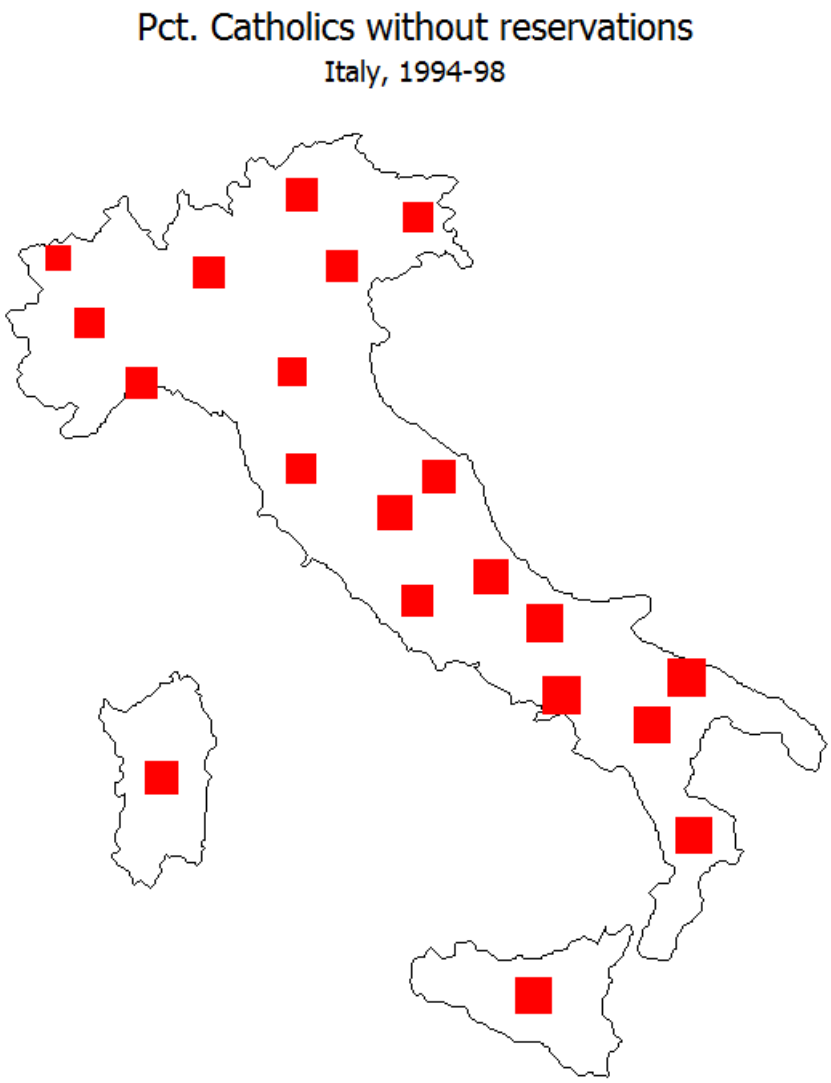


Figura A.14: Proportional symbol maps

```
spmap using "Italy-OutlineCoordinates.dta", id(id)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8))
      point(data("Italy-RegionsData.dta") xcoord(xcoord)
      ycoord(ycoord) proportional(relig1) fcolor(red)
      ocolor(white) size(*3))
      label(data("Italy-RegionsData.dta") xcoord(xcoord)
      ycoord(ycoord) label(relig1) color(white) size(*0.7));
```

Figura A.15: Proportional symbol maps

```
spmap using "Italy-OutlineCoordinates.dta", id(id)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8))
      point(data("Italy-RegionsData.dta") xcoord(xcoord)
      ycoord(ycoord) deviation(relig1) fcolor(red) dmax(30)
      legenda(on) leglabel(Deviation from the mean));
```

Figura A.16: Proportional symbol maps

```
spmap using "Italy-OutlineCoordinates.dta", id(id) fcolor(white)
      title("Catholics without reservations", size(*0.9) box bexpand
      span margin(medsmall) fcolor(sand)) subtitle(" ")
      point(data("Italy-RegionsData.dta") xcoord(xcoord)
      ycoord(ycoord) proportional(relig1) prange(0 70)
      psize(absolute) fcolor(red) ocolor(white) size(*0.6))
      plotregion(margin(medium) color(stone))
      graphregion(fcolor(stone) lcolor(black))
      name(g1, replace) nodraw;
spmap using "Italy-OutlineCoordinates.dta", id(id) fcolor(white)
      title("Catholics with reservations", size(*0.9) box bexpand
      span margin(medsmall) fcolor(sand)) subtitle(" ")
      point(data("Italy-RegionsData.dta") xcoord(xcoord)
      ycoord(ycoord) proportional(relig2) prange(0 70)
      psize(absolute) fcolor(green) ocolor(white) size(*0.6))
      plotregion(margin(medium) color(stone))
      graphregion(fcolor(stone) lcolor(black))
      name(g2, replace) nodraw;
spmap using "Italy-OutlineCoordinates.dta", id(id) fcolor(white)
      title("Other", size(*0.9) box bexpand
      span margin(medsmall) fcolor(sand)) subtitle(" ")
      point(data("Italy-RegionsData.dta") xcoord(xcoord)
      ycoord(ycoord) proportional(relig3) prange(0 70)
      psize(absolute) fcolor(blue) ocolor(white) size(*0.6))
      plotregion(margin(medium) color(stone))
      graphregion(fcolor(stone) lcolor(black))
      name(g3, replace) nodraw;
graph combine g1 g2 g3, rows(1) title("Religious orientation")
      subtitle("Italy, 1994-98" " ") xsize(5) ysize(2.6)
      plotregion(margin(medsmall) style(none))
      graphregion(margin(zero) style(none))
      scheme(s1mono);
```

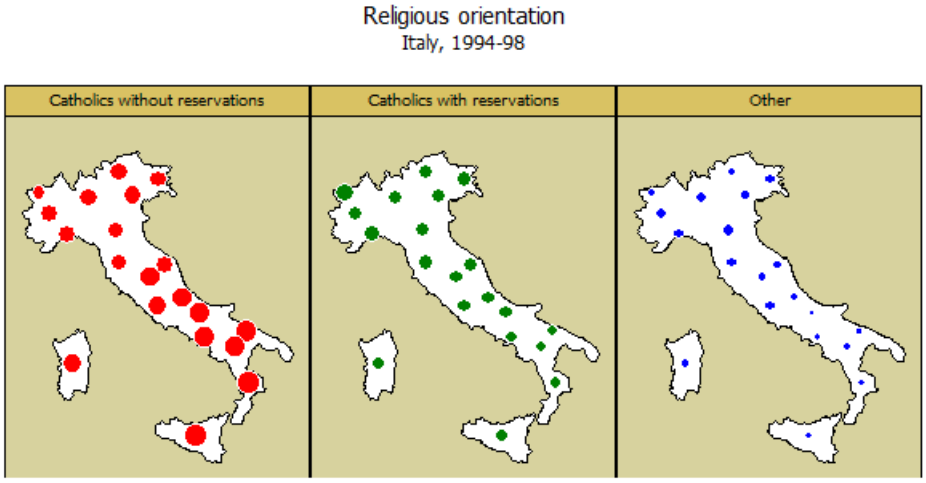


Figura A.17: Proportional symbol maps

```
spmap using "Italy-RegionsCoordinates.dta", id(id) fcolor(stone)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8))
      diagram(variable(relig1) range(0 100) refweight(pop98)
      xcoord(xcoord) ycoord(ycoord) fcolor(red));
```

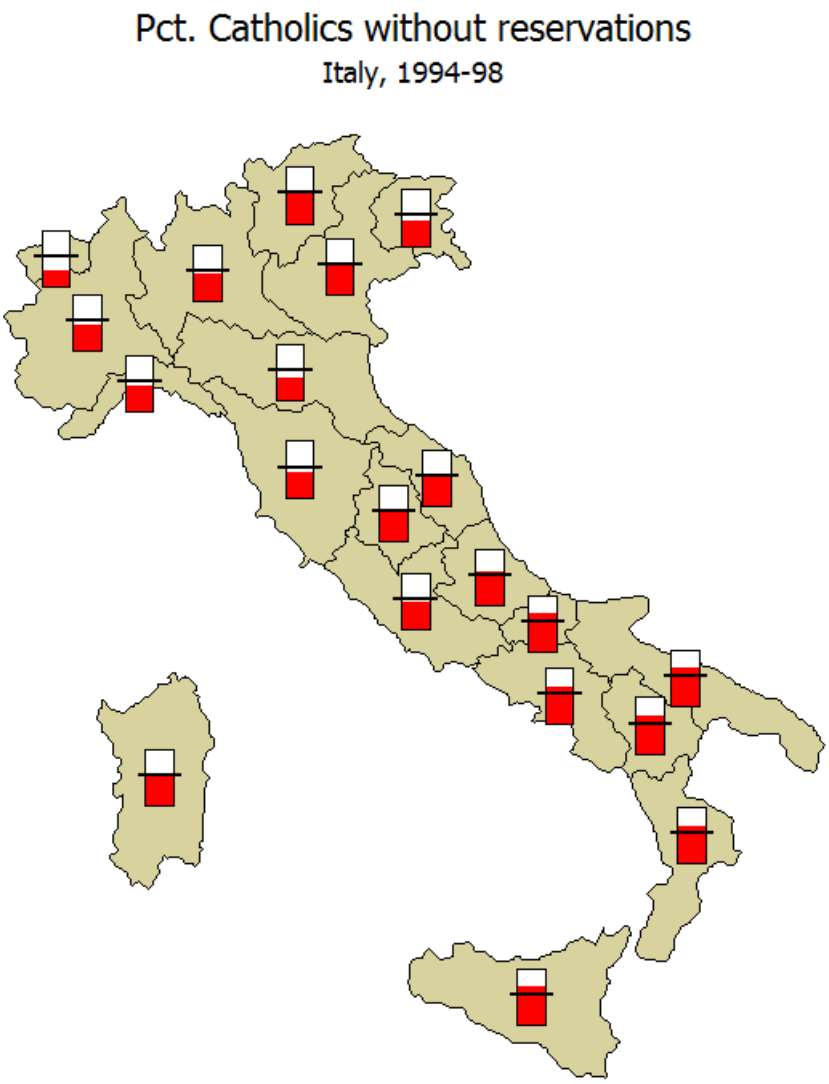


Figura A.18: Other maps

```
spmap using "Italy-RegionsCoordinates.dta", id(id) fcolor(stone)
      diagram(variable(relig1 relig2 relig3) proportional(fortell)
      xcoord(xcoord) ycoord(ycoord) legenda(on))
      legend(title("Religious orientation", size(*0.5) bexpand
      justification(left)))
      note(" "
      "NOTE: Chart size proportional to number of fortune tellers
      per million population",
    size(*0.75));
```

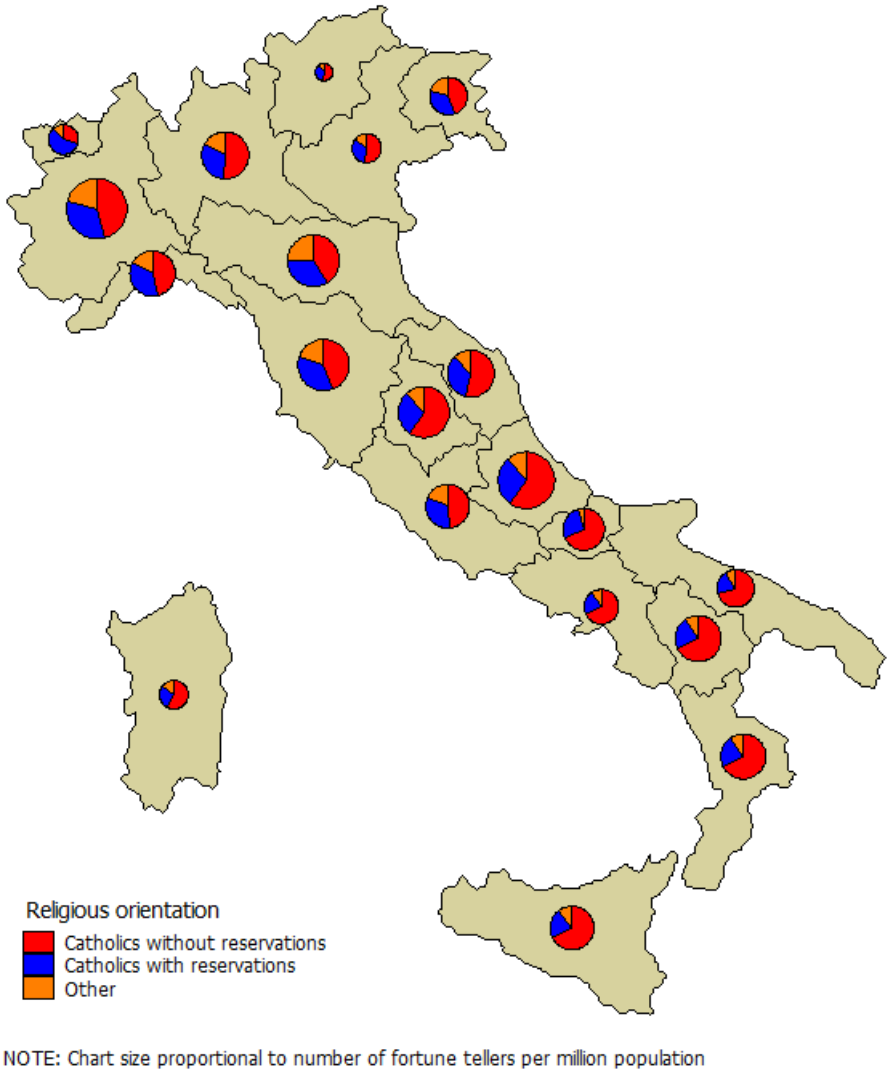


Figura A.19: Other maps

```
spmap relig1 using "Italy-RegionsCoordinates.dta", id(id)
      clmethod(stdev) clnumber(5)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8)) area(pop98)
      note(" "
      "NOTE: Region size proportional to population", size(*0.75));
```

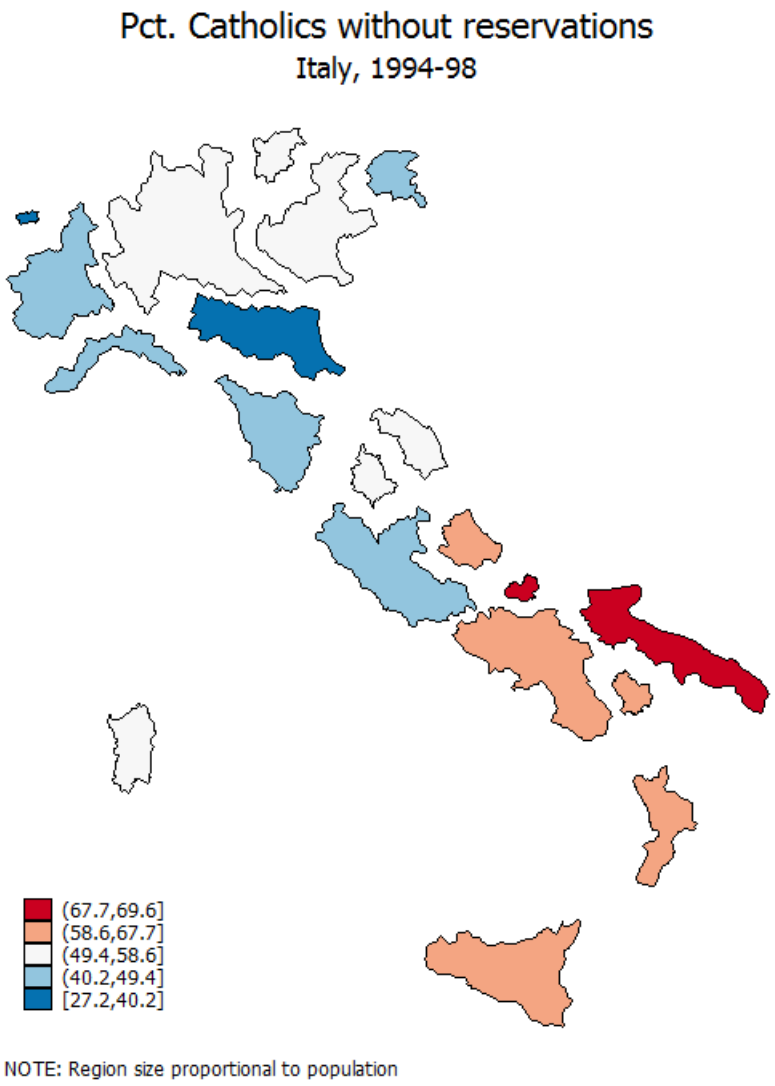


Figura A.20: Other maps

```
spmap relig1 using "Italy-RegionsCoordinates.dta", id(id)
      clmethod(stdev) clnumber(5)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Italy, 1994-98" " ", size(*0.8)) area(pop98)
      map("Italy-OutlineCoordinates.dta") mfcolor(stone)
      note(" "
      "NOTE: Region size proportional to population", size(*0.75));
```

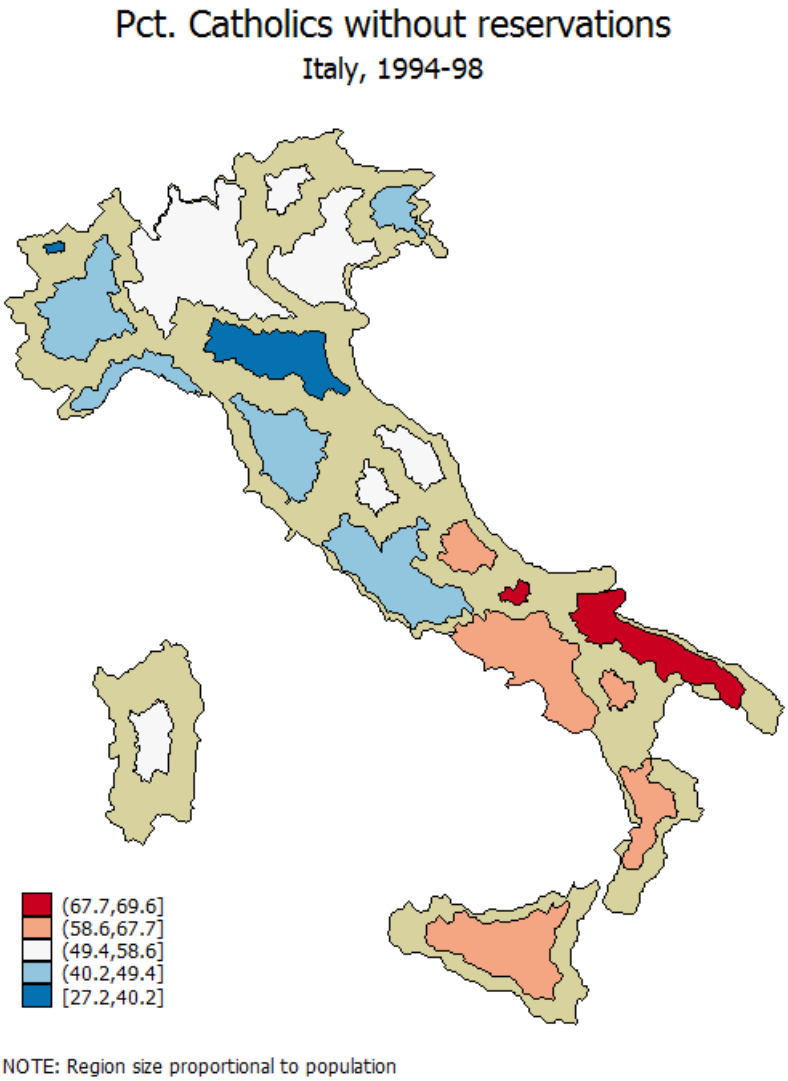


Figura A.21: Other maps

```
spmap using "Italy-OutlineCoordinates.dta", id(id) fc(bluishgray)
      ocolor(none)
      title("Provincial capitals" " ", size(*0.9) color(white))
      point(data("Italy-Capitals.dta") xcoord(xcoord)
      ycoord(ycoord) fcolor(emerald))
      plotregion(margin(medium) icolor(dknavy) color(dknavy))
      graphregion(icolor(dknavy) color(dknavy));
```

