

Manuale di Gretl



Gnu Regression, Econometrics and Time-series Library

Allin Cottrell
Department of Economics
Wake Forest University

Cristian Rigamonti
Traduzione Italiana

Agosto 2005

Manuale di Gretl: Gnu Regression, Econometrics and Time-series Library
Allin Cottrell

Copyright © 2001-2005 Allin Cottrell

È garantito il permesso di copiare, distribuire e/o modificare questo documento seguendo i termini della *Licenza per Documentazione Libera GNU*, Versione 1.1 o ogni versione successiva pubblicata dalla Free Software Foundation (si veda <http://www.gnu.org/licenses/fdl.html>).

Sommario

1. Introduzione	1
Caratteristiche principali	1
Ringraziamenti	1
Installazione del programma	2
Linux	2
MS Windows	3
Aggiornamento	3
2. Iniziare	4
Eseguire una regressione	4
Risultati della stima	6
I menù della finestra principale	8
La barra degli strumenti di gretl	11
3. Modalità di lavoro	13
Script di comandi	13
Salvare oggetti da uno script	14
Il terminale di gretl	15
Il concetto di "sessione"	15
Introduzione	15
La tabella modelli	16
La pagina dei grafici	18
Salvare e riaprire sessioni	19
4. File di dati	20
Formato interno	20
Altri formati dei file di dati	20
Database binari	20
Accesso ai database online	20
Database RATS 4	20
Creare un file di dati da zero	21
Note comuni sui dati importati	21
Importare dati e aggiungerli	22
Usare il foglio elettronico interno	22
Estrarre dati da un database	22
Creare un file di dati nei formati interni di gretl	23
Nota aggiuntiva	23
Valori mancanti nei dati	23
Raccolte di file di dati	24
5. Funzioni speciali in genr	26
Introduzione	26
Filtri per serie storiche	26
Il filtro di Hodrick-Prescott	26
Il filtro di Baxter e King	26
Ricampionamento e bootstrapping	27
Gestione dei valori mancanti	27
Recupero di variabili interne	27
6. Creare dei sotto-campioni	29
Introduzione	29
Impostazione del campione	29
Restrizione del campione	29
Campionamento casuale	31
I comandi del menù Campione	31

7. Dati panel	32
Struttura panel	32
Variabili dummy	32
Ritardi e differenze con dati panel	33
Stima pooled	33
Esempio: la Penn World Table	34
8. Grafici e diagrammi	35
Grafici gnuplot	35
Opzioni generali di gnuplot	35
Mostrare le etichette dei dati	35
Opzioni avanzate	36
Boxplot	36
9. Minimi quadrati non lineari	38
Introduzione ed esempi	38
Inizializzazione dei parametri	38
Finestra di dialogo NLS	38
Derivate analitiche e numeriche	39
Arresto della procedura	39
Dettagli sul codice	40
Accuratezza numerica	40
10. Costrutti loop	42
Introduzione	42
Varianti di controllo del loop	42
Loop limitati	42
Loop di tipo "while"	42
Loop con indice	42
Loop di tipo "for each"	43
Loop di tipo "for"	43
La modalità "progressiva"	44
Esempi di loop	44
Esempio di procedura Monte Carlo	44
Minimi quadrati iterati	45
Esempi di loop con indice	46
11. Funzioni definite dall'utente	48
Introduzione	48
Definizione di una funzione	48
Chiamare una funzione	48
Ambito delle variabili	49
Valori di uscita	49
Controllo degli errori	50
12. Cointegrazione e modelli Vector Error Correction	51
Il test di cointegrazione di Johansen	51
13. Opzioni, argomenti e percorsi	53
gretl	53
gretlcli	54
Ricerca dei percorsi	55
MS Windows	56
14. Guida ai comandi	57
Introduzione	57
Comandi di gretl	57
add	57
addobs	57
addto	58
adf	58
append	59
ar	59

arch	59
arma	60
boxplot	60
chow	61
coeffsum	61
coint	61
coint2	62
corc	63
corr	63
corrgm	63
criteria	63
critical	64
cusum	64
data	64
delete	65
diff	65
else	65
end	65
endif	65
endloop	65
eqnprint	65
equation	65
estimate	66
fcast	66
fcasterr	67
fit	67
freq	68
function	68
garch	68
genr	69
gnuplot	72
graph	73
hausman	73
hccm	74
help	74
hilu	74
hsk	74
hurst	74
if	75
import	75
include	75
info	76
label	76
kpss	76
labels	76
lad	76
lags	77
ldiff	77
leverage	77
lmtest	78
logistic	78
logit	78
logs	79
loop	79
mahal	80
meantest	80
modeltab	80
mpols	80
multiply	81

nls.....	81
noecho.....	82
nulldata.....	82
ols.....	82
omit.....	83
omitfrom.....	83
open.....	84
outfile.....	84
panel.....	84
pca.....	84
pergm.....	85
poisson.....	85
plot.....	86
pooled.....	86
print.....	86
printf.....	87
probit.....	87
pvalue.....	87
pwe.....	88
quit.....	88
rename.....	88
reset.....	88
restrict.....	88
rhodiff.....	89
rmplot.....	89
run.....	90
runs.....	90
scatters.....	90
seed.....	90
set.....	91
setobs.....	92
setmiss.....	92
shell.....	93
sim.....	93
smpl.....	93
spearman.....	94
square.....	94
store.....	95
summary.....	95
system.....	96
tabprint.....	97
testuhat.....	97
tobit.....	97
tsls.....	97
var.....	98
varlist.....	98
vartest.....	98
vecm.....	98
vif.....	99
wls.....	100
Stimatori e test: sommario.....	100
15. Risoluzione dei problemi.....	103
Segnalazione dei bug.....	103
Programmi ausiliari.....	103
16. L'interfaccia a riga di comando.....	104
Gretl sul terminale.....	104
Differenze con ESL di Ramanathan.....	104

A. Dettagli sui file di dati	106
Formato interno di base	106
Formato tradizionale di ESL	106
Dettagli sui database binari	107
B. Note tecniche	109
C. Accuratezza numerica	110
D. Analisi econometrica avanzata con software libero	111
E. Elenco degli URL	112
Bibliografia	114

Lista delle Tabelle

4-1. Posizioni tipiche delle raccolte di file.....	24
9-1. Regressione non lineare: i test NIST	41
13-1. Impostazioni predefinite dei percorsi.....	55
14-1. Esempi di utilizzo del comando <code>genr</code>	72
14-2. Stimatori	100
14-3. Test per i modelli.....	100
14-4. Opzioni in forma lunga e corta.....	101

Capitolo 1. Introduzione

Caratteristiche principali

`gretl` è un pacchetto econometrico che comprende una libreria condivisa, un programma client a riga di comando e un'interfaccia grafica.

Amichevole

`gretl` offre un'interfaccia utente intuitiva, che permette di entrare subito nel vivo dell'analisi econometrica. Grazie all'integrazione con i libri di testo di Ramu Ramanathan, di Jeffrey Wooldridge, di James Stock e Mark Watson, il pacchetto offre molti file di dati e script di comandi, commentati e pronti all'uso.