Figura A.22: Other maps

```
spmap using "Italy-OutlineCoordinates.dta", id(id) fc(bluishgray)
      ocolor(none)
      title("Provincial capitals" " ", size(*0.9) color(white))
      point(data("Italy-Capitals.dta") xcoord(xcoord)
      ycoord(ycoord) by(size) fcolor(orange red maroon) shape(s ..)
      legenda(on))
      legend(title("Population 1998", size(*0.5) bexpand
      justification(left)) region(lcolor(black) fcolor(white))
      position(2))
      plotregion(margin(medium) icolor(dknavy) color(dknavy))
      graphregion(icolor(dknavy) color(dknavy));
```

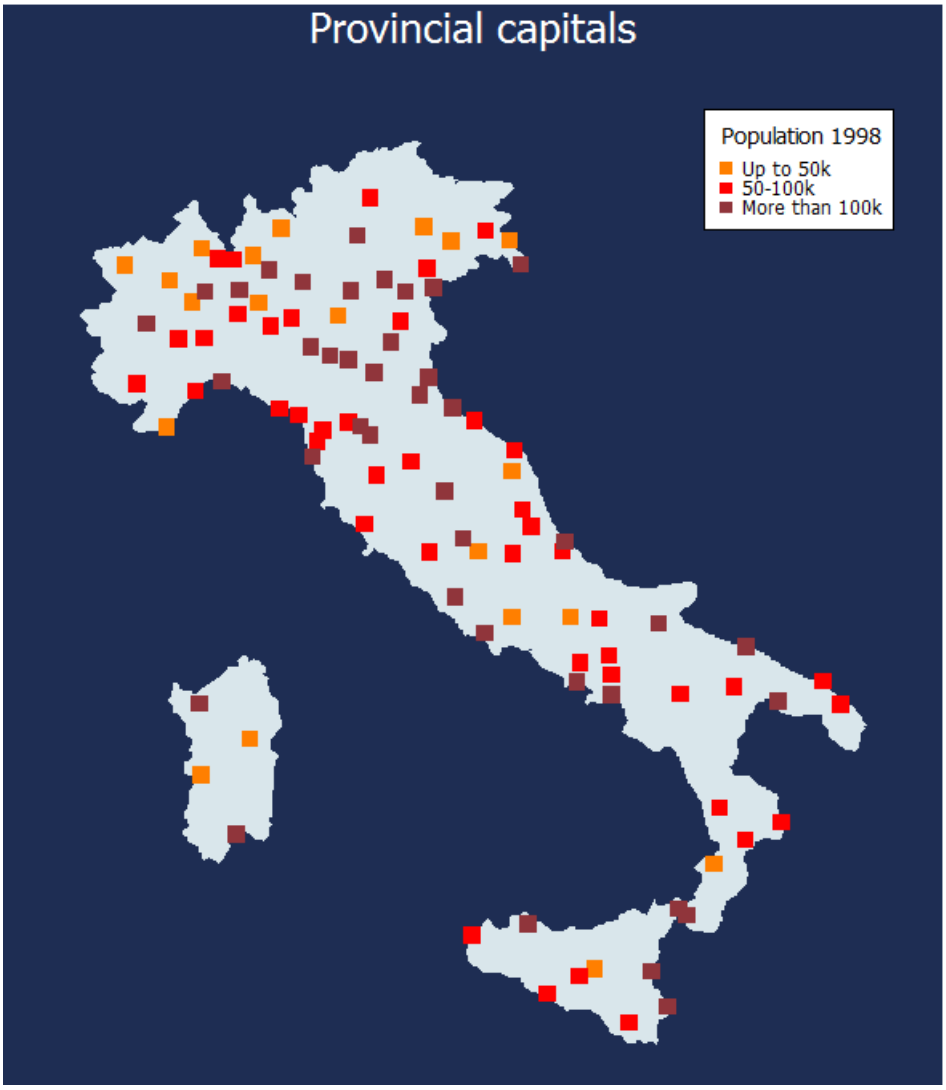


Figura A.23: Other maps

```
spmap using "Italy-OutlineCoordinates.dta", id(id) fc(sand)
      title("Main lakes and rivers" " ", size(*0.9))
      polygon(data("Italy-Lakes.dta") fcolor(blue) ocolor(blue))
      line(data("Italy-Rivers.dta") color(blue) );
```

Figura A.24: Other maps

```
use "Italy-RegionsData.dta", clear;
spmap relig1 using "Italy-RegionsCoordinates.dta" if zone==1,
      id(id) fcolor(Blues2) ocolor(white ..) osize(medthin ..)
      title("Pct. Catholics without reservations", size(*0.8))
      subtitle("Northern Italy, 1994-98" " ", size(*0.8))
      polygon(data("Italy-OutlineCoordinates.dta") fcolor(gs12)
      ocolor(white) osize(medthin)) polyfirst;
```

Figura A.25: Other maps

```
use "Italy-OutlineData.dta", clear;
spmap using "Italy-OutlineCoordinates.dta", id(id) fc(sand)
        title("Main lakes and rivers" " ", size(*0.9))
        polygon(data("Italy-Lakes.dta") fcolor(blue) ocolor(blue))
        line(data("Italy-Rivers.dta") color(blue) )
        freestyle aspect(1.4) xlab(400000 900000 1400000, grid);
```

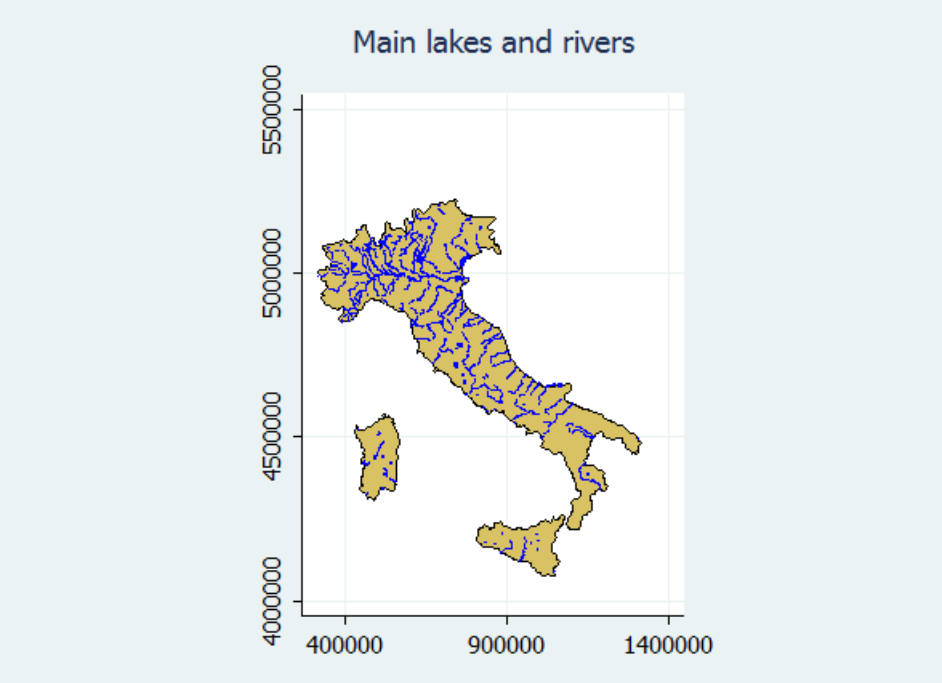


Figura A.26: Other maps

A.15 Acknowledgments

I wish to thank Nick Cox, Ian Evans, and Vince Wiggins for helping set up `tmap` (Pisati 2004), the predecessor of `spmap`. I also thank Kevin Crow, Bill Gould, Friedrich Huebler, and Scott Merryman for promoting tmap by making available to the Stata community several helpful resources related to the program. The development of `spmap` benefitted from suggestions by Joao Pedro Azevedo, Kit Baum, Daniele Checchi, Kevin Crow, David Drukker, Friedrich Huebler, Laszlo Kardos, Ulrich Kohler, Scott Merryman, Derek Wagner, the participants in the 1st Italian Stata Users Group Meeting, and the participants in the 3rd German Stata Users Group Meeting: many thanks to all of them.

Bibliografia

[1] Armstrong, M.P., Xiao, N. and D.A. Bennett. 2003. Using genetic algorithms to create multicriteria class intervals for choropleth maps. Annals of the Association of American Geographers 93: 595-623.

[2] Brewer, C.A. 1999. Color use guidelines for data representation. Proceedings of the Section on Statistical Graphics, American Statistical Association. Alexandria VA, 55-60.

[3] Brewer, C.A., Hatchard, G.W. and M.A. Harrower. 2003. ColorBrewer in print: A catalog of color schemes for maps. Cartography and Geographic Information Science 52: 5-32.

[4] Cleveland, W.S. 1994. The Elements of Graphing Data. Summit: Hobart Press.

[5] Cleveland, W.S. and R. McGill. 1984. Graphical perception: Theory, experimentation, and application to the development of graphical methods. Journal of the American Statistical Association 79: 531-554.

[6] Evans, I.S. 1977. The selection of class intervals. Transactions of the Institute of British Geographers 2: 98-124.

[7] Olson, J.M. 1976. Noncontiguous area cartograms. The Professional Geographer 28: 371-380.

[8] Pisati, M. 2004. Simple thematic mapping. The Stata Journal 4: 361-378.

[9] Slocum, T.A., McMaster, R.B., Kessler, F.C and H.H. Howard. 2005. Thematic Cartography and Geographic Visualization. 2nd ed. Upper Saddle River: Pearson Prentice Hall.

Appendice B

Lista pacchetti aggiuntivi

```
. ssc whatshot, n(.)

Packages at SSC

        Jan 2015
  Rank   # hits    Package       Author(s)
  ------------------------------------------------------------------------
     1  21197.8    qcount        Alfonso Miranda
     2  14291.2    outreg2       Roy Wada
     3  12605.6    estout        Ben Jann
     4   3739.6    psmatch2      Barbara Sianesi, Edwin Leuven
     5   3281.5    outreg        John Luke Gallup
     6   3183.0    winsor        Nicholas J. Cox
     7   2757.6    ivreg2        Steven Stillman, Mark E Schaffer,
                                   Christopher F Baum
     8   2531.0    use13         Sergiy Radyakin
     9   2371.0    regsave       Julian Reif
    10   2342.3    egenmore      Nicholas J. Cox
    11   2327.3    tabout        Ian Watson
    12   2246.7    fre           Ben Jann
    13   2028.7    ranktest      Frank Kleibergen, Mark E Schaffer
    14   2017.7    strgroup      Julian Reif
    15   1742.3    xttest3       Christopher F Baum
    16   1703.3    carryforward  David Kantor
    17   1646.7    spmap         Maurizio Pisati
    18   1642.0    shp2dta       Kevin Crow
    19   1620.7    xtabond2      David Roodman
    20   1610.0    unique        Tony Brady
    21   1503.3    bcuse         Christopher F Baum
    22   1378.7    xtivreg2      Mark E Schaffer
    23   1369.3    tscollap      Christopher F Baum
    24   1336.7    survwgt       Nick Winter
    25   1208.7    distinct      Nicholas J. Cox, Gary Longton
    26   1198.8    parmest       Roger Newson
    27   1181.7    tableplot     Nicholas J. Cox
    28   1117.8    catplot       Nicholas J. Cox
    29   1059.0    _gwtmean      David Kantor
    30   1039.7    mdesc         Rose Anne Medeiros, Dan Blanchette
    31    953.7    diff          Juan M. Villa
    32    947.0    coefplot      Ben Jann
    33    925.6    metan         Jon Deeks, Thomas Steichen, Ross
                                   Harris, Roger Harbord, Jonathan Sterne,
                                   Doug Altman, Mike Bradburn
    34    887.7    whitetst      Christopher F Baum, Nicholas J. Cox
```

35	872.3	fitstat	J. Scott Long, Jeremy Freese
36	846.8	logout	Roy Wada
37	797.2	rd	Austin Nichols
38	785.7	xml_tab	Michael Lokshin, Zurab Sajaia
39	778.7	mif2dta	Maurizio Pisati
40	759.0	xttest2	Christopher F Baum
41	745.3	xtivreg28	Mark E Schaffer
42	729.7	xtcsd	R. E. De Hoyos, Vasilis Sarafidis
43	701.5	ineqdeco	Stephen P. Jenkins
44	687.3	oaxaca	Ben Jann
45	662.8	outtex	Antoine Terracol
46	662.7	xtoverid	Steven Stillman, Mark E Schaffer
47	642.2	gllamm	Sophia Rabe-Hesketh
48	637.0	winsor2	Lian Yu-jun
49	616.7	overid	Mark E Schaffer, Christopher F Baum, Vince Wiggins, Steven Stillman
50	594.7	usespss	Sergiy Radyakin
51	581.3	egranger	Mark E Schaffer
52	566.4	wbopendata	Joao Pedro Azevedo
53	563.3	reclink	Michael Blasnik
54	548.7	synth	Jens Hainmueller, Alberto Abadie, Alexis Diamond
55	535.0	sutex	Antoine Terracol
56	503.1	grqreg	Joao Pedro Azevedo
57	498.0	ginidesc	Roger Aliaga, Silvia Montoya
58	497.7	nsplit	Dan Blanchette
59	486.0	xtscc	Daniel Hoechle
60	466.7	kpss	Christopher F Baum
61	452.7	strip	P.T.Seed
62	447.7	confirmdir	Dan Blanchette
63	429.0	quantiles	Rafael Guerreiro Osorio
64	420.3	glcurve	Stephen P. Jenkins, Philippe Van Kerm
65	419.3	outtable	Christopher F Baum, Joao Pedro Azevedo
66	419.2	fsum	Fred Wolfe
67	412.5	latab	Ian Watson
68	411.3	extremes	Nicholas J. Cox
69	409.0	mmerge	Jeroen Weesie
70	408.3	ainequal	Joao Pedro Azevedo
71	404.3	dummieslab	Nicholas J. Cox, Philippe Van Kerm
72	395.7	mfx2	Richard Williams
73	393.7	corrtex	Nicolas Couderc
74	384.7	nnmatch	Guido W. Imbens, Jane Leber Herr, Alberto Abadie, David M. Drukker
75	379.3	dmariano	Christopher F Baum
76	376.0	binscatter	Michael Stepner
77	366.3	ivendog	Steven Stillman, Mark E Schaffer, Christopher F Baum
78	366.3	tsspell	Nicholas J. Cox
79	366.0	scat3	Nicholas J. Cox
80	363.3	apoverty	Joao Pedro Azevedo
81	356.7	center	Ben Jann
82	348.2	tmpdir	Dan Blanchette
83	346.7	svmatf	Jan Brogger
84	344.6	labutil	Nicholas J. Cox
85	338.8	freduse	David M. Drukker
86	338.5	hprescott	Christopher F Baum
87	333.7	jb	Gregorio Impavido, J. Sky David
88	333.5	somersd	Roger Newson
89	333.0	csipolate	Nicholas J. Cox
90	329.3	bpagan	Vince Wiggins, Christopher F Baum
91	328.3	inequal7	Philippe Van Kerm
92	325.3	lambda	Nicholas J. Cox
93	325.3	mat2txt	Michael Blasnik, Ben Jann
94	323.9	tab_chi	Nicholas J. Cox
95	323.3	mkcorr	Glenn Hoetker

96	319.0	sencode	Roger Newson
97	318.7	xtfmb	Daniel Hoechle
98	315.0	labellist	Daniel Klein
99	307.0	libjson	Erik Lindsley
100	307.0	xtwest	Damiaan Persyn
101	305.7	batplot	Adrian Mander
102	301.3	armadiag	Sune Karlsson
103	300.8	sppack	Hua Peng, Ingmar Prucha, Rafal Raciborski, David M. Drukker
104	299.7	metaninf	Thomas Steichen
105	299.7	shortdir	Dan Blanchette
106	297.3	metabias	Ross J Harris, Thomas Steichen, Robert M Harbord, Jonathan AC Sterne
107	292.3	xsmle	Gordon Hughes, Andrea Piano Mortari, Federico Belotti
108	292.0	insheetjson	Erik Lindsley
109	289.0	xtbalance	Lian Yujun
110	285.7	omodel	Rory Wolfe
111	283.3	weakiv	Keith Finlay, Leandro Magnusson, Mark E Schaffer
112	281.0	cmp	David Roodman
113	274.4	gologit2	Richard Williams
114	274.3	chewfile	Roy Wada
115	272.3	nmissing	Nicholas J. Cox
116	270.5	usesas	Dan Blanchette
117	270.3	cmogram	Christopher Robert
118	270.3	revrs	Kyle C. Longest
119	270.3	sxpose	Nicholas J. Cox
120	267.0	hoi	Joao Pedro Azevedo, Samuel Franco, Eliana Rubiano, Alejandro Hoyos
121	263.7	tab3way	Philip Ryan
122	258.3	metafunnel	Jonathan Sterne
123	257.7	glst	Rino Bellocco, Nicola Orsini, Sander Greenland
124	256.0	geodist	Robert Picard
125	255.5	savasas	Dan Blanchette
126	250.3	mkprofile	Dan Blanchette
127	248.9	mixlogit	Arne Risa Hole
128	247.2	metareg	Julian Higgins, Roger Harbord
129	242.3	fastgini	Zurab Sajaia
130	242.3	gcause	Patrick Joly
131	242.3	zscore06	Jef Leroy
132	237.3	tabstatmat	Austin Nichols
133	235.8	fs	Nicholas J. Cox
134	234.5	kountry	Rafal Raciborski
135	234.3	charlson	Vicki Stagg
136	231.8	saswrapper	Dan Blanchette
137	229.7	geocode3	Stefan Bernhard
138	229.7	povdeco	Stephen P. Jenkins
139	227.5	margeff	Tamas Bartus
140	227.0	matsave	Marc-Andreas Muendler
141	223.0	abar	David Roodman
142	221.3	levinlin	Christopher F Baum, Fabian Bornhorst
143	221.3	tuples	Nicholas J. Cox, Joseph N. Luchman
144	219.7	reg2hdfe	Paulo Guimaraes
145	216.3	ttesttable	Florian Chavez Juarez
146	215.0	xtfisher	Scott Merryman
147	214.0	cem	Giuseppe Porro, Gary King, Stefano Iacus, Matthew Blackwell
148	214.0	sumdist	Stephen P. Jenkins
149	212.0	varprod	Emad Abd Elmessih Shehata
150	210.0	tolower	Nicholas J. Cox
151	209.5	runmlwin	Chris Charlton, George Leckie
152	209.4	reghdfe	Sergio Correia
153	208.8	kdens	Ben Jann

154	207.2	a2reg	Amine Ouazad
155	203.2	onespell	Christopher F Baum
156	202.7	chowreg	Emad Abd Elmessih Shehata
157	197.0	spwmatrix	P. Wilner Jeanty
158	196.3	sutex2	Francesco Scervini
159	194.3	ice	Patrick Royston
160	190.0	ebalance	Jens Hainmueller, Yiqing Xu
161	190.0	qreg2	J.M.C. Santos Silva, P.M.D.C Parente, J.A.F. Machado
162	189.8	ivreg29	Steven Stillman, Mark E Schaffer, Christopher F Baum
163	189.3	hte	Jennie E. Brand, Ben Jann, Yu Xie
164	189.3	ivhettest	Mark E Schaffer
165	189.3	xtsur	Minh Cong Nguyen
166	186.7	mmodes	Adrian Mander
167	185.7	xtpedroni	Timothy Neal
168	184.0	xtgraph	Paul Seed
169	181.8	svr	Nick Winter
170	180.7	appendfile	Julian Reif
171	180.3	stcascoh	Enzo Coviello
172	178.7	pescadf	Piotr Lewandowski
173	176.7	zandrews	Christopher F Baum
174	176.3	regwls	Dany Bahar
175	176.0	stcompet	Enzo Coviello
176	175.7	multipurt	Markus Eberhardt
177	175.7	xtmg	Markus Eberhardt
178	174.0	coldiag2	John Hendrickx
179	174.0	groups	Nicholas J. Cox
180	173.0	hhi	Muhammad Rashid Ansari
181	173.0	usepackage	Eric Booth
182	171.7	tsmktim	Vince Wiggins, Christopher F Baum
183	171.2	todate	Nicholas J. Cox
184	170.0	ftest	Maarten L. Buis
185	169.8	savespss	Sergiy Radyakin
186	168.7	tpm	Federico Belotti, Partha Deb
187	167.7	blp	David Vincent
188	167.7	rbounds	Markus Gangl
189	167.3	ciplot	Nicholas J. Cox
190	167.3	ttable2	Xuan Zhang, Chuntao Li
191	167.2	xtpmg	Edward F. Blackburne III, Mark W. Frank
192	167.0	geonear	Robert Picard
193	166.3	factortest	Joao Pedro Azevedo
194	166.3	texsave	Julian Reif
195	165.3	use13save12	Lars Angquist
196	163.5	newey2	David Roodman
197	163.0	traveltime3	Stefan Bernhard
198	162.7	ipshin	Fabian Bornhorst, Christopher F Baum
199	162.3	clemao_io	Christopher F Baum
200	160.7	cdfplot	Adrian Mander
201	160.2	midas	Ben Dwamena
202	158.8	ineqdec0	Stephen P. Jenkins
203	157.2	oparallel	Maarten L. Buis
204	156.8	adoedit	Dan Blanchette
205	156.0	vincenty	Austin Nichols
206	155.8	mvsumm	Christopher F Baum, Nicholas J. Cox
207	154.0	combomarginsplot	Nick Winter
208	152.8	ghk2	David Roodman
209	152.7	nearmrg	Eric Booth
210	152.0	msp	Jean-Benoit Hardouin
211	150.7	eclplot	Roger Newson
212	150.5	diagt	Paul Seed
213	150.3	ivreg28	Steven Stillman, Mark E Schaffer, Christopher F Baum
214	149.7	xtcd	Markus Eberhardt
215	149.3	sfpanel	Vincenzo Atella, Giuseppe Ilardi,

```
                                Silvio Daidone, Federico Belotti
216    148.0    bygap          Roger Newson
217    148.0    rcsgen         Paul Lambert
218    148.0    strdist        Michael Barker
219    147.3    ascii          Adrian Mander
220    147.3    elapse         Fred Zimmerman
221    145.9    sq             Magdalena Luniak, Christian
                                Brzinsky-Fay, Ulrich Kohler
222    145.0    cusum6         Christopher F Baum
223    144.3    kwallis2       Herve M. Caci
224    144.0    stripplot      Nicholas J. Cox
225    142.7    aaplot         Nicholas J. Cox
226    140.0    r2reg3         Emad Abd Elmessih Shehata
227    138.3    r2sem          Emad Abd Elmessih Shehata
228    138.2    r2nlsur        Emad Abd Elmessih Shehata
229    137.7    egen_inequal   Zurab Sajaia, Michael Lokshin
230    137.4    splagvar       P. Wilner Jeanty
231    137.3    oglm           Richard Williams
232    137.3    todummy        Daniel Klein
233    137.0    pv             Kevin Macdonald
234    134.7    est2tex        Marc-Andreas Muendler
235    134.3    savesome       Nicholas J. Cox
236    134.3    xtdolshm       Diallo Ibrahima Amadou
237    133.7    xtpqml         Tim Simcoe
238    133.3    avar           Christopher F Baum, Mark E Schaffer
239    131.1    gammafit       Nicholas J. Cox, Stephen P. Jenkins
240    130.9    xtlsdvc        Giovanni S.F. Bruno
241    130.3    jb6            Gregorio Impavido, J. Sky David
242    130.3    xtmrho         Lars E. Kroll
243    129.0    rollreg        Christopher F Baum
244    129.0    weakivtest     Su Wang, Carolin E. Pflueger, Jose Luis
                                Montiel Olea
245    128.2    mrtab          Ben Jann, Hilde Schaeper
246    127.7    nnest          Gregorio Impavido
247    126.3    graphexportpdf  Gabriel Rossman
248    125.7    surface        Adrian Mander
249    125.1    mlt            Katja Moehring, Alexander Schmidt
250    124.3    mlowess        Nicholas J. Cox
251    123.7    r2var          Emad Abd Elmessih Shehata, Sahra
                                Khaleel A. Mickaiel
252    123.7    ridgereg       Emad Abd Elmessih Shehata
253    123.0    rsource        Roger Newson
254    122.2    nbercycles     Christopher F Baum
255    122.0    cibar          Alexander Staudt
256    121.2    tabplot        Nicholas J. Cox
257    121.0    charlist       Nicholas J. Cox
258    121.0    vreverse       Nicholas J. Cox
259    120.9    fmm            Partha Deb
260    120.7    xsvmat         Roger Newson
261    120.3    firthlogit     Joseph Coveney
262    119.0    cntrade        Chuntao Li, Xuan Zhang
263    119.0    lars           Adrian Mander
264    119.0    seq            Nicholas J. Cox
265    118.0    chaid          Joseph N. Luchman
266    118.0    ivregress2     Roy Wada
267    117.7    outsum         Kerry L. Papps
268    116.2    listtab        Roger Newson
269    116.0    baplot         Paul Seed
270    116.0    cfout          Ryan Knight, Matthew White
271    115.7    byvar          Patrick Royston
272    115.5    metatrim       Thomas Steichen
273    115.0    wtp            Arne Risa Hole
274    112.8    sdecode        Roger Newson
275    112.7    kdens2         Christopher F Baum
276    112.2    tmap           Maurizio Pisati
```

277	111.7	actest	Mark E Schaffer, Christopher F Baum
278	111.7	senspec	Roger Newson
279	111.3	armaroots	Sune Karlsson
280	111.3	dups	Thomas Steichen, Nicholas J. Cox
281	110.3	poverty	Philippe Van Kerm
282	110.3	ptrend	Patrick Royston
283	110.1	lognfit	Stephen P. Jenkins
284	110.0	mvprobit	Stephen P. Jenkins, Lorenzo Cappellari
285	107.7	mean2	Roy Wada
286	107.7	uselab	Daniel Klein
287	107.0	dmexogxt	Christopher F Baum, Steven Stillman
288	106.8	sfcross	Vincenzo Atella, Federico Belotti, Giuseppe Ilardi, Silvio Daidone
289	106.7	svylorenz	Stephen P. Jenkins
290	106.6	labutil2	Daniel Klein
291	106.0	descsave	Roger Newson
292	106.0	makematrix	Nicholas J. Cox
293	105.7	cltest	Jeph Herrin
294	105.7	descogini	Alejandro Lopez-Feldman
295	105.4	mim	John B. Carlin, Patrick Royston, John C. Galati
296	105.3	mediation	Raymond Hicks, Dustin Tingley
297	104.3	mimstack	Patrick Royston, John C. Galati, John B. Carlin
298	104.0	labvars	Daniel Klein
299	103.0	_peers	Amine Ouazad
300	102.4	stpm2	Paul Lambert
301	101.7	ivreg2h	Mark E Schaffer, Christopher F Baum
302	101.0	ci2	Paul Seed
303	100.7	labvalpool	Daniel Klein
304	100.3	labmv	Daniel Klein
305	99.3	coldiag	Joseph Harkness
306	99.3	khb	Ulrich Kohler, Kristian Karlson
307	99.3	labmm	Daniel Klein
308	99.3	labrec	Daniel Klein
309	99.3	repest	Francesco Avvisati, François Keslair
310	99.3	rev	Daniel Klein
311	99.2	stselpre	Enzo Coviello
312	99.0	bidensity	Christopher F Baum, John Luke Gallup
313	99.0	labvalch3	Daniel Klein
314	99.0	matmap	Nicholas J. Cox
315	98.5	isko	John Hendrickx
316	98.4	listtex	Roger Newson
317	98.0	labvalcl	Daniel Klein
318	98.0	mimrgns	Daniel Klein
319	98.0	regcheck	Mehmet Mehmetoglu
320	96.7	statsmat	Christopher F Baum, Nicholas J. Cox
321	96.7	xtcips	Maximo Sangiacomo
322	96.3	radar	Adrian Mander
323	96.3	spregdpd	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
324	96.2	akdensity	Philippe Van Kerm
325	96.0	txtlabdef	Daniel Klein
326	95.0	hnblogit	Joseph Hilbe
327	94.7	renvarlab	Joe Canner
328	94.3	corrtab	Fred Wolfe
329	94.3	metacum	Ross Harris, Jonathan Sterne
330	93.7	ghansen	Jorge Eduardo Perez Perez
331	93.7	mcenter	Jeffrey S. Simons
332	93.3	fmlogit	Maarten L. Buis
333	93.0	stockquote	Nikos Askitas
334	92.9	spautoreg	Emad Abd Elmessih Shehata
335	92.8	johans	Ken Heinecke, Charles Morris, Patrick Joly
336	92.7	addplot	Ben Jann

337	92.7	full_palette	Nick Winter
338	92.0	dsginideco	Philippe Van Kerm, Stephen P. Jenkins
339	92.0	fagan	Ben Dwamena
340	91.8	metandi	Roger Harbord
341	91.0	spmon	Thomas Plümper, Eric Neumayer
342	90.7	sum2	Roy Wada
343	90.0	ineqfac	Stephen P. Jenkins
344	89.3	curvefit	Liu Wei
345	89.3	quandl	Felix Leung
346	89.2	bioprobit	Zurab Sajaia
347	88.3	heterogi	Iain Buchan, Julian Higgins, Nicola Orsini, Matteo Bottai
348	88.3	percom	Muhammad Rashid Ansari
349	87.8	varlag	Patrick Joly
350	87.0	mvcorr	Christopher F Baum, Nicholas J. Cox
351	86.8	paretofit	Philippe Van Kerm, Stephen P. Jenkins
352	86.0	kernreg2	Isaias H. Salgado-Ugarte, Nicholas J. Cox, Toru Taniuchi, Makoto Shimizu
353	85.7	ivactest	Christopher F Baum, Mark E Schaffer
354	85.1	paramed	Richard Emsley, Hanhua Liu
355	85.0	ivvif	David Roodman
356	83.8	distplot	Nicholas J. Cox
357	83.7	rmse	Roy Wada
358	83.3	dataout	Roy Wada
359	83.0	hadrilm	Christopher F Baum
360	83.0	stcmd	Roger Newson
361	82.3	robreg	Ben Jann
362	81.6	boxtid	Patrick Royston
363	81.3	copydesc	Nicholas J. Cox
364	81.3	csti	Philip M Jones
365	81.0	chaidforest	Joseph N. Luchman
366	81.0	gsample	Ben Jann
367	81.0	semipar	Nicolas Debarsy, Vincenzo Verardi
368	80.5	xtvar	Ulrich Glogowsky, Tobias Cagala
369	80.4	postrcspline	Maarten L. Buis
370	80.3	plotmatrix	Adrian Mander
371	80.3	spweight	Emad Abd Elmessih Shehata
372	80.3	lclogit	Daniele Pacifico, Hong il Yoo
373	80.2	mylabels	Nicholas J. Cox, Scott Merryman
374	80.0	wntstmvq	Christopher F Baum, Richard Sperling
375	79.9	switchr	Fred Zimmerman
376	79.1	clrbound	Victor Chernozhukov, Adam M. Rosen, Sokbae Lee, Wooyoung Kim
377	79.0	centcalc	Patrick Royston, Eileen Wright
378	78.9	betafit	Nicholas J. Cox, Stephen P. Jenkins, Maarten L. Buis
379	78.8	mahapick	David Kantor
380	78.7	mcmccqreg	Matthew Baker
381	78.3	bigtab	Paul H. Bern
382	78.3	designplot	Nicholas J. Cox
383	78.3	odkmeta	Matthew White
384	78.3	tabexport	Nicholas J. Cox
385	78.1	spmlreg	P. Wilner Jeanty
386	78.0	bayesmixedlogit	Matthew Baker
387	78.0	texdoc	Ben Jann
388	77.7	chinafin	Cheng Pan, Chuntao Li, Xuan Zhang
389	77.7	indeplist	Maarten L. Buis
390	77.0	hausman	Jeroen Weesie
391	77.0	spweightxt	Emad Abd Elmessih Shehata
392	76.7	findname	Nicholas J. Cox
393	76.7	kernreg1	Xavi Ramos, Makoto Shimizu, Toru Taniuchi, Isaias H. Salgado-Ugarte
394	76.0	xls2dta	Daniel Klein
395	75.3	estout1	Ben Jann
396	75.0	datacheck	Krishnan Bhaskaran

397	74.8	vececm	Patrick Joly
398	74.3	cleanchars	Lars Angquist
399	74.3	geocode	Adam Ozimek, Daniel Miles
400	74.3	isvar	Nicholas J. Cox
401	74.0	mundlak	Francisco (Paco) Perales
402	73.7	matodd	Nicholas J. Cox
403	73.7	catenate	Nicholas J. Cox
404	73.7	vallist	Patrick Joly
405	73.5	qpfit	Nicholas J. Cox
406	73.3	matvsort	Nicholas J. Cox
407	73.1	gb2fit	Stephen P. Jenkins
408	73.0	filelist	Robert Picard
409	72.7	domin	Joseph N. Luchman
410	72.7	itsa	Ariel Linden
411	72.6	sspecialreg	Christopher F Baum
412	72.3	bcoeff	Zhiqiang Wang, Nicholas J. Cox
413	72.3	byhist	Austin Nichols
414	72.3	iccvar	Eric C. Hedberg
415	72.3	lrdrop1	Zhiqiang Wang
416	72.3	xfrac	Stephen P. Jenkins
417	72.0	spineplot	Nicholas J. Cox
418	72.0	tablecol	Nick Winter
419	72.0	vselect	Charles Lindsey
420	71.3	vecar	Christopher F Baum
421	71.0	cqiv	Ivan Fernandez-Val, Amanda Kowalski, Victor Chernozhukov, Sukjin Han
422	71.0	xttrans2	Nicholas J. Cox
423	70.5	svygei_svyatk	Stephen P. Jenkins, Martin Biewen
424	70.3	difd	Laura Gibbons
425	70.3	doseresponse2	Marco Ventura, Barbara Guardabascio
426	70.3	omninorm	Christopher F Baum, Nicholas J. Cox
427	70.0	mss	J.M.C. Santos Silva, J.A.F. Machado
428	69.7	panelauto	Christopher F Baum
429	69.7	ivreset	Mark E Schaffer
430	69.7	sparkline	Nicholas J. Cox
431	69.7	xtewreg	Robert Parham, Timothy Erickson, Colin Jiang, Toni Whited
432	69.3	estwrite	Ben Jann
433	69.3	ivreg2hdfe	Dany Bahar
434	69.3	_gclsort	Philippe Van Kerm
435	69.0	inequal2	Philippe Van Kerm
436	68.9	stcoxgof	Enzo Coviello
437	68.7	ccmatch	Daniel E. Cook
438	68.6	smfit	Stephen P. Jenkins
439	68.3	goprobit	Stefan Boes
440	68.3	reset	Emad Abd Elmessih Shehata
441	68.0	krippalpha	Mona Krewel, Alexander Staudt
442	68.0	nlcheck	Ben Jann
443	68.0	spgrid	Maurizio Pisati
444	67.7	fbar	Nicholas J. Cox
445	67.7	genscore	Jean-Benoit Hardouin
446	67.7	movestay	Zurab Sajaia, Michael Lokshin
447	67.3	fgtest	Emad Abd Elmessih Shehata
448	67.3	ivtreatreg	Giovanni Cerulli
449	67.0	metaeff	Evangelos Kontopantelis, David Reeves
450	67.0	pantest2	Nicholas Oulton
451	67.0	venndiag	Jens M. Lauritsen
452	66.8	mfpigen	Patrick Royston
453	66.5	apc	Yang Yang, Sam Schulhofer-Wohl
454	66.3	genicv	Daniel Klein
455	66.3	matwrite	Andrew Shephard
456	65.8	dfl	Joao Pedro Azevedo
457	65.7	pvenn	Jan Ostermann, Wenfeng Gong
458	65.6	seqlogit	Maarten L. Buis
459	65.3	hangroot	Maarten L. Buis