Flessibile

È possibile scegliere il proprio metodo di lavoro preferito: dal punta-e-clicca interattivo alla modalità batch, oppure una combinazione dei due approcci.

Multi-piattaforma

La piattaforma di sviluppo di `gretl` è Linux, ma il programma è disponibile anche per MS Windows. È possibile compilarlo per Mac OS X e AIX, e dovrebbe funzionare su qualsiasi sistema operativo simile a UNIX che comprenda le librerie di base richieste (si veda l'[Appendice B](#)).

Open source

L'intero codice sorgente di `gretl` è disponibile per chiunque voglia criticarlo, correggerlo o estenderlo. L'autore accoglie con piacere qualunque segnalazione di bug.

Ragionevolmente sofisticato

`gretl` offre un'ampia varietà di stimatori basati sui minimi quadrati, compresi i minimi quadrati a due stadi e i minimi quadrati non lineari. Offre anche la stima (binomiale) logit e probit, e una struttura loop per le simulazioni Monte Carlo e le procedure di stima iterativa dei modelli non lineari. Anche se non include tutti gli stimatori e i test che potrebbero servire a un econometrico di professione, supporta l'esportazione dei dati nei formati usati da GNU R e GNU Octave consentendo così analisi più sofisticate (si veda l'[Appendice D](#)).

Accurato

`gretl` è stato testato a fondo con il dataset di riferimento NIST. Si veda l'[Appendice C](#).

Pronto per internet

`gretl` può scaricare i database da un server alla Wake Forest University. La versione MS Windows comprende un'utilità di aggiornamento che controlla se è disponibile una nuova versione del programma offrendo la possibilità di aggiornarlo automaticamente.

Internazionale

`gretl` supporta le lingue inglese, francese, italiana, spagnola e polacca, a seconda della lingua impostata sul computer.

Ringraziamenti

Per prima cosa, grazie a Ramu Ramanathan della University of California, San Diego. Qualche anno fa, è stato così gentile da fornirmi il codice sorgente del suo programma ESL ("Econometrics Software Library"), che ho portato su Linux, collaborando con lui alla manutenzione e allo sviluppo del programma. Per il progetto `gretl` ho modificato il codice originale di ESL in modo profondo: sono state aggiunte nuove funzionalità econometriche e una nuova interfaccia grafica. Ovviamente il Professor Ramanathan non è responsabile di eventuali bug presenti in `gretl`.

Sono grato agli autori di molti testi di econometria per avermi dato il permesso di creare delle versioni gretl dei dataset contenuti nei loro libri. Questa lista al momento comprende William Greene, autore di *Econometric Analysis*, Jeffrey Wooldridge (*Introductory Econometrics: A Modern Approach*); James Stock e Mark Watson (*Introduction to Econometrics*); Damodar Gujarati (*Basic Econometrics*); Russell Davidson e James MacKinnon (*Econometric Theory and Methods*).

La stima GARCH in gretl si basa sul codice pubblicato sul *Journal of Applied Econometrics* dai Prof. Fiorentini, Calzolari e Panattoni, mentre il codice per generare i p -value per i test Dickey Fuller è di James MacKinnon. In ognuno dei casi sono grato agli autori per avermi permesso di usare il loro lavoro.

Per quanto riguarda l'internazionalizzazione di gretl, vorrei ringraziare Ignacio Díaz-Empananza, Michel Robitaille, Cristian Rigamonti e Tadeusz Kufel, che hanno curato le traduzioni spagnola, francese, italiana e polacca.

Ho beneficiato largamente del lavoro di molti sviluppatori di software open-source: per i dettagli si veda l'[Appendice B](#) a questo manuale. Devo ringraziare Richard Stallman della Free Software Foundation per il suo supporto al software libero in generale, ma in particolare per aver accettato di "adottare" gretl come programma GNU.

Molti utenti di gretl hanno fornito utili suggerimenti e segnalazioni di errori. Un ringraziamento particolare a Ignacio Díaz-Empananza, Tadeusz Kufel, Pawel Kufel, Alan Isaac, Cristian Rigamonti e Dirk Eddelbuettel, che cura il pacchetto gretl per Debian GNU/Linux. Oltre agli utili suggerimenti, Riccardo "Jack" Lucchetti ha fornito contributi sostanziali in termini di codice e conoscenza econometrica.

Installazione del programma

Linux

Sulla piattaforma Linux¹, è possibile compilare da sé il codice di gretl, oppure usare un pacchetto pre-compilato. Pacchetti già pronti sono disponibili in formato rpm (appropriato per sistemi Red Hat Linux e simili) e anche in formato deb (per Debian GNU/Linux), quest'ultimo grazie a Dirk Eddelbüttel. Se si preferisce compilare da sé (o si usa un sistema UNIX per cui non sono disponibili pacchetti pre-compilati), ecco come procedere:

1. Scaricare il più recente pacchetto dei sorgenti di gretl da gretl.sourceforge.net.
2. Decomprimere il pacchetto. Se si dispone delle utilità GNU, usare il comando `tar -xvzf gretl-N.tar.gz` (sostituire N con il numero di versione specifico del file scaricato).
3. Spostarsi nella directory del codice sorgente di gretl appena creata (ad es. `gretl-1.1.5`).
4. La sequenza basilare di comandi da eseguire è:

```
./configure
make
make check
make install
```

Tuttavia, potrebbe essere utile leggere per prima cosa il file `INSTALL` e/o eseguire `./configure --help` per capire quali opzioni di configurazione sono disponibili. Un'opzione che può essere utile utilizzare è `--prefix`; il comportamento predefinito è quello di installare il programma in `/usr/local`, ma è possibile modificarlo. Ad esempio `./configure --prefix=/usr` installerà il tutto sotto la directory `/usr`. Se la procedura di configurazione si interrompe perchè una delle librerie richieste non è disponibile sul sistema, si veda l'[Appendice B](#) del manuale.

1. Nota terminologica: in questo manuale verrà usata l'abbreviazione "Linux" per riferirsi al sistema operativo GNU/Linux. Ciò che viene detto a proposito di Linux vale anche per altri sistemi simili a UNIX, anche se potrebbero essere necessari alcuni adattamenti.

A partire dalla versione 0.97, gretl supporta il desktop gnome. Per abilitare il supporto, occorre compilare il programma (come descritto sopra), mentre per disabilitare le funzionalità specifiche di gnome, occorre usare l'opzione `--without-gnome` con il comando `configure`.

MS Windows

La versione MS Windows è disponibile sotto forma di file eseguibile auto-estraente. Per installarlo, occorre scaricare `gretl_install.exe` ed eseguire questo programma. Verrà chiesta una posizione in cui installare il pacchetto (quella predefinita è `c:\userdata\gretl`).

Aggiornamento

Se si ha un computer connesso a internet, all'avvio gretl può collegarsi al proprio sito web alla Wake Forest University per vedere se sono disponibili aggiornamenti al programma. In caso positivo, comparirà una finestra informativa. Per attivare questa funzionalità, occorre abilitare la casella "Avvisa in caso di aggiornamenti di gretl" nel menù "File, Preferenze, Generali..." di gretl.