460	65.0	dtobit2	Vince Wiggins
461	65.0	hbar	Nicholas J. Cox
462	65.0	mgof	Ben Jann
463	65.0	pyramid	Jens M. Lauritsen
464	64.9	raschtestv7	Jean-Benoit Hardouin
465	64.7	crossfold	Ben Daniels
466	64.7	dqd	Iliana Reggio, Ricardo Mora
467	64.7	fairlie	Ben Jann
468	64.7	xtile2	Zhiqiang Wang
469	64.5	hotdeck	Adrian Mander, David Clayton
470	64.3	netplot	Rense Corten
471	64.3	table1	Phil Clayton
472	64.3	xtpattern	Nicholas J. Cox
473	64.0	mergepoly	Michael Stepner, Robert Picard
474	64.0	stpm	Patrick Royston
475	63.8	metaan	David Reeves, Evangelos Kontopantelis
476	63.8	vecar6	Patrick Joly, Christopher F Baum
477	63.8	mvmeta	Ian White
478	63.7	mvtobit	Mikkel Barslund
479	63.3	multencode	Nicholas J. Cox
480	63.2	spregsarxt	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
481	62.7	utest	Jo Thori Lind, Halvor Mehlum
482	62.3	cf3	Thomas Steichen
483	62.3	ingap	Roger Newson
484	62.1	icdpic	Turner M. Osler, David R. Hahn, David E. Clark
485	62.0	levels	Nicholas J. Cox
486	62.0	sbbq	Philippe Bracke
487	61.7	graph3d	Davud Rostam-Afschar, Robin Jessen
488	61.7	iia	Jeroen Weesie
489	61.7	lookforit	Dan Blanchette
490	61.3	dirlist	Morten Andersen
491	61.3	nharvey	Christopher F Baum, Fabian Bornhorst
492	61.3	weakiv10	Keith Finlay, Leandro Magnusson, Mark E Schaffer
493	61.2	stpepemori	Enzo Coviello
494	60.5	raschtest	Jean-Benoit Hardouin
495	60.4	bspline	Roger Newson
496	60.3	mkest	Roy Wada
497	60.0	bgtest	Vince Wiggins, Christopher F Baum
498	60.0	seg	Sean F. Reardon
499	60.0	soepren	Ulrich Kohler
500	60.0	stpiece	Jesper B. Sorensen
501	59.9	doseresponse	Alessandra Mattei, Michela Bia
502	59.7	estsave	Michael Blasnik
503	59.7	lgraph	Timothy Mak
504	59.7	shapley2	Florian Chavez Juarez
505	59.7	spgmmxt	Emad Abd Elmessih Shehata
506	59.3	cf2	Thomas Steichen
507	59.3	imbalance	Shenyang Guo
508	59.0	lmcol	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
509	59.0	matpwcorr	Adrian Mander
510	59.0	spregfext	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
511	58.9	frm	Joaquim J. S. Ramalho
512	58.8	graphlog	Martin Rune Hansen
513	58.8	oaxaca8	Ben Jann
514	58.4	alorenz	Joao Pedro Azevedo, Samuel Franco
515	58.3	denton	Sylvia Hristakeva, Christopher F Baum
516	58.3	gpscore2	Marco Ventura, Barbara Guardabascio
517	58.3	parallel	George Vega Yon
518	58.2	lincomest	Roger Newson
519	58.0	cutpt	Phil Clayton

520	58.0	ipf	Adrian Mander
521	58.0	lmhmss2	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
522	58.0	spregdhp	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
523	57.7	eqprhistogram	Nicholas J. Cox
524	57.3	drarea	Adrian Mander
525	57.3	kr20	Herve M. Caci
526	57.3	ttab	Federico Belotti
527	57.0	semean	Christopher F Baum
528	56.7	ralloc	Philip Ryan
529	56.3	btobit	David Vncent
530	56.3	diagtest	Aurelio Tobias
531	56.0	haif	Roger Newson
532	56.0	sampsi_mcc	Adrian Mander
533	55.7	strparse	Michael Blasnik, Nicholas J. Cox
534	55.7	worldstat	Damian Clarke
535	55.5	capass	Max Loeffler
536	55.5	tabform	Le Dang Trung
537	55.3	combineplot	Nicholas J. Cox
538	55.3	concindc	Zhuo (Adam) Chen
539	55.3	dfsummary	Maximo Sangiacomo
540	55.3	intgph	Bennet Zelner, Dan Blanchette
541	55.3	qll	Christopher F Baum
542	55.3	spkde	Maurizio Pisati
543	55.3	tablemat	Amadou Bassirou Diallo
544	55.0	cpigen	Austin Nichols
545	54.8	nearstat	P. Wilner Jeanty
546	54.7	allpossible	Nicholas J. Cox
547	54.7	doubleb	Alejandro Lopez-Feldman
548	54.7	fitstat_ers	Christian Gregory
549	54.3	asciiplot	Svend Juul, Michael Blasnik, Nicholas J. Cox
550	54.3	decompose	Ben Jann
551	54.2	mfpi	Patrick Royston
552	54.2	loevh	Jean-Benoit Hardouin
553	54.2	markdoc	E.F. Haghish
554	54.0	dfgls	Richard Sperling, Christopher F Baum
555	54.0	dta2sav	Dirk Enzmann
556	54.0	inteff3	Katja Sonderhof, Thomas Cornelissen
557	54.0	pisareg	Maciej Jakubowski
558	54.0	svyselmlog	R. E. De Hoyos
559	53.9	weaver	E.F. Haghish
560	53.7	markov	Nicholas J. Cox
561	53.7	tab2way	Philip Ryan
562	53.3	csvconvert	Alberto A. Gaggero
563	53.3	ddf2dct	Austin Nichols
564	53.3	grstest2	Markus Ibert
565	53.3	pgmhaz8	Stephen P. Jenkins
566	53.0	poi2hdfe	Paulo Guimaraes
567	53.0	ssm	Sophia Rabe-Hesketh, Alfonso Miranda
568	52.7	digdis	Ben Jann
569	52.7	ipdmetan	David Fisher
570	52.7	norm	Chiara Mussida, Muhammad Rashid Ansari
571	52.3	ineqrbd	Stephen P. Jenkins, Carlo V. Fiorio
572	52.3	rc_spline	William D. Dupont, W. Dale Plummer, Jr.
573	52.2	duncan	Ben Jann
574	51.7	etime	Dan Blanchette
575	51.7	perturb	John Hendrickx
576	51.6	qsim	Fred Wolfe
577	51.5	spregsdmxt	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
578	51.5	ivpois	Austin Nichols
579	51.0	cfvars	Nicholas J. Cox
580	51.0	clorenz	Araar Abdelkrim

581	50.8	regoprob	Stefan Boes
582	50.7	madfuller	Christopher F Baum
583	50.7	xrigls	Eileen Wright, Patrick Royston
584	50.5	synlight	E.F. Haghish
585	50.3	lstrfun	Dan Blanchette
586	50.3	touch	Ari Friedman
587	50.2	cart	Wim van Putten
588	50.0	moss	Nicholas J. Cox, Robert Picard
589	49.9	regoprob2	Udo Schneider, Christian Pfarr, Andreas Schmid
590	49.7	radiusmatch	Martin Huber, Andreas Steinmayr, Michael Lechner
591	49.7	samplepps	Stephen P. Jenkins
592	49.5	adecomp	Viviane Sanfelice, Joao Pedro Azevedo, Minh Cong Nguyen
593	49.0	checkfor2	Amadou Bassirou Diallo, Jean-Benoit Hardouin
594	49.0	concord	Nicholas J. Cox, Thomas Steichen
595	49.0	gpreg	Johannes F. Schmieder
596	48.7	movavg	George Vega Yon
597	48.7	partgam	Jens M. Lauritsen, Svend Kreiner
598	48.5	psiduse	Ulrich Kohler
599	48.3	usesome	Daniel Klein
600	48.0	icc23	Luis C.Orozco, Paul F. Visintainer
601	48.0	optifact	Paul Millar
602	48.0	pctrim	Michael Barker
603	48.0	ketchup	E.F. Haghish
604	47.7	strrec	Daniel Klein
605	47.7	xcontract	Roger Newson
606	47.6	piaactools	Maciej Jakubowski, Artur Pokropek
607	47.3	fese	Austin Nichols
608	47.3	mvfiles	Lars Angquist
609	47.3	num2words	Eric Booth
610	47.3	spweightcs	Emad Abd Elmessih Shehata
611	47.0	grubbs	Nicolas Couderc
612	47.0	xbrcspline	Nicola Orsini
613	46.8	hshaz	Stephen P. Jenkins
614	46.7	cohend	David Tannenbaum
615	46.7	hireg	Paul H. Bern
616	46.7	rdcv	Boris Kaiser
617	46.7	renames	Nicholas J. Cox
618	46.6	mmsel	Sami Souabni
619	46.3	geekel2d	Jean-Benoit Hardouin
620	46.3	pcorr2	Richard Williams
621	46.2	xriml	Patrick Royston
622	46.2	rowsort	Nicholas J. Cox
623	46.0	eba	Gregorio Impavido
624	46.0	gausshermite	Jean-Benoit Hardouin
625	46.0	jmpierce	Ben Jann
626	46.0	paverage	P. Wilner Jeanty
627	46.0	regdis	Roy Wada
628	46.0	suchowtest	Diallo Ibrahima Amadou
629	45.9	spregsemxt	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
630	45.7	listutil	Nicholas J. Cox
631	45.7	arimafit	Christopher F Baum
632	45.7	corr_svy	Nick Winter
633	45.7	tabmult	Minh Nguyen
634	45.7	twoway_parea	Sergiy Radyakin
635	45.5	hplogit	Joseph Hilbe
636	45.5	tabhbar	Nicholas J. Cox
637	45.3	grstest	Rajesh Tharyan
638	45.3	jmpierce2	Ben Jann
639	45.3	orth_out	Joe Long
640	45.3	workdays	Bill Rising

641	45.0	ctreatreg	Giovanni Cerulli
642	45.0	dftol	Ignacio López de Ullibarri
643	45.0	punaf	Roger Newson
644	45.0	xtregre2	Scott Merryman
645	45.0	xtsemipar	Vincenzo Verardi, François Libois
646	44.8	tfr2	Bruno SCHOUMAKER
647	44.8	eventstudy	Xin Xu, Xuan Zhang, Chuntao Li
648	44.7	xtregdhp	Emad Abd Elmessih Shehata
649	44.3	landemets	Ignacio López de Ullibarri
650	44.3	matsort	Paul Millar
651	44.3	scatter3d	Thomas Roca
652	44.3	vioplot	Nick Winter, Austin Nichols
653	44.2	hcavar	Jean-Benoit Hardouin
654	44.2	spregsem	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
655	44.0	centroid	D. H. Judson
656	44.0	cndnmb3	Michael Blasnik
657	44.0	rowranks	Nicholas J. Cox
658	43.7	ivgmm0	David M. Drukker, Christopher F Baum
659	43.7	labsumm	Thomas Steichen
660	43.7	lomackinlay	Christopher F Baum
661	43.7	oddsrisk	Joseph Hilbe
662	43.3	tex3pt	Derek Wolfson
663	43.0	censornb	Joseph Hilbe
664	43.0	gs2slsarxt	Emad Abd Elmessih Shehata
665	43.0	pcmodel	Jean-François Hamel
666	43.0	scdensity	Joerg Luedicke
667	43.0	ssi	Philip M Jones
668	43.0	xtab	Tony Brady
669	42.7	gammasym	Jean-Benoit Hardouin
670	42.7	meoprobit	Thomas Cornelissen
671	42.7	sensatt	Tommaso Nannicini
672	42.7	spregrext	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
673	42.7	svret	Julian Reif
674	42.6	rtfutil	Roger Newson
675	42.3	cipolate	Nicholas J. Cox
676	42.3	matnames	Austin Nichols
677	42.3	pairplot	Nicholas J. Cox
678	42.3	plotbeta	Adrian Mander
679	42.3	twfe	Nikolas Mittag
680	42.3	gformula	Rhian Daniel
681	42.0	adjacent	Nicholas J. Cox
682	42.0	devcon	Ben Jann
683	42.0	povimp	Hai-Anh H. Dang, Minh Cong Nguyen
684	42.0	treatrew	Giovanni Cerulli
685	42.0	white	Jeroen Weesie
686	41.9	modeldiag	Nicholas J. Cox
687	41.8	pisatools	Maciej Jakubowski, Artur Pokropek
688	41.7	anketest	P. Wilner Jeanty
689	41.7	chunky	David Elliott
690	41.7	fitmacro	John Hendrickx
691	41.7	smhsiao	Nick Winter
692	41.3	gologit29	Richard Williams
693	41.3	pcmtest	Jean-François Hamel
694	41.3	rcd	Nikos Askitas, Dan Blanchette
695	41.3	rcspline	Nicholas J. Cox
696	41.2	stmixed	Michael J. Crowther
697	41.1	tobithetm	Emad Abd Elmessih Shehata
698	41.0	dummies	Nicholas J. Cox
699	41.0	gs2sls	Emad Abd Elmessih Shehata
700	41.0	spregsdm	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
701	41.0	wtpcikr	P. Wilner Jeanty
702	41.0	xcollapse	Roger Newson

703	40.9	rnd	Joseph Hilbe
704	40.7	fraclogit	Daniel A. Powers
705	40.7	fren	Liu Wei
706	40.7	ineq	Nicholas J. Cox
707	40.7	mergeall	Ryan Knight
708	40.7	mgen	Ben Jann
709	40.7	stkerhaz	Enzo Coviello
710	40.5	clogithet	Arne Risa Hole
711	40.5	triprobit	Antoine Terracol
712	40.3	corsp	Daniel Klein
713	40.3	ipfweight	Michael Bergmann
714	40.3	plausexog	Damian Clarke
715	40.3	stlda	Ignacio López de Ullibarri
716	40.3	xtregfem	Emad Abd Elmessih Shehata
717	40.2	wdireshape	P. Wilner Jeanty
718	40.0	checkrob	Mikkel Barslund
719	40.0	metaprop	Marc Arbyn, Marc Aerts, Victoria Nyawira Nyaga
720	40.0	mm_regress	Christophe Croux, Vincenzo Verardi
721	39.7	mpovline	Joao Pedro Azevedo, Viviane Sanfelice
722	39.7	psemail	Xuan Zhang, Chuntao Li, Dongliang Cui
723	39.5	spmstardhxt	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
724	39.3	qic	James Cui
725	39.3	quine	Matthew Baker
726	39.3	spcs2xt	Emad Abd Elmessih Shehata
727	39.3	spregsacxt	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
728	39.3	_gprod	Philip Ryan
729	39.2	dpplot	Nicholas J. Cox
730	39.0	dyads	John-Paul Ferguson
731	39.0	lookfor_all	Dan Blanchette, Michael Lokshin, Zurab Sajaia
732	38.7	bihist	Austin Nichols
733	38.7	gs3sls	Emad Abd Elmessih Shehata
734	38.7	psacalc	Emily Oster
735	38.7	sortl	Dirk Enzmann
736	38.7	spglsxt	Emad Abd Elmessih Shehata
737	38.5	isco	John Hendrickx
738	38.5	concindex	Amadou Bassirou Diallo
739	38.3	beamplot	Nicholas J. Cox
740	38.3	gologit	Vincent Kang Fu
741	38.3	reformat	Tony Brady
742	38.3	statplot	Nicholas J. Cox, Eric Booth
743	38.0	checkreg3	Christopher F Baum
744	38.0	corrtable	Nicholas J. Cox
745	38.0	hmap	Austin Nichols
746	38.0	keyplot	Nicholas J. Cox
747	38.0	mvport	Alberto Dorantes
748	38.0	powercal	Roger Newson
749	38.0	readreplace	Matthew White, Ryan Knight
750	38.0	spwmatfill	P. Wilner Jeanty
751	38.0	tabcount	Nicholas J. Cox
752	38.0	mycd10	Joe Canner
753	37.8	sphdist	Bill Rising
754	37.7	fastcd	Nick Winter
755	37.7	fastxtile	Michael Stepner
756	37.7	funnelcompar	Sylvia Forni, Rosa Gini
757	37.7	rhausman	Boris Kaiser
758	37.7	rocmic	Robert Froud
759	37.7	xpredict	Patrick Royston
760	37.7	xtregsam	Emad Abd Elmessih Shehata
761	37.5	segregation	Carlos Gradin
762	37.4	decomp	Ian Watson
763	37.3	cycleplot	Nicholas J. Cox

764	37.3	dpredict	J. Katriak
765	37.3	emh	Joseph Coveney
766	37.3	hplot	Nicholas J. Cox
767	37.3	xtnptimevar	Diallo Ibrahima Amadou
768	37.2	confa	Stanislav Kolenikov
769	37.0	autorename	Julian Reif
770	37.0	detect	Jean-Benoit Hardouin
771	37.0	krls	Chad Hazlett, Jens Hainmueller, Jeremy Ferwerda
772	37.0	sf36v2	Monica Daigl
773	37.0	traveltime	Adam Ozimek, Daniel Miles
774	37.0	xtcsi	Maximo Sangiacomo
775	36.7	adjmean	Joanne M. Garrett
776	36.7	cochran	Ben Jann
777	36.7	difwithpar	Laura Gibbons
778	36.7	find	Austin Nichols
779	36.7	frontier_teci	Scott Merryman
780	36.7	glcurve7	Philippe Van Kerm, Stephen P. Jenkins
781	36.7	leebounds	Harald Tauchmann
782	36.7	rollstat	Maximo Sangiacomo, Demian Panigo
783	36.3	gs2slsxt	Emad Abd Elmessih Shehata
784	36.3	lmhwaldxt	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
785	36.3	soepuse	Ulrich Kohler
786	36.3	spsurv	Stephen P. Jenkins
787	36.2	stjm	Michael J. Crowther
788	36.0	dsconcat	Roger Newson
789	36.0	lmnad	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
790	36.0	loocv	Manuel Barron
791	36.0	mcmclinear	Sam Schulhofer-Wohl
792	36.0	missing	Jose Maria Sanchez Saez
793	36.0	mlogitroc	Leif E. Peterson
794	36.0	overlay	Adrian Mander
795	35.7	epiweek	Tim Chu
796	35.7	evhistplot	Henrik Stovring
797	35.7	jarowinkler	James Feigenbaum
798	35.5	intext	Roger Newson
799	35.3	iop	Isidro Soloaga, Florian Chavez Juarez
800	35.3	more_clarify	Javier Marquez Pena
801	35.3	rwg	Muhammad Rashid Ansari
802	35.0	cid	Patrick Royston
803	35.0	ivreg210	Steven Stillman, Mark E Schaffer, Christopher F Baum
804	34.9	dirtools	Roy Wada, Ulrich Kohler
805	34.8	sptobitsac	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
806	34.7	codebookout	Kishor K. Das
807	34.7	histbox	Philip B. Ender
808	34.7	integrate	Adrian Mander
809	34.7	ldecomp	Maarten L. Buis
810	34.7	paran	Alexis Dinno
811	34.5	cv	Mehmet Mehmetoglu
812	34.4	cluster	D. H. Judson
813	34.3	distrate	Enzo Coviello
814	34.3	gdecomp	Tamas Bartus
815	34.3	hansen2	Nicholas J. Cox
816	34.3	ipdforest	Evangelos Kontopantelis, David Reeves
817	34.3	metagen	Pantelis Bagos
818	34.3	tddens	Austin Nichols
819	34.3	trimmean	Nicholas J. Cox
820	34.3	xtregam	Emad Abd Elmessih Shehata
821	34.0	bme	Daniel A. Powers, Paul T. von Hippel
822	34.0	calibrate	John D´Souza
823	34.0	dmout	Michael Barker