La versione MS Windows di gretl fa un passo in più: dà anche la possibilità di aggiornare automaticamente il programma. È sufficiente seguire le indicazioni nella finestra pop-up: chiudere gretl ed eseguire il programma di aggiornamento "gretl updater" (che di solito si trova vicino alla voce gretl nel gruppo Programmi del menù Avvio di Windows). Quando il programma di aggiornamento ha concluso il suo funzionamento, è possibile avviare di nuovo gretl.

Capitolo 2. Iniziare

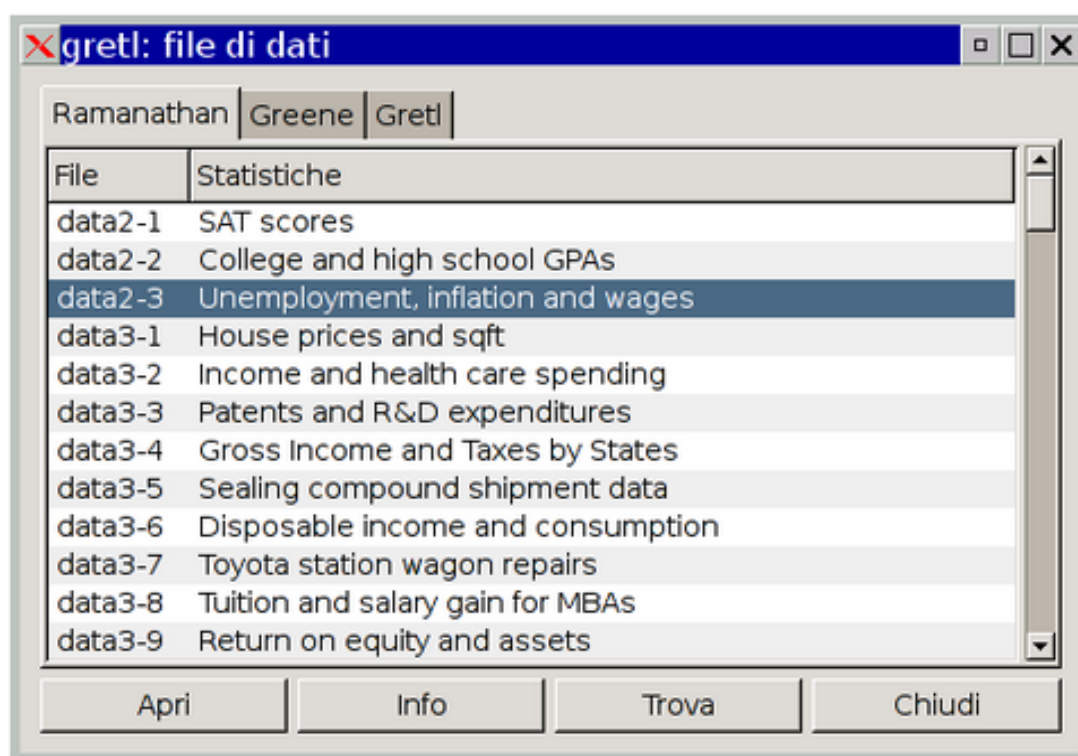
Eseguire una regressione

Questa introduzione è dedicata prevalentemente alla versione grafica del programma; si veda il [Capitolo 14](#) e il [Capitolo 16](#) per i dettagli sulla versione a riga di comando del programma, `gretlcli`.

È possibile fornire il nome di un file di dati da aprire come argomento a `gretl`, ma per il momento non facciamo: avviamo semplicemente il programma¹. Appare la finestra principale (che di solito mostra le informazioni sul dataset, ma che ora è vuota) e vari menù, alcuni dei quali disabilitati.

Cosa si può fare a questo punto? Si possono sfogliare i file di dati (o i database) forniti, aprire un file di dati, crearne uno nuovo, leggere l'aiuto in linea, o aprire un file di comandi. Per ora, sfogliamo i file di dati forniti: dal menù File, scegliamo **Apri dati**, **File di esempio**, **Ramanathan**. . . Si dovrebbe aprire una seconda finestra, che presenta un elenco dei file di dati forniti con il pacchetto (si veda la [Figura 2-1](#)). La numerazione dei file corrisponde all'organizzazione dei capitoli di Ramanathan (2002), che descrive l'analisi di questi dati, ma i dati sono utilizzabili per fare pratica anche senza avere il testo.

Figura 2-1. Finestra dei file di esempio



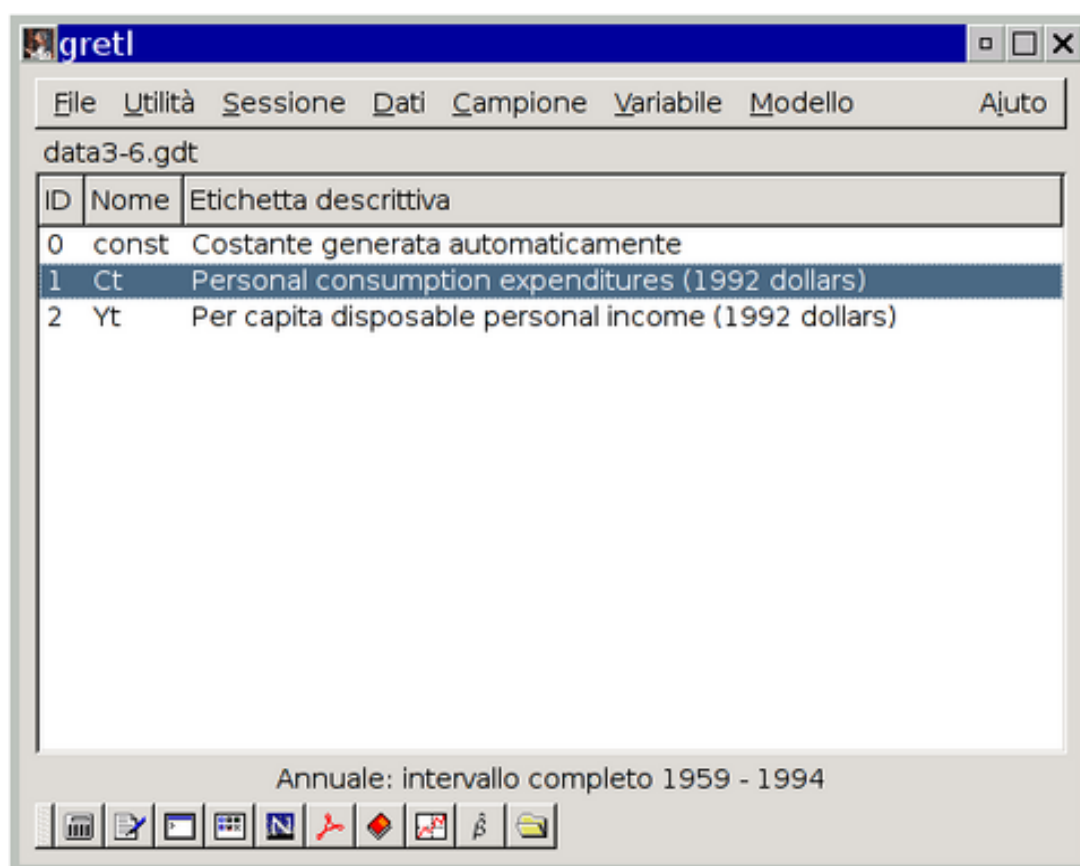
Selezionando una riga in questa finestra e facendo clic su “Info”, si aprirà una finestra di descrizione del dataset in questione (che può contenere informazioni a proposito della fonte dei dati e della definizione delle variabili). Se si trova un file interessante, è possibile aprirlo facendo clic su “Apri”, o semplicemente facendo doppio clic sul nome del file. Per il momento, apriamo `data3-6`.