824	34.0	lincheck	Alex Gamma
825	34.0	lmnadxt	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
826	34.0	log2html	Christopher F Baum, Bill Rising, Nicholas J. Cox
827	34.0	spxttobit	Emad Abd Elmessih Shehata
828	34.0	vmatch	Guy D. van Melle
829	33.8	spregsac	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
830	33.8	mtreatreg	Partha Deb
831	33.7	zoib	Maarten L. Buis
832	33.7	anogi	Ben Jann
833	33.7	expandby	Nicholas J. Cox
834	33.7	expgen	Roger Newson
835	33.7	meta_lr	Aijing Shang
836	33.7	mktab	Nick Winter
837	33.6	runmplus	Rich Jones
838	33.6	xtreghet	Emad Abd Elmessih Shehata
839	33.3	bsopm	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
840	33.3	lmcovreg3	Emad Abd Elmessih Shehata
841	33.3	powersim	Joerg Luedicke
842	33.3	xgroup	Roger Newson
843	33.2	usagelog	Dan Blanchette
844	33.1	fodstr	William Gould
845	33.0	crossplot	Nicholas J. Cox
846	33.0	des2	Daniel Klein
847	33.0	lmhwald	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
848	33.0	oaxaca9	Ben Jann
849	33.0	regcoef	Mehmet Mehmetoglu
850	33.0	sskapp	Jan Brogger
851	33.0	xtregwhm	Emad Abd Elmessih Shehata
852	32.8	hlm	Sean F. Reardon
853	32.7	fvregen	Roger Newson
854	32.7	misum	Daniel Klein
855	32.7	outfix	Gero Lipsmeier
856	32.7	panels	Ben Jann
857	32.7	prepar	Laura Gibbons
858	32.7	quickicc	Eric C. Hedberg
859	32.7	spgmm	Emad Abd Elmessih Shehata
860	32.7	xtidt	Emad Abd Elmessih Shehata
861	32.7	xtpatternvar	Nicholas J. Cox
862	32.5	goelevation	Chamara Anuranga, J.V. Jayanthan
863	32.5	kalpha	Daniel Klein
864	32.5	triplot	Nicholas J. Cox
865	32.4	spmstarxt	Emad Abd Elmessih Shehata
866	32.3	diagreg2	Emad Abd Elmessih Shehata
867	32.3	fdta	Liu Wei
868	32.3	graphbinary	Adrian Mander
869	32.3	imvol	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
870	32.3	nopomatch	Nopo, Hugo, Hoyos, Alejandro, Atal, Juan Pablo
871	32.3	project	Robert Picard
872	32.3	robjb	Catherine Vermandele, Vincenzo Verardi, Wouter Gelade
873	32.3	shapley	Stanislav Kolenikov
874	32.3	theilr2	Emad Abd Elmessih Shehata
875	32.3	xtarsim	Giovanni S.F. Bruno
876	32.2	panelunit	Christopher F Baum
877	32.1	hapipf	Adrian Mander
878	32.0	distmatch	Roy Wada
879	32.0	gengroup	Jean-Benoit Hardouin
880	32.0	geocodeopen	Michael L. Anderson

881	32.0	gvselect	Simon Sheather, Charles Lindsey
882	32.0	gzsave	Henrik Stovring
883	32.0	lmasem	Emad Abd Elmessih Shehata
884	32.0	reset2	Emad Abd Elmessih Shehata
885	32.0	sptobitsar	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
886	32.0	tsgraph	Christopher F Baum, Nicholas J. Cox
887	32.0	writekml	Gabi Huiber
888	31.8	matrixof	Nicholas J. Cox
889	31.7	iccconf	Paul F. Visintainer
890	31.7	lmareg3	Emad Abd Elmessih Shehata
891	31.7	maketex	Antoine Terracol
892	31.7	randomtag	Robert Picard
893	31.7	sampsi_reg	Adrian Mander
894	31.6	dirifit	Nicholas J. Cox, Maarten L. Buis, Stephen P. Jenkins
895	31.3	bicdrop1	Paul Millar
896	31.3	colelms	Zhiqiang Wang, Mark S Pearce
897	31.3	fsx	Gabriel Rossman, Nicholas J. Cox
898	31.3	gmap	Thomas Roca
899	31.3	keepvar	P. Wilner Jeanty
900	31.3	lmavonxt	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
901	31.3	pwcorr2	Christopher F Baum
902	31.3	robumeta	Eric C. Hedberg
903	31.3	scls	J.M.C. Santos Silva
904	31.3	scores	Dirk Enzmann
905	31.3	sf36	Philip Ryan
906	31.3	smcl2do	Bill Rising
907	31.3	sptobitsem	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
908	31.2	grcompare	Jun Xu
909	31.2	spmstardxt	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
910	31.2	weibullfit	Stephen P. Jenkins, Nicholas J. Cox
911	31.0	archqq	Sune Karlsson
912	31.0	cortesti	Herve M. Caci
913	31.0	lmhreg3	Emad Abd Elmessih Shehata
914	31.0	lxpct_2	Margaret M. Weden
915	31.0	ordplot	Nicholas J. Cox
916	31.0	sparl	Nicholas J. Cox
917	31.0	spregsar	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
918	31.0	usexmlex	Sergiy Radyakin
919	31.0	xtregrem	Emad Abd Elmessih Shehata
920	30.9	poparms	Ashley Holland, David M. Drukker, Matias D. Cattaneo
921	30.9	switch_oprobit	Christian Gregory
922	30.8	stcompadj	Enzo Coviello
923	30.8	ivprob-ivtobit6	Joseph Harkness
924	30.7	equation	Liu Wei
925	30.7	fmmlc	Joerg Luedicke
926	30.7	lmnsem	Emad Abd Elmessih Shehata
927	30.7	mca	Philippe Van Kerm
928	30.7	mcl	John Hendrickx
929	30.7	probexog-tobexog	Christopher F Baum
930	30.7	proprcspline	Maarten L. Buis
931	30.7	runparscale	Laura Gibbons
932	30.7	sls	Michael Barker
933	30.7	svvarlbl	Desmond E. Williams
934	30.7	traces	Jean-Benoit Hardouin
935	30.7	xtregbem	Emad Abd Elmessih Shehata
936	30.6	spreghetxt	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
937	30.4	extreme	David Roodman

938	30.3	addnotes	Jeff Arnold
939	30.3	archlm	Christopher F Baum, Vince Wiggins
940	30.3	changemean	Samuel Franco, Joao Pedro Azevedo
941	30.3	ctabstat	Nicholas J. Cox
942	30.3	d3network	Sabrina Vogel, Sebastian Pink
943	30.3	factmerg	Roger Newson
944	30.3	lmhgl	Emad Abd Elmessih Shehata
945	30.3	probitiv	Jonah B. Gelbach
946	30.3	xtregbn	Emad Abd Elmessih Shehata
947	30.2	scsomersd	Roger Newson
948	30.1	episens	Rino Bellocco, Sander Greenland, Nicola Orsini
949	30.0	ashell	Nikos Askitas
950	30.0	bayerhanck	Christoph Hanck, Christian Bayer
951	30.0	bpmedian	Roger Newson
952	30.0	centpow	Z. P. Neal
953	30.0	gs3slsar	Emad Abd Elmessih Shehata
954	30.0	historaj	Rajesh Tharyan
955	30.0	invcdf	Ben Jann
956	30.0	lmhlmxt	Emad Abd Elmessih Shehata
957	30.0	lmndh	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
958	30.0	lmnwhitext	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
959	30.0	ocratio	Rory Wolfe
960	30.0	pam	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
961	30.0	plssas	Adrian Mander
962	30.0	reshape8	Bill Rising
963	30.0	tabletutorial	Ben Jann
964	30.0	tomode	Nicholas J. Cox, Fred Wolfe
965	30.0	xtregmle	Emad Abd Elmessih Shehata
966	30.0	xtregwem	Emad Abd Elmessih Shehata
967	30.0	spautoc	Nicholas J. Cox
968	29.9	outdat	Ulrich Kohler
969	29.8	skdecomp	Bernardo Atuesta, Joao Pedro Azevedo, Andres Castaneda, Viviane Sanfelice
970	29.7	avplot3	Christopher F Baum
971	29.7	benford	Nikos Askitas
972	29.7	cctable	Peter Makary, Gilles Desve
973	29.7	ci_marg_mu	Sophia Rabe-Hesketh
974	29.7	collapse2	David Roodman
975	29.7	durbinh	Christopher F Baum, Vince Wiggins
976	29.7	hummels	Muhammad Rashid Ansari
977	29.7	lmalb	Emad Abd Elmessih Shehata
978	29.7	metagraph	Adrian Mander
979	29.7	pajek2stata	Rense Corten
980	29.7	reffadjust	Tom Palmer, Corrie Macdonald-Wallis
981	29.7	samplesize	Adrian Mander
982	29.7	spearman2	Christopher F Baum
983	29.7	varlocal	Viviane Sanfelice
984	29.7	xbalance	Mark Fredrickson, Jake Bowers, Ben Hansen
985	29.6	ckvar	Bill Rising
986	29.5	sptobitsarxt	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
987	29.3	adjprop	Joanne M. Garrett
988	29.3	apcd	Louis Chauvel
989	29.3	autofmt	Roy Wada
990	29.3	calibest	John D´Souza
991	29.3	gets	Damian Clarke
992	29.3	harmby	Roger Newson
993	29.3	hash	Andrew Maurer
994	29.3	lmfreg	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata

995	29.3	lmhsem	Emad Abd Elmessih Shehata
996	29.3	metamiss	Julian Higgins, Ian White
997	29.3	ordvar	Mike Lacy
998	29.3	rolling2	Christopher F Baum
999	29.3	scoregof	Richard Chiburis
1000	29.3	spmstarh	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1001	29.3	sptobitsacxt	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1002	29.2	partpred	Paul Lambert
1003	29.2	stgenreg	Paul Lambert, Michael J. Crowther
1004	29.0	circular	Nicholas J. Cox
1005	29.0	anymatch	Roy Wada
1006	29.0	fracirf	Christopher F Baum
1007	29.0	ghxt	Emad Abd Elmessih Shehata
1008	29.0	group1d	Nicholas J. Cox
1009	29.0	jonter	Joseph Coveney
1010	29.0	lmabxt	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
1011	29.0	medoid	D. H. Judson
1012	29.0	mergemany	Damian Clarke
1013	29.0	mod11	George Vega Yon
1014	29.0	moreobs	Nicholas J. Cox
1015	29.0	nearest	Nicholas J. Cox
1016	29.0	nwind	Helmut Farbmacher
1017	29.0	overfit	Marcel Bilger
1018	29.0	ppschromy	Jonathan Mendelson
1019	29.0	selectvars	Nicholas J. Cox
1020	29.0	svytabs	Michael Blasnik
1021	29.0	xtine	Christine Cook
1022	28.9	dthaz	Alexis Dinno
1023	28.8	crtest	Joao Pedro Azevedo
1024	28.8	leftalign	Robert Picard
1025	28.8	mcmcstats	Sam Schulhofer-Wohl
1026	28.7	barplot2	Nicholas J. Cox
1027	28.7	intcens	Jamie Griffin
1028	28.7	labelmiss	Stanislav Kolenikov
1029	28.7	lmeg	Emad Abd Elmessih Shehata
1030	28.7	lmhhp2	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
1031	28.7	lrplot	Jan Brogger
1032	28.7	pre	Paul Millar
1033	28.7	qqvalue	Roger Newson
1034	28.7	riigen	Lars E. Kroll
1035	28.7	rmfiles	Lars Angquist
1036	28.7	sslope	Jeffrey S. Simons
1037	28.7	svybsamp2	R. E. De Hoyos
1038	28.7	swboot	Joanne M. Garrett
1039	28.7	urcovar	Christopher F Baum
1040	28.5	bking	Martha Lopez, Christopher F Baum
1041	28.5	cihplot	Nicholas J. Cox
1042	28.3	bitobit	Daniel Lawson
1043	28.3	cpcorr	Nicholas J. Cox
1044	28.3	findval	Stanislav Kolenikov
1045	28.3	lmnreg3	Emad Abd Elmessih Shehata
1046	28.3	probcalc	Leif E. Peterson
1047	28.3	rdpower	Eric C. Hedberg
1048	28.3	title	Jan Brogger
1049	28.3	turnbull	Joao Pedro Azevedo
1050	28.2	python	James Fiedler
1051	28.2	bcii	Robert Froud
1052	28.2	sptobitsemxt	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1053	28.1	sptobitmstardhxt	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel

1054	28.1	sptobitsdmxt	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1055	28.0	codebook2	Paul H. Bern
1056	28.0	dataframe	Andrew Maurer
1057	28.0	ds3	Nicholas J. Cox
1058	28.0	gnpoisson	Joseph Hilbe
1059	28.0	lmadurhxt	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1060	28.0	lmgc	Emad Abd Elmessih Shehata
1061	28.0	lmngr	Emad Abd Elmessih Shehata
1062	28.0	median	Mario Cleves
1063	28.0	rdci	Joseph Coveney
1064	28.0	varcase	John R. Gleason
1065	28.0	xtmixed_corr	Roberto G. Gutierrez
1066	27.9	spmstard	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1067	27.8	drdecomp	Andres Castaneda, Joao Pedro Azevedo, Viviane Sanfelice
1068	27.7	sptobitmstarh	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1069	27.7	alsmle	Emad Abd Elmessih Shehata
1070	27.7	butterworth	Christopher F Baum, Martha Lopez
1071	27.7	ellipticity	Catherine Vermandele, Vincenzo Verardi
1072	27.7	estrat	Jeremy Ferwerda
1073	27.7	holsti	Mona Krewel, Alexander Staudt, Julia Partheymüller
1074	27.7	levene	Herve M. Caci
1075	27.7	lmabg	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1076	27.7	lmadurmxt	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1077	27.7	sbri	Nicola Orsini
1078	27.7	slideplot	Nicholas J. Cox
1079	27.7	tabhplot	Nicholas J. Cox
1080	27.7	texify	Roy Wada
1081	27.7	treatoprobit	Christian Gregory
1082	27.3	addtxt	Gary Longton
1083	27.3	agrm	Alejandro Ecker
1084	27.3	classplot	Lars E. Kroll
1085	27.3	dfao	Richard Sperling
1086	27.3	dotex	Roger Newson
1087	27.3	eret2	Ben Jann
1088	27.3	fmiss	Florian Chavez Juarez
1089	27.3	ftrans	Liu Wei
1090	27.3	kitchensink	Francisco (Paco) Perales
1091	27.3	levpredict	Christopher F Baum
1092	27.3	listsome	Robert Picard
1093	27.3	lmabgnl	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1094	27.3	lmcovsem	Emad Abd Elmessih Shehata
1095	27.3	lmhhpxt	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
1096	27.3	personage	Nicholas J. Cox
1097	27.3	ridge2sls	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
1098	27.3	sbrowni	Herve M. Caci
1099	27.3	stexpect	Enzo Coviello
1100	27.3	xglm	Roger Newson
1101	27.2	logithetm	Emad Abd Elmessih Shehata
1102	27.2	punafcc	Roger Newson
1103	27.2	staticfc	Christopher F Baum
1104	27.1	spmstar	Emad Abd Elmessih Shehata
1105	27.0	cb2html	Phil Bardsley
1106	27.0	genhwcci	James Cui
1107	27.0	geochart	Sergiy Radyakin

```
1108    27.0    grfreq          Jan Brogger
1109    27.0    idonepsu        Joshua H. Sarver
1110    27.0    irr             Maximo Sangiacomo
1111    27.0    lmabg2          Emad Abd Elmessih Shehata, Sahra
                                  Khaleel A. Mickaiel
1112    27.0    lmabp2          Sahra Khaleel A. Mickaiel, Emad Abd
                                  Elmessih Shehata
1113    27.0    lmcovxt         Emad Abd Elmessih Shehata
1114    27.0    lmhhp           Emad Abd Elmessih Shehata, Sahra
                                  Khaleel A. Mickaiel
1115    27.0    lmnjbnl         Emad Abd Elmessih Shehata, Sahra
                                  Khaleel A. Mickaiel
1116    27.0    locpr           Austin Nichols
1117    27.0    mrdum           Lee E. Sieswerda
1118    27.0    mulogit         Leif E. Peterson
1119    27.0    parplot         Nicholas J. Cox
1120    27.0    pem             Sahra Khaleel A. Mickaiel, Emad Abd
                                  Elmessih Shehata
1121    27.0    pieplot         Nicholas J. Cox
1122    27.0    progres         Philippe van Kerm, Andreas Peichl
1123    27.0    recap           Matthias an der Heiden
1124    27.0    shufflevar      Gabriel Rossman
1125    26.8    psidtools       Ulrich Kohler
1126    26.8    regaxis         Roger Newson
1127    26.8    fastcollapse    Andrew Maurer
1128    26.7    eaalogit        Arne Risa Hole
1129    26.7    gs2slsar        Emad Abd Elmessih Shehata
1130    26.7    hodgesl         Shenyang Guo
1131    26.7    ksi             Muhammad Rashid Ansari
1132    26.7    lmhcw           Sahra Khaleel A. Mickaiel, Emad Abd
                                  Elmessih Shehata
1133    26.7    modlpr          Vince Wiggins, Christopher F Baum
1134    26.7    polyspline      Roger Newson
1135    26.7    sampicc         Paul F. Visintainer
1136    26.7    sptobitmstarhxt  Sahra Khaleel A. Mickaiel, Emad Abd
                                  Elmessih Shehata
1137    26.7    stns            Isabelle Urmès, Michel Grzebyk
1138    26.7    stockcapit      Diallo Ibrahima Amadou
1139    26.6    cquad           Francesco Bartolucci
1140    26.5    complogit       Glenn Hoetker
1141    26.5    tobitiv         Jonah B. Gelbach
1142    26.4    sptobitmstardxt  Emad Abd Elmessih Shehata, Sahra
                                  Khaleel A. Mickaiel
1143    26.4    spmstardh       Sahra Khaleel A. Mickaiel, Emad Abd
                                  Elmessih Shehata
1144    26.3    alphawgt        Ben Jann
1145    26.3    avg_effect      Christopher Robert
1146    26.3    betacoef        Christopher F Baum
1147    26.3    cfitzrw         Christopher F Baum, Martha Lopez
1148    26.3    clv             Jean-Benoit Hardouin
1149    26.3    domdiag         Nicholas J. Cox
1150    26.3    emailme         Ed Gerrish
1151    26.3    er              Carlos Gradin
1152    26.3    geivars         Stephen P. Jenkins
1153    26.3    glmdeco         Boris Kaiser
1154    26.3    lmhlrxt         Emad Abd Elmessih Shehata
1155    26.3    lmndp           Emad Abd Elmessih Shehata, Sahra
                                  Khaleel A. Mickaiel
1156    26.3    lmnjb           Emad Abd Elmessih Shehata
1157    26.3    lmsrd           Emad Abd Elmessih Shehata
1158    26.3    png2rtf         Austin Nichols
1159    26.3    pspline         Ben Jann, Roberto G. Gutierrez
1160    26.3    raewma          Brent McSharry
1161    26.3    running         Patrick Royston, Peter Sasieni,
                                  Nicholas J. Cox
```