1. Per comodità, in questo manuale chiamerò semplicemente `gretl` il client grafico del programma; si noti comunque che il nome specifico del programma è differente a seconda della piattaforma: su Linux si chiama `gretl_x11`, mentre su MS Windows è `gretlw32.exe`. Sui sistemi Linux viene installato anche uno script chiamato `gretl`, si veda anche il [Capitolo 13](#).

 Nelle finestre di gret1 che contengono liste, facendo doppio clic su una riga viene eseguita l'azione predefinita per la relativa voce nella lista: ad esempio mostrare i valori di una serie, o aprire un file.

Questo file contiene dati relativi a un oggetto classico dell'econometria, la funzione di consumo. La finestra dei dati dovrebbe ora contenere il nome del file di dati in uso, l'intervallo completo dei dati e quello del campione, i nomi delle variabili, insieme a delle loro brevi descrizioni (si veda la [Figura 2-2](#)).

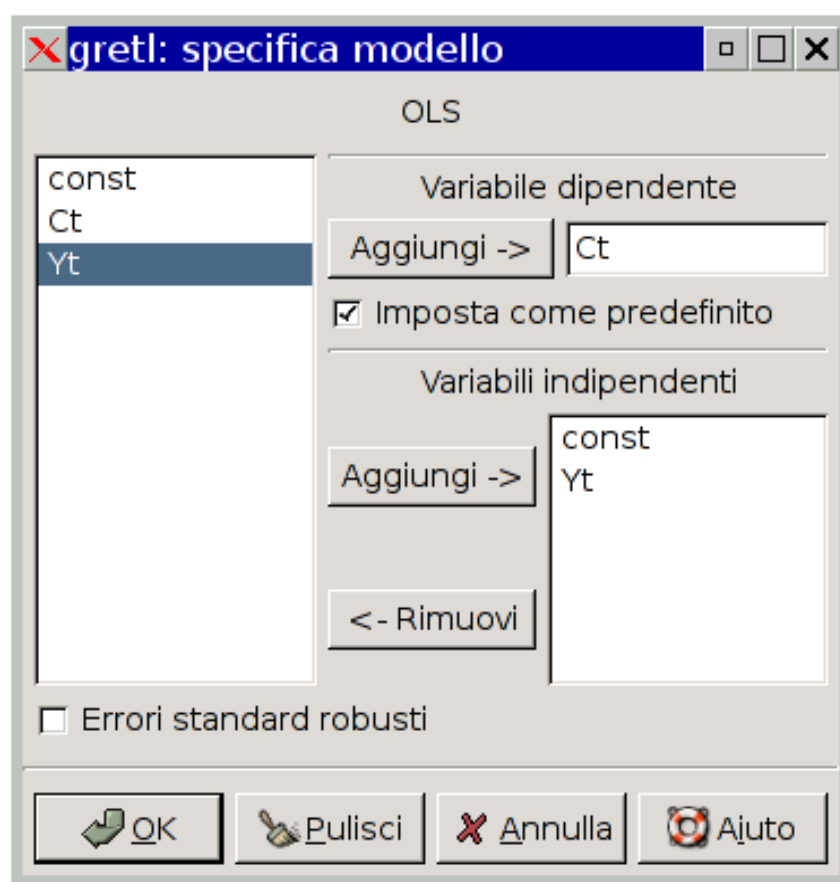
Figura 2-2. Finestra principale, con un file di esempio aperto



OK, cosa possiamo fare ora? Le varie opzioni dei menù dovrebbero essere abbastanza chiare: per ora ci concentreremo sul menù Modello, ma una panoramica di tutti i menù della finestra principale è fornita più avanti (si veda [la Sezione I menù della finestra principale](#)).

Il menù Modello di gret1 offre varie routine di stima econometrica: quella più semplice e nota è rappresentata dai minimi quadrati ordinari (Ordinary Least Squares - OLS). Scegliendo OLS, si apre una finestra di dialogo che richiede una *specificazione del modello*; si veda la [Figura 2-3](#).

Figura 2-3. Specificazione del modello



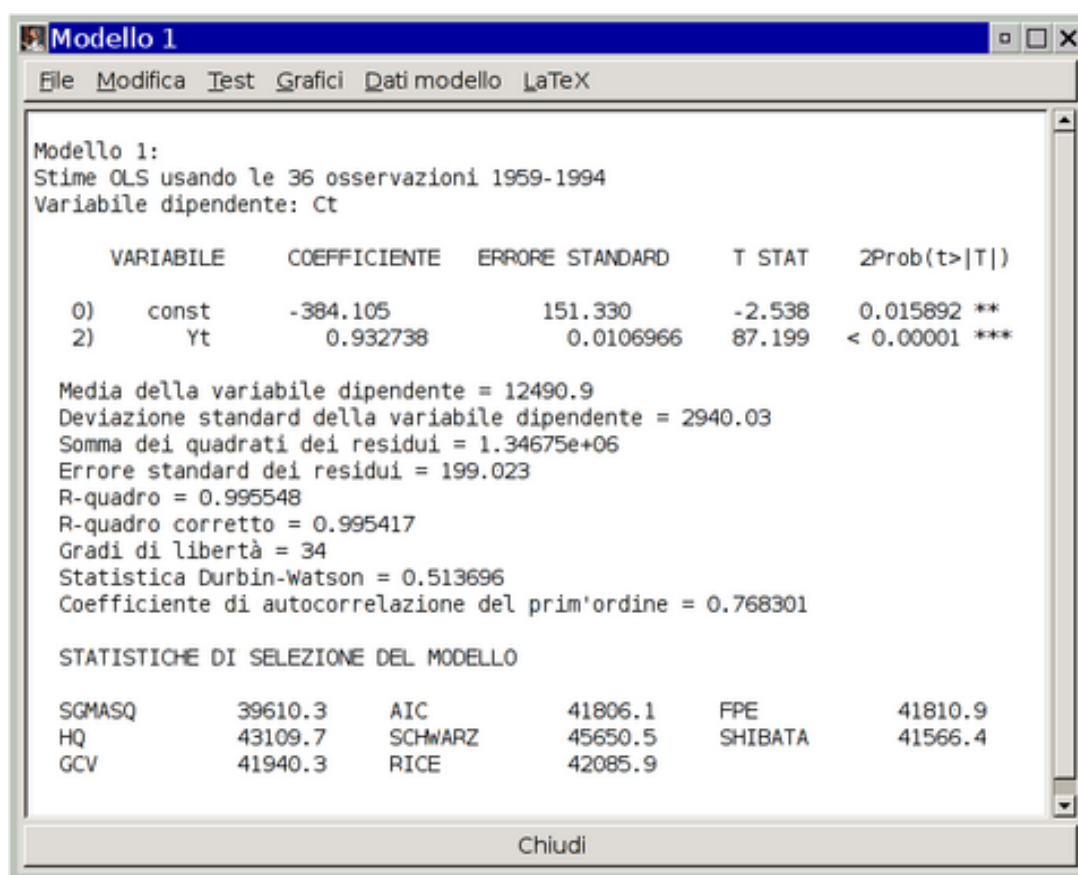
Per selezionare la variabile dipendente, fare clic su una variabile nella lista di sinistra e premere il pulsante “Scegli” con la freccia che punta verso il riquadro della variabile dipendente. Selezionando la casella “Imposta come predefinito”, la variabile scelta verrà sempre pre-selezionata come variabile dipendente durante le prossime aperture della finestra di dialogo. Trucco: facendo doppio clic su una variabile sulla sinistra, viene selezionata come variabile dipendente e impostata come scelta predefinita. Per selezionare le variabili indipendenti, fare clic su di esse nella lista di sinistra e premere il pulsante “Aggiungi” (o fare clic col pulsante destro del mouse). È possibile selezionare più variabili contigue trascinando il mouse; se le variabili da selezionare non sono contigue, occorre fare clic tenendo premuto il tasto **Ctrl**.

Per eseguire una regressione con il consumo come variabile dipendente e il reddito come variabile indipendente, fare clic su **Ct** nel riquadro della variabile dipendente e aggiungere **Yt** alla lista delle variabili indipendenti.

Risultati della stima

Una volta specificato un modello, apparirà una finestra che mostra i risultati della regressione, in un formato sufficientemente chiaro e standard (Figura 2-4).

Figura 2-4. Finestra dei risultati del modello



La finestra dei risultati contiene dei menù che consentono di ispezionare o mostrare graficamente i residui e i valori stimati, e di eseguire vari test diagnostici sul modello.

Per la maggior parte dei modelli c'è anche un'opzione per stampare il risultato della regressione in formato LaTeX. È possibile stampare i risultati in formato tabulare (simile a quello usato nella finestra dei risultati, ma con una migliore composizione tipografica) o sotto forma di equazione. Per ognuna di queste opzioni è possibile scegliere di vedere un'anteprima della pagina finale, o di salvare il risultato in un file da includere in un documento LaTeX. L'anteprima richiede che si disponga di un sistema TeX funzionante sul proprio computer. È possibile controllare l'aspetto del LaTeX generato da gretl includendo un file chiamato `gretlpre.tex` nella propria directory utente gretl (si veda [Capitolo 13](#)). Il contenuto di tale file verrà usato come "preambolo" LaTeX, il cui valore predefinito è il seguente:

```
\documentclass[11pt]{article}
\usepackage[latin1]{inputenc}
\usepackage{amsmath}
\usepackage{dcolumn,longtable}
\begin{document}
\thispagestyle{empty}
```

Si noti che sono richiesti i pacchetti `amsmath` e `dcolumn`.

Per importare i risultati di gretl in un word processor, è possibile fare copia e incolla da una finestra dei risultati usando il menù Modifica (o il pulsante Copia, in alcuni contesti) nel programma di arrivo. Molte (non tutte) finestre di gretl offrono l'opzione di copiare in formato RTF (il "Rich Text Format" di Microsoft) o come LaTeX. Se si deve incollare in un word processor, RTF può essere una buona opzione, visto che il formato tabulare dei

risultati viene preservato². In alternativa, è possibile salvare i risultati come file di testo semplice e importare successivamente il file nel programma di arrivo: quando si conclude una sessione di gret1 si ha l'opportunità di salvare tutti i risultati della sessione in un unico file.

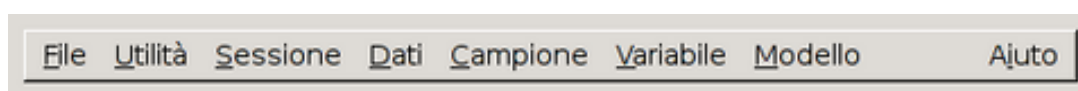
Si noti che nel desktop gnome e in MS Windows, il menù File contiene un comando per inviare i risultati direttamente a una stampante.



Quando si incollano o si importano dei risultati di gret1 sotto forma di testo semplice in un word processor, conviene selezionare un carattere a spaziatura fissa, in stile macchina da scrivere (ad es. il Courier), per preservare il formato tabulare dei risultati. Selezionare un carattere non troppo grande (Courier da 10 punti dovrebbe andare bene) eviterà che le righe dei risultati vengano spezzate nei punti sbagliati.

I menù della finestra principale

Sulla barra dei menù della finestra principale si trovano, nell'ordine da sinistra a destra, i menù File, Utilità, Sessione, Dati, Campione, Variabile, Modello e Aiuto.



§ Menù file

- Apri dati: apre un file di dati in formato interno di gret1 o lo importa da altri formati. Si veda il [Capitolo 4](#).
- Aggiungi dati: Aggiunge dati al dataset in uso, da un file di dati di gret1, un file con dati separati da virgole, o un foglio elettronico.
- Salva dati: salva il file di dati gret1 in uso.
- Salva dati come: salva il dataset in uso in formato interno, con la possibilità di usare la compressione gzip. Si veda il [Capitolo 4](#).
- Esporta dati: salva il dataset in uso in formato CSV (valori separati da virgole), o nei formati di GNU R o GNU Octave. Si veda il [Capitolo 4](#) e anche l'[Appendice D](#).
- Abbandona dataset: cancella dalla memoria il dataset in uso. Di solito questa operazione non è necessaria (visto che aprendo un nuovo file di dati, quello in uso viene sostituito), ma ci sono casi in cui è utile (si veda [la Sezione Creare un file di dati da zero nel Capitolo 4](#)).
- Consulta database: Si veda [la Sezione Database binari nel Capitolo 4](#) e [la Sezione Creare un file di dati da zero nel Capitolo 4](#).
- Crea dataset: Apre il foglio elettronico integrato per inserire dati manualmente. Si veda [la Sezione Creare un file di dati da zero nel Capitolo 4](#).
- Visualizza log comandi: Apre una finestra che contiene la registrazione dei comandi eseguiti finora.
- Apri file comandi: apre un file di comandi di gret1 creato dall'utente o uno dei file di esempio forniti con il programma. Se si intende creare un file di comandi da zero, occorre usare il comando successivo, Nuovo file comandi.
- Preferenze: imposta i percorsi per vari file a cui gret1 deve accedere; sceglie i caratteri usati per mostrare i risultati; attiva la "modalità esperto" (che sopprime la

2. Si noti che quando si copia come RTF in MS Windows, Windows permetterà di incollare il materiale solo in applicazioni che "comprendono" l'RTF. Quindi, sarà possibile incollare in MS Word, ma non nel Blocco Note. Inoltre sembra esserci un bug in alcune versioni di Windows, per cui l'operazione di copia non funziona se l'applicazione "di arrivo" (ad es. MS Word) non è stata avviata prima di copiare il materiale in questione.

visualizzazione di alcuni messaggi di avvertimento); attiva o sopprime il controllo per la disponibilità in rete di aggiornamenti di gret1; configura o disattiva la barra degli strumenti nella finestra principale. Si veda il [Capitolo 13](#) per i dettagli.