1162	26.3	tmpm	Alan Cook
1163	26.3	sptobitmstar	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
1164	26.1	smileplot	Roger Newson
1165	26.0	b1x2	Jonah Gelbach
1166	26.0	eperiod	Juan M. Villa
1167	26.0	genspec	Damian Clarke
1168	26.0	lmabpgxt	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1169	26.0	lmavon	Emad Abd Elmessih Shehata
1170	26.0	mvtest	David E. Moore
1171	26.0	slist	Svend Juul, John Luke Gallup, Jens M. Lauritsen
1172	26.0	str2d	Patrick Royston
1173	25.8	spike	Joao Pedro Azevedo
1174	25.7	desmat	John Hendrickx
1175	25.7	sptobitsdm	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1176	25.7	cal	Chamara Anuranga
1177	25.7	dmerge	Fred Wolfe
1178	25.7	ellip	Anders Alexandersson
1179	25.7	genass	Neil Shephard
1180	25.7	himap	Thomas Roca
1181	25.7	isopoverty	Samuel Franco, Joao Pedro Azevedo
1182	25.7	logitcprplot	Ben Jann
1183	25.7	r2o	Mike Lacy
1184	25.7	ralpha	Eric Booth
1185	25.7	sdtest	Bill Sribney
1186	25.7	spell	Nicholas J. Cox, Richard Goldstein
1187	25.7	splitvallabels	Nick Winter, Ben Jann
1188	25.7	stpm2cif	Sally R. Hinchliffe, Paul Lambert
1189	25.7	studysi	Abdel G. Babiker
1190	25.7	summdate	Nicholas J. Cox
1191	25.7	tryem	Al Feiveson
1192	25.7	xcorplot	Nicholas J. Cox, Aurelio Tobias
1193	25.6	sptobitmstarxt	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
1194	25.5	regpar	Roger Newson
1195	25.3	bandplot	Nicholas J. Cox
1196	25.3	bugwrite	Al Feiveson, James Fiedler
1197	25.3	expl	K. Chamara Anuranga
1198	25.3	gsreg	Pablo Gluzmann, Demian Panigo
1199	25.3	hegy4	Christopher F Baum, Richard Sperling
1200	25.3	kaputil	David Harrison
1201	25.3	lmadw	Emad Abd Elmessih Shehata
1202	25.3	lmngry	Emad Abd Elmessih Shehata
1203	25.3	lmnwhite	Emad Abd Elmessih Shehata
1204	25.3	lmnwhitenl	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1205	25.3	matin4-matout4	Christopher F Baum, William Gould
1206	25.3	meloreg2	Emad Abd Elmessih Shehata
1207	25.3	mstore	Michael Blasnik
1208	25.3	p2ci	Nicola Orsini
1209	25.3	pwcorrw	Nicholas J. Cox
1210	25.3	sampsi_rho	Adrian Mander
1211	25.3	truecrypt	Matthew White
1212	25.3	univstat	Nicholas J. Cox
1213	25.2	estparm	Roger Newson
1214	25.0	apch	Louis Chauvel
1215	25.0	boxpanel	Brent McSharry
1216	25.0	bugsdat	Adrian Mander
1217	25.0	cpr	Nicholas J. Cox
1218	25.0	diagreg	Emad Abd Elmessih Shehata
1219	25.0	dseg	Ricardo Mora
1220	25.0	fulltab	Guy D. van Melle

1221	25.0	gamet	Nicola Orsini, Nicola Nante, Debora Rizzuto
1222	25.0	grcomb	Alex Gamma
1223	25.0	group_id	Robert Picard
1224	25.0	hildasetup	Francisco (Paco) Perales
1225	25.0	lmhcwnl	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
1226	25.0	pathof	Michael Barker
1227	25.0	ppssampford	Jonathan Mendelson
1228	25.0	ssizebi	Abdel G. Babiker
1229	25.0	stcstat2	Patrick Royston
1230	25.0	switch	Rodrigo Martell
1231	24.7	sptobitmstardh	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
1232	24.7	bic	Paul Millar
1233	24.7	birthsim	Stephen Cranney
1234	24.7	cfby	Ryan Knight
1235	24.7	datesum	Gary Longton
1236	24.7	effects	Michael Hills
1237	24.7	egfr	Phil Clayton
1238	24.7	erepost	Ben Jann
1239	24.7	fracdydx	Patrick Royston
1240	24.7	hetprob	William Gould
1241	24.7	icomp	Stanislav Kolenikov
1242	24.7	lablist	Roger Newson
1243	24.7	lmadurh2	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
1244	24.7	lmadw2	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1245	24.7	lmhharv	Emad Abd Elmessih Shehata
1246	24.7	predxcon	Joanne M. Garrett
1247	24.7	rocss	Matteo Bottai, Nicola Orsini
1248	24.5	singleb	Alejandro Lopez-Feldman
1249	24.5	variog	Nicholas J. Cox
1250	24.3	blinding	Jiefeng Chen
1251	24.3	bmjcip	Roger Newson
1252	24.3	contour	Adrian Mander
1253	24.3	cprplots	Ben Jann
1254	24.3	dataplink	Daniel E. Cook
1255	24.3	explist	Roger Newson
1256	24.3	finddup	Fred Wolfe
1257	24.3	hetred	Nikolaos A. Patsopoulos
1258	24.3	lmadurh	Emad Abd Elmessih Shehata
1259	24.3	lmadwnl	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1260	24.3	onemode	Zachary Neal
1261	24.3	pchipolate	Nicholas J. Cox
1262	24.3	valtovar	Johannes N. Blumenberg
1263	24.2	sptobitmstard	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
1264	24.0	adjust	Kenneth Higbee
1265	24.0	cstable	Gilles Desve, Peter Makary
1266	24.0	firstdigit	Nicholas J. Cox
1267	24.0	genstack	Gregorio Impavido
1268	24.0	gentrun	Hung-Jen Wang
1269	24.0	hbox	Nicholas J. Cox
1270	24.0	hlpedit	Daniel Klein
1271	24.0	lmabpnl	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
1272	24.0	lmadurmnl	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
1273	24.0	lmaz	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1274	24.0	lmfreg2	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel

1275	24.0	log2do2	Nick Winter
1276	24.0	lrutil	Jan Brogger
1277	24.0	mehetprob	Thomas Cornelissen
1278	24.0	metaprop_one	Marc Arbyn, Marc Aerts, Victoria Nyawira Nyaga
1279	24.0	outfixt	Austin Nichols
1280	24.0	predxcat	Joanne M. Garrett
1281	24.0	subsetplot	Nicholas J. Cox
1282	24.0	valuesof	Ben Jann
1283	23.8	isa	Masataka Harada
1284	23.8	itsp_ado	Christopher F Baum
1285	23.7	bnormpdf	Gary Longton
1286	23.7	dissim	Nicholas J. Cox
1287	23.7	ghistcum	Christopher F Baum, Nicholas J. Cox
1288	23.7	lmabp	Emad Abd Elmessih Shehata
1289	23.7	lmanlsur	Emad Abd Elmessih Shehata
1290	23.7	lmcovvar	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1291	23.7	lmhnlsur	Emad Abd Elmessih Shehata
1292	23.7	lmngryxt	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1293	23.7	lmnnlsur	Emad Abd Elmessih Shehata
1294	23.7	mdepriv	Maria Noel Pi Alperin, Philippe Van Kerm
1295	23.7	pindex	Muhammad Rashid Ansari, Chiara Mussida
1296	23.7	qap	William Simpson
1297	23.7	reorder	Nicholas J. Cox
1298	23.7	sixplot	Peter Lachenbruch
1299	23.7	sizefx	Matthew Openshaw
1300	23.7	sknor	Evangelos Kontopantelis
1301	23.7	umeta	Ben Dwamena
1302	23.4	treatoprobitsim	Christian Gregory
1303	23.3	catgraph	Nick Winter
1304	23.3	epiconf	Zhiqiang Wang
1305	23.3	ewma	Nicholas J. Cox
1306	23.3	genfreq	Nicholas J. Cox
1307	23.3	glmcorr	Nicholas J. Cox
1308	23.3	ivcheck	Anirban Basu
1309	23.3	lmavonnl	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
1310	23.3	lmcovnlsur	Emad Abd Elmessih Shehata
1311	23.3	lmharch	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
1312	23.3	lmhhpnl	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel
1313	23.3	nbstrat	Joseph Hilbe, Roberto Martinez-Espineira
1314	23.3	normtest	Herve M. Caci
1315	23.3	oglm9	Richard Williams
1316	23.3	qqplot3	Ariel Linden
1317	23.3	rasprt	Brent McSharry
1318	23.3	roblpr	Christopher F Baum, Vince Wiggins
1319	23.3	skprobit	Diallo Ibrahima Amadou
1320	23.3	smithwelch	Ben Jann
1321	23.3	split	Nicholas J. Cox
1322	23.3	stjmgraph	Michael J. Crowther
1323	23.3	xdatelist	Roger Newson
1324	23.3	xsampsi	Jan Brogger
1325	23.3	_gslope	Jeroen Weesie
1326	23.2	gnbstrat	Joseph Hilbe
1327	23.2	grnote	Michael Blasnik
1328	23.0	ccweight	Roger Newson
1329	23.0	crm	Adrian Mander
1330	23.0	dsearch	Ulrich Kohler
1331	23.0	eofplot	Nicholas J. Cox
1332	23.0	excelsave	Lars Angquist
1333	23.0	linkplot	Nicholas J. Cox
1334	23.0	lmalbnl	Emad Abd Elmessih Shehata, Sahra Khaleel A. Mickaiel

1335	23.0	majority	Nicholas J. Cox
1336	23.0	mivif	Daniel Klein
1337	23.0	pciplot	Patrick Royston
1338	23.0	rfregk	Kevin McKinney
1339	23.0	stnet	Enzo Coviello, Karri Seppa, Arun Pokhrel, Paul Dickman
1340	23.0	_gsoundex	Michael Blasnik
1341	22.8	survsim	Michael J. Crowther
1342	22.8	ie_rate	Daniel A. Powers
1343	22.7	arrowplot	Damian Clarke
1344	22.7	blist	Adrian Mander
1345	22.7	csjl	Thomas Steichen, Jens M. Lauritsen
1346	22.7	ftocci	Nicola Orsini
1347	22.7	gpreset	Roger Newson
1348	22.7	insob	Bas Straathof
1349	22.7	labmatch	Austin Nichols
1350	22.7	lmalb2	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
1351	22.7	medcouple	Catherine Vermandele, Vincenzo Verardi, Wouter Gelade
1352	22.7	mlcoint	Ken Heinecke
1353	22.7	oeratio	Brent McSharry
1354	22.7	orse	Christopher F Baum
1355	22.7	parseuas	Tobias Gummer, Joss Rossmann
1356	22.7	richness	Andreas Peichl, Thilo Schaefer
1357	22.7	spdir	Thomas Plümper, Eric Neumayer
1358	22.5	bipolate	Joe Canner
1359	22.5	gzipuse	Nikos Askitas
1360	22.3	alignedpairs	Ariel Linden
1361	22.3	dlist	Nicholas J. Cox
1362	22.3	prosperity	Oscar Barriga Cabanillas
1363	22.3	qqplot2	Nicholas J. Cox
1364	22.3	scm	Zachary Neal
1365	22.3	simirt	Jean-Benoit Hardouin
1366	22.3	strdate	Roger Newson
1367	22.2	ceq	Sean Higgins
1368	22.2	hallt-skewt	Scott Merryman, Rajesh Tharyan
1369	22.2	stpm2illd	Paul Lambert, Sally R. Hinchliffe, David A. Scott
1370	22.2	testcase	James Fiedler
1371	22.2	use10	Sergiy Radyakin
1372	22.1	heckprob2	Jerzy Mycielski
1373	22.1	surveybias	Kai Arzheimer
1374	22.0	alignedranks	Ariel Linden
1375	22.0	assertky	David Kantor
1376	22.0	avplots4	Ben Jann
1377	22.0	chardef	Roger Newson
1378	22.0	checkvar	Phil Bardsley
1379	22.0	cquantile	Nicholas J. Cox
1380	22.0	extfunnel	Michael J. Crowther
1381	22.0	gphudak	Vince Wiggins, Christopher F Baum
1382	22.0	grlogit	Jan Brogger
1383	22.0	invcise	Roger Newson
1384	22.0	kernel	Florian Chavez Juarez
1385	22.0	lgamma2	Joseph Hilbe
1386	22.0	lrmatx	Jan Brogger
1387	22.0	mog	Matt Hurst
1388	22.0	onewplot	Nicholas J. Cox
1389	22.0	predcalc	Joanne M. Garrett
1390	22.0	rca	Muhammad Rashid Ansari
1391	22.0	rctable	Adrien Bouguen
1392	22.0	repsample	Evangelos Kontopantelis
1393	22.0	rewrite	Rosa Gini
1394	22.0	stpm2cm	Paul Lambert
1395	22.0	swapval	Nicholas J. Cox
1396	22.0	twofold	Cathy Welch
1397	22.0	xtvc	Nicola Orsini, Matteo Bottai
1398	21.9	dologx	Roger Newson
1399	21.8	cpoisson	Joseph Hilbe

1400	21.8	sortrows	Jeff Arnold
1401	21.8	ptvtools	Mark Franklin, Lorenzo De Sio
1402	21.7	canon	Bill Sribney
1403	21.7	combine	Philip M Jones
1404	21.7	countmatch	Nicholas J. Cox
1405	21.7	cuentacot	George Vega Yon
1406	21.7	flower	Nicholas J. Cox, Thomas Steichen
1407	21.7	lomodrs	Tairi Room, Christopher F Baum
1408	21.7	mail	Nikos Askitas
1409	21.7	moments2	Dirk Enzmann
1410	21.7	obsdiff	Eric Booth
1411	21.7	outfix2	Nicholas J. Cox
1412	21.7	petpoisson	Alfonso Miranda
1413	21.7	ridder	Tim McGuire
1414	21.7	rq	Carlos Gradin
1415	21.7	runmixregls	George Leckie
1416	21.7	sepscatter	Nicholas J. Cox
1417	21.7	sf36fr	Jean-Benoit Hardouin
1418	21.7	sheafcoef	Maarten L. Buis
1419	21.7	strsrcs	C. P. Nelson
1420	21.7	tabxml	Richard Ryall, Jason Ferris
1421	21.7	tkdensity	Nicholas J. Cox
1422	21.7	tknz	David C. Elliott
1423	21.7	truernd	Sergiy Radyakin
1424	21.7	violin	Thomas Steichen
1425	21.7	wbull	Nicholas J. Cox
1426	21.4	ellip7	Anders Alexandersson
1427	21.3	backrasch	Jean-Benoit Hardouin
1428	21.3	barplot	Nicholas J. Cox
1429	21.3	cgroup	Roger Newson
1430	21.3	civplot	Nicholas J. Cox
1431	21.3	dotemplate	Andres Castaneda
1432	21.3	keeporder	James Feigenbaum
1433	21.3	lmabpxt	Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
1434	21.3	matamatrix	Timothy Mak
1435	21.3	miinc	Joseph N. Luchman
1436	21.3	npseries	Boris Kaiser
1437	21.3	obsofint	Ronnie Babigumira, Maarten L. Buis
1438	21.3	oprobpr	Nick Winter
1439	21.3	palette_all	Adrian Mander
1440	21.3	pgamma	Nicholas J. Cox
1441	21.3	prodvars	Roger Newson
1442	21.3	r2c	Joseph N. Luchman
1443	21.3	rcentile	Roger Newson
1444	21.3	rsort	Phil Clayton
1445	21.3	save9	Marco G. Ercolani
1446	21.3	setrngseed	William Gould, Antoine Terracol
1447	21.3	tablepc	Nicholas J. Cox
1448	21.3	wgttest	Ben Jann
1449	21.2	gb2lfit	Stephen P. Jenkins
1450	21.0	alignedsets	Ariel Linden
1451	21.0	alignmicro	Jinjing Li
1452	21.0	bynote	Roger Newson
1453	21.0	casefat	Azra Ghani, Jamie Griffin
1454	21.0	cleanlog	Lee E. Sieswerda
1455	21.0	cpyxplot	Nicholas J. Cox
1456	21.0	cureregr	Allen Buxton
1457	21.0	flex	J.M.C. Santos Silva
1458	21.0	fprank	Mamoun BenMamoun
1459	21.0	glgamma2	Joseph Hilbe
1460	21.0	grouplabs	Sergiy Radyakin
1461	21.0	gwhet	Gregorio Impavido
1462	21.0	hdquantile	Nicholas J. Cox
1463	21.0	katego	Andres L Gonzalez Rangel
1464	21.0	labelsof	Ben Jann

```
1465    21.0    longplot      Zhiqiang Wang, Nicholas J. Cox
1466    21.0    missingplot   Nicholas J. Cox
1467    21.0    povtime       Carlos Gradin
1468    21.0    qenv          Nicholas J. Cox, Maarten L. Buis
1469    21.0    raschcvt      Fred Wolfe
1470    21.0    regpred       Joanne M. Garrett
1471    21.0    sampsi_fleming  Adrian Mander
1472    21.0    textbarplot   Nicholas J. Cox
1473    21.0    trellis       Adrian Mander
1474    21.0    tsplot        Aurelio Tobias
1475    21.0    ueve          Aliaksandr Amialchuk
1476    20.9    ellip6        Anders Alexandersson
1477    20.9    spellutil     Edwin Leuven
1478    20.8    cnbreg        Joseph Hilbe
1479    20.8    gbgfit        Austin Nichols
1480    20.8    mkdat         Ulrich Kohler
1481    20.7    airnet        Zachary Neal
1482    20.7    cme           Sophia Rabe-Hesketh
1483    20.7    cnsrsig       Christopher F Baum, Vince Wiggins
1484    20.7    dummies2      Daniel Klein
1485    20.7    elife         Liu Wei
1486    20.7    gconc         Zurab Sajaia, Stanislav Kolenikov
1487    20.7    grby          Nicola Orsini, Matteo Bottai
1488    20.7    hoishapley    Alejandro Hoyos Suarez
1489    20.7    hue           Seth Lirette
1490    20.7    kdmany        Stanislav Kolenikov
1491    20.7    listmiss      Paul Millar
1492    20.7    lmadurm       Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
1493    20.7    longshape     Nicholas J. Cox
1494    20.7    lookfor_val   Daniel Klein
1495    20.7    mmsrm         Jean-Benoit Hardouin
1496    20.7    mol           Joao Pedro Azevedo
1497    20.7    multibar      Fred Wolfe
1498    20.7    newsimpact    Sune Karlsson
1499    20.7    openall       Ari Friedman
1500    20.7    orthog        Bill Sribney
1501    20.7    pdplot        Nicholas J. Cox
1502    20.7    pwmc          Daniel Klein
1503    20.7    setpoisson    Alfonso Miranda
1504    20.7    soreg         Mark Lunt
1505    20.7    stata2pajek   Gabriel Rossman
1506    20.7    summout       Andres L Gonzalez Rangel
1507    20.7    tomata        William Gould
1508    20.7    unitab        Nicola Orsini, Matteo Bottai
1509    20.7    vallab        Nicholas J. Cox
1510    20.7    vanelteren    Joseph Coveney
1511    20.5    cisd          Morten Frydenberg, Svend Juul
1512    20.5    electool      Antonio M. Jaime-Castillo
1513    20.5    ivglog        Joseph Hilbe
1514    20.5    pvw           Jonathan Bartlett
1515    20.3    affiliationexposure  Kayo Fujimoto
1516    20.3    avplot2       Nicholas J. Cox
1517    20.3    bpass         Eduard Pelz
1518    20.3    cibplot       Nicholas J. Cox
1519    20.3    factext       Roger Newson
1520    20.3    fieller       Joseph Coveney
1521    20.3    hl            Catherine Vermandele, Vincenzo Verardi, Wouter Gelade
1522    20.3    lrchg         Jan Brogger
1523    20.3    mbsens        Eric Overby, Hemang C Subramanian
1524    20.3    mypkg         Nicholas J. Cox
1525    20.3    tabcond       Nicholas J. Cox
1526    20.3    unemp         Carlos Gradin
1527    20.3    xb2pi         Nicola Orsini
1528    20.3    zinb          Jesper Sorensen
1529    20.3    lmoments      Nicholas J. Cox
```