- Esci: abbandona il programma. Se non si è in modalità esperto, verrà proposto di salvare il lavoro svolto.

§ Menù utilità

- Tavole statistiche: cerca i valori critici per alcune distribuzioni di uso comune (normale o Gaussiana, t , chi-quadro, F e Durbin-Watson).
- Calcola p-value: apre una finestra che mostra i p-value per le distribuzioni Gaussiana, t , chi-quadro, F o gamma. Si veda anche il comando `pvalue` nel [Capitolo 14](#).
- Calcola test: calcola le statistiche test e i p-value per una serie di test di ipotesi di uso comune (media della popolazione, varianza e proporzione, differenza delle medie o delle varianze e proporzioni). Occorre inserire nella finestra di dialogo le necessarie statistiche del campione. Alcuni semplici test richiedono che sia indicata una serie di dati, invece che delle statistiche del campione: si veda “Differenza delle medie” e “Differenza delle varianze” nel menù Dati.
- Terminale di Gretl: apre una finestra di “terminale” in cui è possibile digitare dei comandi, come se si stesse usando la versione a riga di comando `gretlcli`, invece di quella con interfaccia grafica. Si veda il [Capitolo 14](#).
- Avvia Gnu R: Avvia R (se è presente sul sistema) e vi carica una copia del dataset in uso in gret1. Si veda l'[Appendice D](#).
- Test NIST: controlla l'accuratezza numerica di gret1 usando i test di riferimento per la regressione lineare adottati dal National Institute of Standards and Technology statunitense.

§ Menù sessione

- Visualizza icone: apre una finestra che mostra la sessione corrente di gret1 sotto forma di un insieme di icone. Per i dettagli si veda [la Sezione Il concetto di “sessione” nel Capitolo 3](#).
- Apri: apre un file di sessione salvato in precedenza.
- Salva: salva la sessione corrente in un file.
- Salva come: salva la sessione corrente in un file, scegliendone il nome.

§ Menù Dati

- Mostra valori: apre una finestra con un elenco (non modificabile) dei valori delle variabili (tutte o un sottoinsieme di esse).
- Modifica valori: apre una finestra di foglio elettronico, con cui è possibile modificare valori, aggiungere nuove variabili, o estendere il numero delle osservazioni.
- Ordina variabili: riordina l'elenco delle variabili nella finestra principale, seguendo l'ordine numerico dell'ID o quello alfabetico del nome.
- Grafico delle variabili: apre una finestra di dialogo che permette di scegliere tra un grafico temporale, un grafico a dispersione X-Y, un grafico X-Y a impulsi (barre verticali), un grafico X-Y “con fattore” (ossia, con i punti colorati in modo diverso a seconda del valore di una data variabile dummy), un boxplot e un grafico 3D. Si veda il [Capitolo 8](#) per i dettagli.
- Grafici multipli a dispersione: mostra una raccolta di grafici (al massimo sei), con una variabile sull'asse y , rappresentata rispetto a diverse variabili sull'asse x , oppure con diverse variabili sull'asse y rappresentate rispetto a una data variabile sull'asse x . Può essere utile per l'analisi esplorativa dei dati.

- Visualizza descrizione, Modifica descrizione: “Visualizza descrizione” mostra le informazioni disponibili per il file di dati in uso; “Modifica descrizione” permette di modificarle (se si ha il permesso di farlo).
- Visualizza informazioni complete: apre una finestra con una descrizione completa del dataset in uso, che include le informazioni di riepilogo e quelle specifiche di ogni variabile.
- Statistiche descrittive: mostra un insieme abbastanza ricco di statistiche descrittive per tutte le variabili del dataset, o per le variabili selezionate.
- Matrice di correlazione: mostra i coefficienti di correlazione per tutte le coppie di variabili nel dataset, o per quelle selezionate.
- Componenti principali: attivo solo se si selezionano due o più variabili; produce un'analisi delle componenti principali delle variabili selezionate.
- Distanze di Mahalanobis: attivo solo se si selezionano due o più variabili; calcola la distanza di Mahalanobis per ogni osservazione dal centroide dell'insieme di variabili selezionate.
- Differenza delle medie: calcola la statistica t per l'ipotesi nulla che le medie della popolazione siano uguali per due variabili scelte, mostrando il p-value.
- Differenza delle varianze: calcola la statistica F per l'ipotesi nulla che le varianze della popolazione siano uguali per due variabili scelte, mostrando il p-value.
- Aggiungi variabili: mostra un menù di trasformazioni standard per le variabili (logaritmi, ritardi, quadrati, ecc) che è possibile aggiungere al dataset. Dà anche l'opzione di aggiungere variabili casuali e (per i dataset di serie storiche) variabili dummy stagionali (ad es. variabili dummy trimestrali per dati trimestrali). Include la possibilità di impostare il seme del generatore di numeri pseudo-casuali.
- Aggiorna finestra: a volte i comandi di `gretl` generano nuove variabili. Questo comando sincronizza l'elenco delle variabili visualizzato nella finestra principale con l'elenco mantenuto internamente dal programma.
- Aggiungi osservazioni: Mostra una finestra di dialogo in cui è possibile scegliere un numero di osservazioni da aggiungere alla fine del dataset attuale; da usare per le previsioni.
- Rimuove le osservazioni aggiuntive: attivo solo se sono state aggiunte automaticamente delle osservazioni durante la procedura di previsione; cancella queste osservazioni aggiuntive.

§ Menù campione

- Imposta intervallo: seleziona punti di partenza e arrivo diversi per il campione in uso, all'interno dell'intervallo di dati disponibili.
- Ripristina campione completo: si spiega da sé.
- Struttura dataset: permette di modificare l'interpretazione strutturale del dataset in uso. Ad esempio, se i dati sono stati importati come cross section, è possibile fare in modo che il programma li interpreti come serie storiche o come panel. Si veda anche il [Capitolo 7](#).
- Compatta dati: per serie storiche con frequenza superiore a quella annuale, permette di diminuire la frequenza dei dati, usando uno dei quattro metodi di compattamento disponibili (media, somma, inizio del periodo, fine del periodo).
- Imposta in base a dummy: data una variabile dummy (indicatore) con valori 0 o 1, vengono scartate dal campione tutte le osservazioni per cui la variabile dummy vale 0.
- Imposta in base a condizione: simile al precedente, tranne per il fatto che non si ha bisogno di una variabile predefinita: basta fornire una condizione Booleana (ad es. `sqft > 1400`) e il campione sarà ristretto alle osservazioni che soddisfano la condizione. Si

veda la voce *genr* nel [Capitolo 14](#) per maggiori dettagli sugli operatori Booleani che possono essere usati.