1530	20.2	r2_mz	Dirk Enzmann
1531	20.2	pweibull	Nicholas J. Cox
1532	20.2	xrewide	Roger Newson
1533	20.1	dagumfit	Stephen P. Jenkins
1534	20.0	addinby	Roger Newson
1535	20.0	ccmpre	Mike Lacy
1536	20.0	chiplot	Thomas Steichen
1537	20.0	ciform	Roger Newson
1538	20.0	diptest	Nicholas J. Cox
1539	20.0	dolog	Roger Newson
1540	20.0	email	William Buchanan
1541	20.0	fndmtch	Desmond E. Williams, Nicholas J. Cox
1542	20.0	gevfit	David Roodman, Scott Merryman
1543	20.0	gmci	John Carlin
1544	20.0	melt	Johannes N. Blumenberg
1545	20.0	pcorrmat	Maarten L. Buis
1546	20.0	povguide	David Kantor
1547	20.0	pwcorrs	Fred Wolfe
1548	20.0	quadas	Ben Dwamena
1549	20.0	recast2	Fred Wolfe
1550	20.0	rii	David T. Robinson, Dan Blanchette
1551	20.0	skewplot	Nicholas J. Cox
1552	20.0	tihtest	Federico Belotti
1553	20.0	tpvar	Nicholas J. Cox
1554	19.9	adjksm	Isaias H. Salgado-Ugarte, Makoto Shimizu
1555	19.8	ciw	Nicholas J. Cox
1556	19.8	sproper	Austin Nichols
1557	19.8	contrast	Patrick Royston
1558	19.7	acplot	Nicholas J. Cox
1559	19.7	cflpois	Jens M. Lauritsen
1560	19.7	dagfit	Austin Nichols
1561	19.7	doubmass	Nicholas J. Cox
1562	19.7	mi_twoway	Jean-François Hamel
1563	19.7	nnipolate	Nicholas J. Cox
1564	19.7	raschpower	Myriam Blanchin, Jean-Benoit Hardouin
1565	19.7	rc2	John Hendrickx
1566	19.7	rtmci	Ariel Linden
1567	19.7	sreweight	Daniele Pacifico
1568	19.7	starjas	Enzo Coviello
1569	19.7	summvl	Jeroen Weesie
1570	19.6	mqgamma	David M. Drukker
1571	19.5	invgammafit	Nicholas J. Cox, Stephen P. Jenkins
1572	19.4	cvxhull	R. Allan Reese
1573	19.3	bsens	Hemang C Subramanian, Eric Overby
1574	19.3	bys	Jeroen Weesie
1575	19.3	dta2ddi	Minh Cong Nguyen
1576	19.3	gsum	Eric C. Hedberg
1577	19.3	irscharities	Billy Buchanan
1578	19.3	margintegrate	Christian Gregory
1579	19.3	metatrend	Pantelis Bagos
1580	19.3	mvsamp1i	David E. Moore
1581	19.3	pairdata	Richard J Williamson
1582	19.3	paragr	Roy Wada
1583	19.3	seldum	Jean Ries
1584	19.3	spseudor2	P. Wilner Jeanty
1585	19.3	svysampsi	Ariel Linden
1586	19.3	wridit	Roger Newson
1587	19.3	writepsfrag	Ryan Kessler
1588	19.3	zmap	Nicholas J. Cox
1589	19.2	nct	Thomas Steichen
1590	19.2	stjm11	Michael J. Crowther
1591	19.2	asl_norm	Maarten L. Buis
1592	19.2	hglogit	Joseph Hilbe
1593	19.2	ivgauss2	Joseph Hilbe
1594	19.2	smvcir	Charles Lindsey

1595	19.1	rfl	Dankwart Plattner
1596	19.0	backup	Andres Castaneda
1597	19.0	cns	Matthew Openshaw
1598	19.0	confall	Zhiqiang Wang
1599	19.0	grand2	Vince Wiggins
1600	19.0	hist3	Ulrich Kohler, Steffen Kuehnel
1601	19.0	inccat	Roger Newson
1602	19.0	kwstat	Florian Chavez Juarez
1603	19.0	logpred	Joanne M. Garrett
1604	19.0	mdensity	Nicholas J. Cox
1605	19.0	pagetrend	Hong Il Yoo
1606	19.0	stmix	Paul Lambert, Michael J. Crowther
1607	19.0	sunflower	W. Dale Plummer Jr., William D. Dupont
1608	19.0	xvalols	Joel Middleton, John Ternovski
1609	18.8	cprplot2	Ben Jann
1610	18.8	fastsample	Andrew Maurer
1611	18.8	hnbclg	Joseph Hilbe
1612	18.8	margprev	Roger Newson
1613	18.8	gsa	Masataka Harada
1614	18.7	blogit2	Nicholas J. Cox
1615	18.7	chaos	Nicholas J. Cox
1616	18.7	conddens	Nikolas Mittag
1617	18.7	confnd	Zhiqiang Wang
1618	18.7	couliari	Jorge Eduardo Perez Perez
1619	18.7	creplace	Roger Newson
1620	18.7	devnplot	Nicholas J. Cox
1621	18.7	frcount	Minh Cong Nguyen, Hoa Bao Nguyen
1622	18.7	gen_tail	David Kantor
1623	18.7	hcnbreg	Joseph Hilbe
1624	18.7	imputemok	Jean-Benoit Hardouin
1625	18.7	inorm	John C. Galati, John B. Carlin
1626	18.7	keyby	Roger Newson
1627	18.7	labsort	Ross Odell
1628	18.7	nysiis	Adrian Sayers
1629	18.7	nytimes	Neal Caren
1630	18.7	pvvar	Maximo Sangiacomo
1631	18.7	qn	Catherine Vermandele, Wouter Gelade, Vincenzo Verardi
1632	18.7	recode2	John Hendrickx
1633	18.7	renfiles	Lars Angquist
1634	18.7	saveresults	Ben Jann
1635	18.7	simpplot	Maarten L. Buis
1636	18.7	spechist	Alfonso Sanchez-Penalver
1637	18.7	sqr	Nicholas J. Cox
1638	18.7	stsurvdiff	Patrick Royston
1639	18.7	wdiscrim	Philippe Van Kerm
1640	18.7	xtmis	Minh Cong Nguyen
1641	18.7	ztnbp	Helmut Farbmacher
1642	18.6	efetch_tools	Daniel E. Cook
1643	18.6	propcnsreg	Maarten L. Buis
1644	18.5	geneigen	Christopher F Baum
1645	18.5	givgauss2	Joseph Hilbe
1646	18.4	lcmc	Alfonso Miranda, Sophia Rabe-Hesketh
1647	18.4	qv	Aspen Chen
1648	18.3	autolog	Ian Watson
1649	18.3	cddens	Nikolas Mittag
1650	18.3	clstop_lbt	Dirk Enzmann
1651	18.3	eclpci	Nicola Orsini
1652	18.3	fitint	Neville Verlander, André Charlett
1653	18.3	ifin	Ari Friedman
1654	18.3	marglmean	Roger Newson
1655	18.3	mlogpred	Bill Sribney
1656	18.3	nstage	Frederike Maria-Sophie Barthel, Patrick Royston, Daniel Bratton, Babak Chood
1657	18.3	pascal	Amadou Bassirou Diallo
1658	18.3	poisml	Joseph Hilbe
1659	18.3	ppplot	Nicholas J. Cox

1660	18.3	regen	Daniel Klein
1661	18.3	relrank	Ben Jann
1662	18.3	stquant	Enzo Coviello
1663	18.3	stsurvimpute	Patrick Royston
1664	18.3	surloads	Malte Hoffmann
1665	18.3	usd	Nikos Askitas
1666	18.3	varlabdef	Roger Newson
1667	18.3	xtmod	Daniel Seifert
1668	18.3	zip	Jesper Sorensen
1669	18.3	gumbelfit	Nicholas J. Cox, Stephen P. Jenkins
1670	18.2	groupcl	Paulo Guimaraes
1671	18.2	pbreg	Adrian Sayers
1672	18.2	cpoissone	Joseph Hilbe
1673	18.2	marktouse	Ben Jann
1674	18.2	zb_qrm	Eric Zbinden
1675	18.0	biplot	Ulrich Kohler
1676	18.0	bystore	David Harrison
1677	18.0	dash	Monica Daigl
1678	18.0	digits	Richard J. Atkins
1679	18.0	gipf	Adrian Mander
1680	18.0	git	Rodrigo Martell
1681	18.0	group_twoway	Aguinaldo Nogueira Maciente, Lucas Ferreira Mation
1682	18.0	hlist	Nicholas J. Cox
1683	18.0	istdize	Mario Cleves
1684	18.0	nstagebin	Daniel Bratton
1685	18.0	outseries	Christopher F Baum
1686	18.0	prolist	Chamara Anuranga
1687	18.0	qweibull	Nicholas J. Cox
1688	18.0	regresby	Nicholas J. Cox
1689	18.0	skilmack	Mark Chatfield
1690	18.0	spundir	Eric Neumayer, Thomas Plümper
1691	18.0	strgen	Nicholas J. Cox
1692	18.0	vam	Michael Stepner
1693	17.9	invgaussfit	Nicholas J. Cox, Stephen P. Jenkins
1694	17.9	rrlogit	Ben Jann
1695	17.8	cenpois	Joseph Hilbe, Dean Judson
1696	17.8	collapseunique	David Kantor
1697	17.8	zicen	Marcelo Coca Perraillon
1698	17.8	charutil	Nicholas J. Cox
1699	17.7	cdreg	Nikolas Mittag
1700	17.7	clump	Adrian Mander
1701	17.7	codci	Mamoun BenMamoun
1702	17.7	deci	Liu Wei
1703	17.7	ebrowse	Markus H. Hahn
1704	17.7	effcon	Al Feiveson
1705	17.7	hlp2winpdf	P. Wilner Jeanty
1706	17.7	hnbreg1	Joseph Hilbe
1707	17.7	intterms	Vince Wiggins
1708	17.7	kdbox	Philip B. Ender
1709	17.7	labgen	Nicholas J. Cox
1710	17.7	leanout	Nathaniel Beck
1711	17.7	minap	Stephen Soldz
1712	17.7	mrprobit	Nikolas Mittag
1713	17.7	msplot	Nicholas J. Cox
1714	17.7	npinfo	Sabrina Vogel, Sebastian Pink
1715	17.7	parmhet	Roger Newson
1716	17.7	parmparse	Roger Newson
1717	17.7	payper	Maximo Sangiacomo
1718	17.7	rct_minim	Philip Ryan
1719	17.7	rglm	Roger Newson
1720	17.7	stcstat	William Gould
1721	17.7	svypxcat	Joanne M. Garrett
1722	17.7	tabone	Roy Wada
1723	17.7	tex2col	Santiago Garriga
1724	17.7	whotdeck	Adrian Mander

1725	17.5	copycode	Daniel C. Schneider
1726	17.3	abg	Louis Chauvel
1727	17.3	bronch	Paul Walsh, Carl Mitchell
1728	17.3	cbarplot	Nicholas J. Cox
1729	17.3	chm	Daniel Klein
1730	17.3	clustpop	Paul Millar
1731	17.3	confsvy	Zhiqiang Wang
1732	17.3	datmat	Bill Sribney
1733	17.3	delta	Jean-Benoit Hardouin
1734	17.3	diffpi	Nicola Orsini
1735	17.3	doub2flt	Fred Wolfe
1736	17.3	ds5	Nicholas J. Cox
1737	17.3	episensrri	Rino Bellocco, Nicola Orsini, Sander Greenland
1738	17.3	favplots	Nicholas J. Cox
1739	17.3	gby	Zhiqiang Wang
1740	17.3	gsgroup	Roger Newson
1741	17.3	hlp2html	P. Wilner Jeanty
1742	17.3	imputeitems	Jean-Benoit Hardouin
1743	17.3	iquantile	Nicholas J. Cox
1744	17.3	lincom2	Jan Brogger
1745	17.3	ovbd	Joseph Coveney
1746	17.3	rel_clust	Dirk Enzmann
1747	17.3	sampsi_sccs	Philip Ryan
1748	17.3	sbplot	Nicholas J. Cox
1749	17.3	smwoodbury	Sam Schulhofer-Wohl
1750	17.3	statsbyfast	Michael Blasnik
1751	17.3	wclogit	Adrian Mander
1752	17.3	weathr	Neal Caren
1753	17.2	dsweight	Roger Newson
1754	17.2	tosql	Christopher F Baum
1755	17.1	switchoprobitsim	Christian Gregory
1756	17.1	jnsn	Joseph Coveney
1757	17.0	cistat	Nicholas J. Cox
1758	17.0	coranal	Philippe Van Kerm
1759	17.0	discrepancy	Brendan Halpin
1760	17.0	enlarge	Stanislav Kolenikov
1761	17.0	erate	Pavel Luengas Sierra, Damian Clarke
1762	17.0	far5	Abdel G. Babiker
1763	17.0	fedit	Nicholas J. Cox
1764	17.0	fixsort	Nicholas J. Cox
1765	17.0	gdsum	Daniel Klein
1766	17.0	grexport	Lars E. Kroll
1767	17.0	groupdist	Lutz Bornmann, Adam Ozimek
1768	17.0	hlp2pdf	Christopher F Baum
1769	17.0	insingap	Roger Newson
1770	17.0	kapprevi	Nicola Orsini, Debora Rizzuto
1771	17.0	ljs	Nicholas J. Cox
1772	17.0	mapch	Ward Vanlaar
1773	17.0	mark_changes	David Kantor
1774	17.0	nonparmde	John Ternovski, Joel Middleton
1775	17.0	odi	Johann Blauth, Monica Daigl
1776	17.0	pbeta	Nicholas J. Cox
1777	17.0	primes	Stanislav Kolenikov
1778	17.0	raschfit	Jean-Benoit Hardouin
1779	17.0	scenttest	Roger Newson
1780	17.0	sratio	Mamoun BenMamoun
1781	17.0	stak	Thomas Steichen
1782	17.0	stat2data	P. Wilner Jeanty
1783	17.0	supclust	Ben Jann
1784	17.0	tfv	Daniel Klein
1785	17.0	tpoisson	Joseph Hilbe
1786	17.0	use10save9	Lars Angquist
1787	16.9	mcqscore	E. Paul Wileyto
1788	16.8	st2openbugs	Ignacio López de Ullibarri
1789	16.8	mlboolean	Bear F. Braumoeller