- Scarta valori mancanti: scarta dal campione corrente tutte le osservazioni per cui almeno una variabile ha un valore mancante (si veda [la Sezione Valori mancanti nei dati nel Capitolo 4](#)).
- Conta valori mancanti: produce un rapporto sulle osservazioni per cui mancano dei valori. Può essere utile durante l'esame di un dataset panel, dove è abbastanza comune incontrare valori mancanti.
- Imposta codice valori mancanti: imposta il valore numerico che sarà interpretato come “mancante” o “non disponibile”.
- Aggiungi marcatori: richiede di specificare un file di testo che contiene “marcatori per le osservazioni” (brevi stringhe che identificano singole osservazioni) e aggiunge queste informazioni al dataset. Si veda il [Capitolo 4](#).
- Rimuovi marcatori: attivo solo se il dataset contiene marcatori per le osservazioni; rimuove questi marcatori.
- Ristruttura panel: permette di convertire un dataset panel da una struttura "pila di dati cross-section" a una "pila di serie storiche", o viceversa. A differenza del comando *Struttura dataset*, questo modifica effettivamente l'organizzazione dei dati.
- Trasponi dati: trasforma ogni osservazione in una variabile e viceversa (o, in altre parole, ogni riga della matrice dei dati diventa una colonna della nuova matrice dei dati); può essere utile per “raddrizzare” dati importati in modo errato.

§ **Menù variabile:** la maggior parte di questi comandi operano su una sola variabile alla volta. La variabile “attiva” viene impostata facendo clic sulla riga che la contiene nella finestra principale. La maggior parte delle opzioni si spiegano da sole. Si noti che è possibile rinominare una variabile e modificare la sua etichetta descrittiva usando “Modifica attributi”. È anche possibile definire una nuova variabile attraverso una formula (che può essere una funzione di una o più variabili esistenti). Per la sintassi di queste formule, si veda la sezione “Definisci nuova variabile” della guida in linea o la voce *genr* nel [Capitolo 14](#). Un semplice esempio:

```
pippo = x1 * x2
```

creerà la nuova variabile *pippo* come prodotto delle variabili esistenti *x1* e *x2*. In queste formule, le variabili devono essere indicate per nome, non per numero identificativo.

§ **Menù modello:** per i dettagli sui vari stimatori offerti in questo menù, si consulti [la Sezione Stimatori e test: sommario nel Capitolo 14](#) e il resto del [Capitolo 14](#), oppure il capitolo “Stima” della guida in linea. Si veda anche il [Capitolo 9](#) a proposito della stima di modelli non lineari.

§ **Menù Aiuto:** usatelo! Fornisce dettagli sulla sintassi dei comandi e delle finestre di dialogo.

La barra degli strumenti di gretl

In basso a sinistra nella finestra principale si trova la barra degli strumenti.



Le icone sulla barra hanno il seguente significato, nell'ordine:

1. Avvia una calcolatrice. Una funzione comoda quando si ha bisogno di usare velocemente una calcolatrice mentre si lavora in gretl. Il programma avviato in modo pre-

definito è `calc.exe` in MS Windows, o `xcalc` nel sistema X window. È possibile cambiare il programma nel menù File, Preferenze, Generali, sezione “Programmi”.

2. Inizia un nuovo script. Apre una finestra in cui è possibile digitare una serie di comandi da eseguire in modalità batch.
3. Apri il terminale di gretl. Una scorciatoia per il comando del menù “Terminale di Gretl” (si veda [la Sezione I menù della finestra principale](#)).
4. Apri la finestra di sessione di gretl.
5. Apri il sito web di gretl nel proprio browser (funziona solo se si è connessi a internet e si dispone di un browser).
6. Apri l'ultima versione del manuale di gretl in formato PDF. Richiede una connessione a internet e un browser in grado di gestire i file PDF.
7. Apri la guida in linea per la sintassi dei comandi (che mostra i dettagli di tutti i comandi disponibili).
8. Apri la finestra di dialogo per costruire un grafico.
9. Apri la finestra di dialogo per stimare un modello con i minimi quadrati ordinari.
10. Apri una finestra che elenca i dataset relativi al libro *Introductory Econometrics* di Ramanathan (e anche i dataset del libro di Jeffrey Wooldridge, se sono installati; si veda il [Capitolo 13](#)).

Se non si desidera visualizzare la barra degli strumenti, è possibile disabilitarla nel menù File, Preferenze, Generali, de-selezionando la casella “Mostra la barra degli strumenti di gretl”.

Capitolo 3. Modalità di lavoro

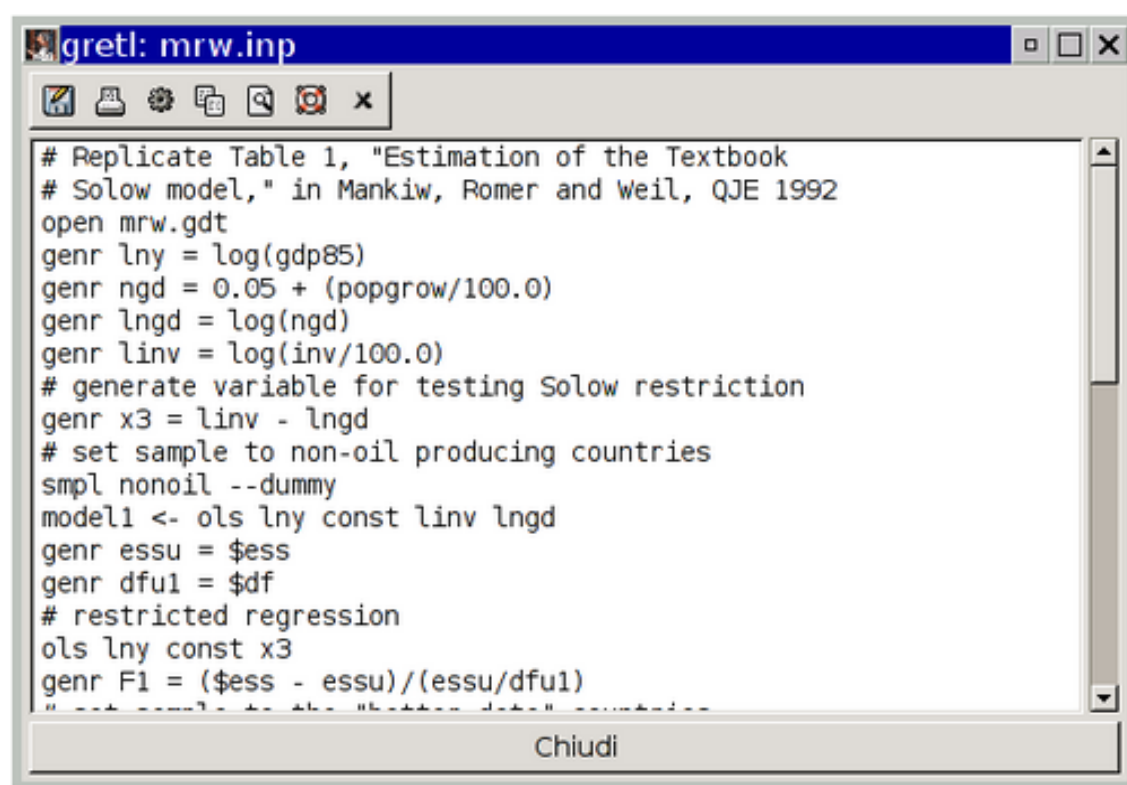
Script di comandi

I comandi eseguiti in gretl usando le finestre di dialogo dell'interfaccia grafica vengono registrati sotto forma di un file "script" o "batch". Questi file possono essere modificati e ri-eseguiti, usando gretl o l'applicazione a riga di comando gretlcli.