1790	16.7	betaprior	Adrian Mander
1791	16.7	buckley	James Cui
1792	16.7	cmaxuse	Christopher F Baum
1793	16.7	convert_top_lines	David Kantor
1794	16.7	dashln	Michael Blasnik
1795	16.7	esli	Nicola Orsini
1796	16.7	grep	Nikos Askitas
1797	16.7	histplot	Nicholas J. Cox
1798	16.7	hutchens	Stephen P. Jenkins
1799	16.7	imputerasch	Jean-Benoit Hardouin
1800	16.7	moments	Nicholas J. Cox
1801	16.7	mvsktest	Stanislav Kolenikov
1802	16.7	polar	Joe Canner
1803	16.7	qgamma	Nicholas J. Cox
1804	16.7	reganat	Valerio Filoso
1805	16.7	romantoarabic	Nicholas J. Cox
1806	16.7	scoregrp	Paulo Guimaraes
1807	16.7	sdline	Nicholas J. Cox
1808	16.7	simsum	Ian White
1809	16.7	stgtcalc	Peter Sasieni, Patrick Royston
1810	16.7	tr	Ben Jann
1811	16.7	trimplot	Nicholas J. Cox
1812	16.7	varlab	Patrick Joly
1813	16.7	ztpflex	Helmut Farbmacher
1814	16.5	frontierhtail	Diallo Ibrahima Amadou
1815	16.5	validly	Kenneth Macdonald
1816	16.5	metapow	Sally R. Hinchliffe, Michael J. Crowther
1817	16.4	stbtcalc	Peter Sasieni, Patrick Royston
1818	16.4	stgit	Matthew White
1819	16.3	allcross	Kenneth Higbee
1820	16.3	biplotvlab	Jean-Benoit Hardouin
1821	16.3	cij	Nicholas J. Cox
1822	16.3	crater	Leah W. McGuire
1823	16.3	grand	Vince Wiggins
1824	16.3	ifwins	Dan Blanchette
1825	16.3	igencox	Rafal Raciborski
1826	16.3	metadata	Nikos Askitas
1827	16.3	miparallel	Timothy Mak
1828	16.3	mira	Rodrigo Alfaro
1829	16.3	mtad	Timothy Simcoe
1830	16.3	mvsampsi	David E. Moore
1831	16.3	nicedates	Nicholas J. Cox
1832	16.3	pwcov	Christopher F Baum
1833	16.3	pwploti	Zhiqiang Wang
1834	16.3	qbeta	Nicholas J. Cox
1835	16.3	regintfe	Paulo Guimaraes
1836	16.3	safedrop	Nicholas J. Cox
1837	16.3	sdlim	Ulrich Kohler
1838	16.3	shuffle	Ben Jann
1839	16.3	simon2stage	Adrian Mander
1840	16.3	spec_stand	Rosa Gini
1841	16.3	sssplot	Nicholas J. Cox
1842	16.3	stcoxplt	Joanne M. Garrett
1843	16.3	t2way5	Nicholas J. Cox
1844	16.3	uniquestrata	Ari Friedman
1845	16.3	writeinput	Eric Booth
1846	16.3	zcq	Johann Blauth, Monica Daigl
1847	16.3	zipsave	Henrik Stovring
1848	16.2	gmmcovearn	Olive Sweetman, Aedin Doris, Donal O´Neill
1849	16.2	nstagebinopt	Daniel Bratton
1850	16.1	meresc	Ulrich Kohler, Dirk Enzmann
1851	16.0	allsubsets	Thomas A. Trikalinos
1852	16.0	chunky8	David Elliott
1853	16.0	clustsens	Paul Millar
1854	16.0	dashgph	Nick Winter

1855	16.0	doubletofloat	David Kantor
1856	16.0	factref	Roger Newson
1857	16.0	fractileplot	Nicholas J. Cox
1858	16.0	freplace	Liu Wei
1859	16.0	genvars	Jan Brogger
1860	16.0	gphepssj	Roger Newson
1861	16.0	himatrix	Ulrich Kohler
1862	16.0	labcenswdi	P. Wilner Jeanty
1863	16.0	ltable2	Mario Cleves
1864	16.0	mmws	Ariel Linden
1865	16.0	panelthin	Nicholas J. Cox
1866	16.0	powerq	Tiago V Pereira, Nikolaos A Patsopoulos
1867	16.0	reffect	Michael W Gruszczynski
1868	16.0	reswage	John Reynolds
1869	16.0	rhsbsample	Philippe Van Kerm
1870	16.0	spspc	Eric Neumayer, Thomas Plümper
1871	16.0	stack	William Gould
1872	16.0	stcband	Enzo Coviello
1873	16.0	subsave	Roger Newson
1874	16.0	ttestplus	Ben Daniels
1875	16.0	_gapport	Ulrich Kohler
1876	15.8	trpois0	Joseph Hilbe
1877	15.8	examples	Nicholas J. Cox
1878	15.7	devr2	Sam Brilleman
1879	15.7	distan	Jose Maria Sanchez Saez
1880	15.7	fracdiff	Christopher F Baum
1881	15.7	gpfobl	Herve M. Caci
1882	15.7	hapblock	Adrian Mander
1883	15.7	hgclg	Joseph Hilbe
1884	15.7	hotvalue	Paul Millar
1885	15.7	mac_unab	Eric Booth
1886	15.7	makehlp	Adrian Mander
1887	15.7	seast	Mark S. Pearce, Richard Feltbower
1888	15.7	showgph	Nicholas J. Cox, Jan Brogger
1889	15.7	stcumh	Kim Lyngby Mikkelsen
1890	15.7	survtime	Allen Buxton
1891	15.7	svypxcon	Joanne M. Garrett
1892	15.7	swblock	Adrian Mander
1893	15.7	tgmixed	Christopher F Baum
1894	15.7	vartyp	Paul H. Bern
1895	15.7	weeklyclaims	Nikos Askitas
1896	15.4	roman	Nicholas J. Cox
1897	15.3	ggtax	Andres L Gonzalez Rangel
1898	15.3	irrepro	Nicholas J. Cox
1899	15.3	lms	Michael Blasnik
1900	15.3	longch	Tony Brady
1901	15.3	muxyplot	Nicholas J. Cox
1902	15.3	ncf	Thomas Steichen
1903	15.3	postrri	Sander Greenland, Rino Bellocco, Nicola Orsini
1904	15.3	preparation	Johannes N. Blumenberg
1905	15.3	ranova	Joseph Hilbe
1906	15.3	regtable	John Ternovski
1907	15.3	sbplot5	Nicholas J. Cox
1908	15.3	sliceplot	Nicholas J. Cox
1909	15.3	surrog	Malte Hoffmann
1910	15.3	taba	Nicholas J. Cox
1911	15.2	hpclg	Joseph Hilbe
1912	15.2	storecmd	Nicholas J. Cox
1913	15.1	lfsum	Fred Wolfe
1914	15.0	bkrosenblatt	Nicholas J. Cox
1915	15.0	cnormp	Austin Nichols
1916	15.0	for211	Patrick Royston
1917	15.0	ftree	Liu Wei
1918	15.0	gprefscode	Jan Brogger
1919	15.0	hsmode	Nicholas J. Cox

1920	15.0	joinvars	Daniel Klein
1921	15.0	labellacking	Nicholas J. Cox, Robert Picard
1922	15.0	lrseq	Zhiqiang Wang
1923	15.0	mequate	Leah McGuire
1924	15.0	mi_mvncat	Daniel Klein
1925	15.0	mstdize	Nicholas J. Cox
1926	15.0	nbfit	Nicholas J. Cox, Roberto G. Gutierrez
1927	15.0	rmanova	George M. Hoffman
1928	15.0	simpute	Adrian Mander
1929	15.0	textpad	Roger Newson
1930	15.0	tfinsert	Roger Newson
1931	14.9	margdistfit	Maarten L. Buis
1932	14.8	nruns	Nigel Smeeton, Nicholas J. Cox
1933	14.8	qog	Christoph Thewes
1934	14.8	simuped	James Cui
1935	14.8	cm2in	Ulrich Kohler
1936	14.7	dep4asm	Daniel Klein
1937	14.7	ds2	Nicholas J. Cox
1938	14.7	faam	Monica Daigl, Johann Blauth
1939	14.7	fvfix	Maximo Sangiacomo
1940	14.7	hlpdir	Nicholas J. Cox
1941	14.7	lisrelinput	Paul Millar
1942	14.7	lprplot	Bill Sribney
1943	14.7	reglike	Bill Sribney
1944	14.7	rensheet	Austin Nichols
1945	14.7	rrreg	Ben Jann
1946	14.7	sigcoef	Jun Xu
1947	14.7	smgfit	Austin Nichols
1948	14.7	spaces	Jan Brogger
1949	14.7	stbget	Nicholas J. Cox
1950	14.7	swain	John Antonakis, Nicolas Bastardoz
1951	14.7	symmetry	Mario Cleves
1952	14.7	tex_equal	Santiago Garriga
1953	14.7	umbrella	William D. Dupont, W. Dale Plummer, Jr.
1954	14.7	vlc	Austin Nichols
1955	14.7	witch	Thomas Steichen
1956	14.5	sortlistby	Ben Jann
1957	14.5	williams	Joseph Hilbe
1958	14.4	trinary	David Kantor
1959	14.3	dbmscopybatch	Amadou Bassirou Diallo
1960	14.3	doparser	George Vega Yon
1961	14.3	dta2html	P. Wilner Jeanty
1962	14.3	dta2kml	Benjamin Daniels
1963	14.3	eitc	Kerry L. Papps
1964	14.3	floattolong	David Kantor
1965	14.3	kaplansky	Nicholas J. Cox
1966	14.3	linequate	Leah McGuire
1967	14.3	mfilegr	Philip Ryan
1968	14.3	muxplot	Nicholas J. Cox
1969	14.3	pcmdif	Jean-Francois Hamel-Broza
1970	14.3	pexp	Nicholas J. Cox
1971	14.3	phenotype	James Cui
1972	14.3	pnrcheck	Nicola Orsini
1973	14.3	probitmiss	Donal O´Neill
1974	14.3	rsoort	Jean-Benoit Hardouin
1975	14.3	shuffle8	Ben Jann
1976	14.3	skbim	Evangelos Kontopantelis
1977	14.3	slideviewer	Eric Booth
1978	14.3	soepdo	Tim Stegmann
1979	14.3	sortobs	Julian Reif
1980	14.3	sto	Nicholas J. Cox
1981	14.1	nproc	Fred Wolfe, Philip Price
1982	14.1	rpnfcn	Henrik Stovring
1983	14.0	disjoint	Nicholas J. Cox
1984	14.0	forfile	Jan Brogger

1985	14.0	fractal	Paul Millar
1986	14.0	lbpower	Amado David Quezada Sanchez
1987	14.0	parttau	James Fiedler, Alan H. Feiveson
1988	14.0	popsim	Stephen Cranney
1989	14.0	shownear	Nicholas J. Cox
1990	14.0	spagg	Thomas Plümper, Eric Neumayer
1991	14.0	torats	Nicholas J. Cox, Christopher F Baum
1992	14.0	tpred	William Gould
1993	13.8	kappa2	Javier Lazaro, Javier Zamora, Victor Abraira, Alexander Zlotnik
1994	13.8	qexp	Nicholas J. Cox
1995	13.8	qrowname	Roger Newson
1996	13.8	tarow	Allen Buxton
1997	13.8	cureregr8	Allen Buxton
1998	13.7	epsigr	Henrik Stovring
1999	13.7	getfilename2	Jeff Arnold
2000	13.7	lstack	Nicholas J. Cox
2001	13.7	mnthplot	Nicholas J. Cox
2002	13.7	profhap	Adrian Mander
2003	13.7	shorth	Nicholas J. Cox
2004	13.7	tablab	Nicholas J. Cox
2005	13.7	tolerance	Peter Lachenbruch
2006	13.7	wosaddress	Lutz Bornmann, Adam Ozimek
2007	13.7	wosload	Lutz Bornmann, Adam Ozimek
2008	13.7	zipffit	Alexander Koplenig
2009	13.5	nbinreg	Joseph Hilbe
2010	13.3	diplot	Nicholas J. Cox
2011	13.3	fvprevar	Roger Newson
2012	13.3	labup	Roy Wada
2013	13.3	printgph	Jan Brogger
2014	13.3	pvfix	Maximo Sangiacomo
2015	13.3	qhapipf	Adrian Mander
2016	13.3	readlog	Jan Brogger
2017	13.3	scenreg	Maarten L. Buis
2018	13.3	spikeplt	Tony Brady, Nicholas J. Cox
2019	13.3	tdpd	Sylvain Weber
2020	13.2	psbayes	Nicholas J. Cox
2021	13.0	filei	Nicholas J. Cox
2022	13.0	mkstrsn	William Gould
2023	13.0	normalizepath	George Vega Yon
2024	13.0	ranvar	Frauke Kreuter
2025	13.0	rgroup	Ulrich Kohler
2026	13.0	vtokenize	Bill Rising
2027	12.8	trnbin0	Joseph Hilbe
2028	12.7	filtertrace	Daniel Klein
2029	12.7	fpref	Arnelyn Abdon
2030	12.7	fvvar	Maximo Sangiacomo
2031	12.7	loopplot	Nicholas J. Cox
2032	12.7	mkbilogn	Stephen P. Jenkins
2033	12.7	ndbci	Fred Wolfe
2034	12.7	tabmerge	Nicholas J. Cox
2035	12.7	tslist	Michael S. Hanson, Christopher F Baum
2036	12.7	vlgen	James Fiedler
2037	12.5	fiskfit	Maarten L. Buis, Stephen P. Jenkins
2038	12.5	valcuofon	George Vega Yon
2039	12.5	ztg	Joseph Hilbe
2040	12.3	feldti	Herve M. Caci
2041	12.3	tgraph	Patrick Royston
2042	12.3	vplplot	Nicholas J. Cox
2043	12.3	_grpos	Fred Wolfe
2044	12.2	subsim_1	Abdelkrim Araar, Paolo Verme
2045	12.0	msdirb	Roger Newson
2046	12.0	prwe	Monica Daigl
2047	12.0	qogbook	Richard Svensson
2048	12.0	vlist	David E. Moore
2049	12.0	vmerge	Joe Canner

```
2050     12.0    wtd           Henrik Stovring
2051     11.8    viewresults   Ben Jann
2052     11.4    adolist       Ben Jann, Stefan Wehrli
2053     11.3    textgph       Nick Winter
2054     11.3    torumm        Fred Wolfe
2055     11.3    varsearch     Jeff Arnold
2056     11.3    vcf           Daniel E. Cook
2057     11.3    _gvreldif     Stanislav Kolenikov
2058     11.3    m_stats       Pietro Tebaldi
2059     11.0    tcod          Mamoun BenMamoun
2060     11.0    _grmedf       Stanislav Kolenikov
2061     10.8    quantil2      Nicholas J. Cox
2062     10.7    vclose        Nicholas J. Cox
2063     10.6    nlcorr        Kristian Bernt Karlson, Ulrich Kohler
2064     10.3    fview         Ben Jann
2065     10.3    adodev        Roger Newson
2066     10.2    callsado      Daniel Klein
2067     10.0    qlognorm      Nicholas J. Cox
2068     10.0    _grprod       Philip Ryan
2069      9.9    subsim_2      Paolo Verme, Abdelkrim Araar
2070      9.8    addtex        Guy D. van Melle
2071      9.5    adotype       Nicholas J. Cox
2072      9.4    neoclassical  Samuel R. Lucas
2073      9.3    excelcol      Sergiy Radyakin
2074      9.3    vorter        Daniel Klein
2075      8.4    ldtest        Stephen Donald, Garry Barrett
2076      8.1    spmstarhxt    Sahra Khaleel A. Mickaiel, Emad Abd Elmessih Shehata
2077      4.6    reweight2     James Browne
2078      1.0    percat        Mehmet Mehmetoglu
2079      0.4    ginireg       Mark E Schaffer
2080      0.3    divcat        Dirk Enzmann
-------------------------------------------------------------------------
(Click on package name for description)
```

Parte IV

Indici

Indice analitico

*, 34
==, 33
?, 34
&, 34
_N, 83
_n, 83
|, 34
>, 33
<, 33
~ 34
~=, 33
>=, 33

abbrev, 75
about, 7
abs, 71
adoupdate, 22
aorder, 66
append, 117

betaden, 73
binomial, 73
box, 113
by, 35, 84
bysort, 36, 84

ceil, 71
chi2, 74
codebook, 54
collapse, 126
colsof, 81
compress, 42
cond, 78
contract, 132
correlate, 109

dati missing, 34, 36
decode, 90
delimit, 19
delimitatori fine comando, 19
describe, 53
destring, 90
diag, 80
dictionary, 44, 45
dir, 18
directory di lavoro, 15
drop, 66
duplicates report, 62

egen, 85
egenmore, 85
egenodd, 85
encode, 90
erase, 18
ereturn list, 152
excel, 48
exp, 72

F, 74
Fden, 74
findit, 23
finestra Review, 3
finestra Stata Command, 3
finestra Stata Results, 3
finestra Variables, 3
floor, 71
foreach, 145
format, 68
forvalues, 149
fre, 97
fsum, 96
Funioni di probabilita', 73
Funzioni di densita', 73

Funzioni matematiche, 71
Funzioni random, 75
Funzioni stringa, 75

gammaden, 74
generate, 71
global, 143
GME, 195
grubbs, 111
gsort, 65

help, 23

if, 32
in, 31
infile, 43
inlist, 78, 88
inputst, 47
inrange, 79
insheet, 43
inspect, 55
int, 71
inv, 81
invnormal, 75

keep, 66

label data, 56
label define, 57
label dir, 58
label drop, 58
label list, 58
label values, 57
label variable, 56
labelbook, 59
labutil, 61
labvalch, 58
lenght, 75
limits, 5
ln, 72
local, 143, 151
log, 21
log10, 72
long form, 127
lower, 76
ltrim, 76

macros, 143
max, 72, 86
mdy, 79
mean, 86
median, 87
merge, 119
mif2dta, 155
min, 73, 87
mkdir, 18
mmerge, 120
mode, 88
move, 66

normalden, 74
notes, 61
nullmat, 82
numlabel, 61

operatori di relazione, 33
operatori logici, 34
order, 65
outliers, 111
outputst, 47
outsheet, 48

preserve, 49
pwcorr, 110
pwd, 15

r(), 153
recast, 68
recode, 89
rename, 63
renvars, 63
replace, 88
reshape, 127
restore, 49
reverse, 76
round, 71
rowmax, 88
rowmean, 88
rowmin, 88
rowmiss, 88

rownomiss, 88
rowsd, 88
rowsof, 81
rtrim, 76

sample, 66
scalar, 144
search, 23
separatori, 30
set dp, 70
set memory, 39
set seed, 66
set varlabelpos, 4
shp2dta, 155
SO supportati, 3
sort, 65
spmap, 205
ssc, 21
strmatch, 76
subinstr, 77
subinword, 77
substr, 77
sum, 73, 88
summarize, 95
sysdir, 11

tab2, 104
table, 106
tabstat, 108
tabulate, 96
tipo variabili, 67
tostring, 91
trace, 82
trim, 76

uniform, 75
update, 21
upper, 76
use, 29, 41

versioni, 3

wide form, 127
word, 77
wordcount, 78

XML, 48
xmlsave, 48
xmluse, 48

Elenco delle figure

1 II Incontro degli Utenti di Stata, Milano, 10-11 ottobre 2005 iii

1.1 Le finestre di Stata 4

9.1 Box Plot 113

15.1 Mappa pre correzione 159
15.2 Mappa post correzione 160
15.3 Mappa con colori predefiniti 161
15.4 Mappa con colori assegnati 163

A.1 Choropleth maps 237
A.2 Choropleth maps 238
A.3 Choropleth maps 239
A.4 Choropleth maps 240
A.5 Choropleth maps 241
A.6 Choropleth maps 242
A.7 Choropleth maps 243
A.8 Choropleth maps 244
A.9 Choropleth maps 245
A.10 Choropleth maps 246
A.11 Choropleth maps 247
A.12 Choropleth maps 248
A.13 Proportional symbol maps 249
A.14 Proportional symbol maps 250
A.15 Proportional symbol maps 251
A.16 Proportional symbol maps 252
A.17 Proportional symbol maps 253
A.18 Other maps 254
A.19 Other maps 255
A.20 Other maps 256
A.21 Other maps 257
A.22 Other maps 258
A.23 Other maps 259

A.24 Other maps 260
A.25 Other maps 261
A.26 Other maps 262

Elenco delle tabelle