Per visualizzare lo stato attuale dello script durante una sessione di gretl, basta scegliere Visualizza log comandi dal menù File. Questo file di log si chiama session.inp e viene sovrascritto ogni volta che si inizia una nuova sessione: per conservarlo, basta salvarlo con un nome diverso. I file di comandi vengono visualizzati più facilmente nella finestra di selezione dei file se vengono salvati con l'estensione ".inp".

Per aprire uno script di diversa provenienza, occorre usare il comando del menù File, Apri file comandi; per creare uno script da zero, occorre usare File, Nuovo file comandi dal menù, oppure il pulsante Nuovo file comandi dalla barra degli strumenti. In entrambi i casi si aprirà una finestra comandi (si veda la [Figura 3-1](#)).

Figura 3-1. Finestra comandi, modifica di un file di comandi



La barra degli strumenti in cima alla finestra comandi offre le seguenti funzioni (da sinistra a destra): (1) Salva il file; (2) Salva il file con un nome specifico; (3) Stampa il file (solo in Windows o nel desktop gnome); (4) Esegui i comandi nel file; (5) Copia il testo selezionato; (6) Incolla il testo selezionato; (7) Cerca e sostituisci testo; (8) Annulla l'ultimo comando Incolla o Sostituisci; (9) Aiuto (spostando il cursore sulla parola di un comando e premendo il punto di domanda si ottiene aiuto su quel comando); (10) Chiudi la finestra.

Facendo clic sull'icona Esegui o scegliendo File, Esegui dal menù, i risultati sono diretti in un'unica finestra, dove possono essere modificati, salvati, o copiati negli appunti.

Per conoscere meglio le possibilità fornite dagli script, è possibile usare il comando del menù Aiuto, Comandi testuali, o eseguire la versione a riga di comando del programma

`gretlcli` e consultare il suo aiuto, oppure consultare il [Capitolo 14](#) in questo manuale. In aggiunta, il pacchetto `gretl` contiene più di 70 script “di esempio”: la maggior parte di essi sono relativi a Ramanathan (2002), ma possono essere usati in modo autonomo anche come introduzione alle possibilità di script offerte da `gretl` e ad alcuni aspetti di teoria econometrica. È possibile esplorare i file di esempio dal menù **File**, **Apri file comandi**, **File di esempio**: si troverà un elenco dei file, insieme a una breve descrizione dei problemi illustrati e dei dati utilizzati. Basta aprire un file ed eseguirlo per vederne i risultati.

Si noti che in uno script i comandi lunghi possono essere suddivisi in due o più righe, usando una barra inversa come carattere di continuazione della riga.

È anche possibile, se si vuole, usare insieme l'interfaccia grafica e le funzionalità di script, sfruttando le comodità offerte da ognuno dei due approcci. Ecco alcuni suggerimenti:

- § Aprire un file di dati dall'interfaccia grafica, esplorare i dati, generare grafici, stimare regressioni, eseguire test. Quindi aprire il log dei comandi, rimuovere eventuali comandi ridondanti, salvarlo con un nome specifico ed eseguirlo, in modo da generare un singolo file che contiene i risultati della propria sessione di lavoro.
- § Partire da un nuovo file script e scrivere i comandi necessari per eseguire le trasformazioni desiderate su un dataset (si veda il comando `genr` nel [Capitolo 14](#)). Tipicamente è possibile svolgere questo tipo di operazioni in modo più efficace scrivendo una sequenza ben ragionata di comandi, piuttosto che puntando e cliccando nell'interfaccia grafica. Salvare ed eseguire lo script: la finestra dei dati verrà aggiornata e sarà possibile continuare l'esplorazione dei dati attraverso l'interfaccia grafica. Per ripristinare lo stato iniziale dei dati in un secondo momento, è sufficiente aprire ed eseguire di nuovo lo script “preparatorio”.

Salvare oggetti da uno script

Se si stima un modello usando l'interfaccia grafica, i risultati vengono mostrati in una finestra separata, che comprende menù da cui è possibile effettuare test, disegnare grafici, salvare dati dal modello, e così via. Se invece si stima un modello usando uno script, si ottiene un tabulato non interattivo dei risultati, ma è possibile “catturare” i modelli stimati in uno script, in modo da esaminarli interattivamente dopo l'esecuzione dello script. Ecco un esempio:

```
Modello1 <- ols Ct 0 Yt
```

Ossia: viene indicato un nome con cui verrà salvato il modello, seguito da una “freccia di assegnazione” rivolta all'indietro e dal comando di stima del modello. È possibile usare spazi nei nomi dei modelli, ma occorre racchiudere il nome tra virgolette doppie:

```
"Modello 1" <- ols Ct 0 Yt
```

I modelli salvati in questo modo appariranno come icone nella finestra di sessione di `gretl` session window (si veda [la Sezione Il concetto di “sessione”](#)) dopo l'esecuzione dello script. Inoltre, è possibile fare in modo che un modello venga mostrato in una finestra in modo automatico, usando:

```
Modello1.show
```

Ancora: se il nome contiene spazi, occorre metterlo tra virgolette:

```
"Modello 1".show
```

È possibile usare la stessa procedura anche per i grafici; ad esempio, il comando seguente crea un grafico di `Ct` rispetto a `Yt`, lo salva come “Grafico” (apparirà con questo nome nella finestra di sessione) e lo mostra automaticamente:

```
Grafico <- gnuplot Ct Yt  
Grafico.show
```

È anche possibile salvare i risultati di un comando come oggetti testuali identificati da un nome (anche questi appariranno nella finestra della sessione, da cui sarà possibile aprirli in seguito). Ad esempio, questo comando invia i risultati di un test Dickey-Fuller aumentato a un “oggetto testo” chiamato ADF1 e li mostra in una finestra:

```
ADF1 <- adf 2 x1  
ADF1.show
```

Gli oggetti salvati in questo modo (siano essi modelli, grafici o parti di testo) possono essere eliminati usando il comando `.free` aggiunto al nome dell’oggetto, ad esempio con `ADF1.free`.

Il terminale di gretl

Un'altra funzionalità comoda è contenuta nel menù Utilità di gretl: il Terminale di gretl (c'è anche un pulsante Terminale di gretl nella barra degli strumenti nella finestra principale). Si tratta di una finestra in cui è possibile scrivere comandi ed eseguirli interattivamente uno alla volta (premendo il tasto Invio), così come avviene nella versione a riga di comando `gretlcli`, con la differenza che l'interfaccia grafica viene aggiornata in base ai comandi eseguiti dal terminale, permettendo di lavorare con entrambi gli strumenti.

Nel terminale è disponibile la “storia dei comandi”, ossia è possibile usare i tasti freccia su e freccia giù per scorrere la lista dei comandi già eseguiti. È possibile quindi recuperare un comando, modificarlo ed eseguirlo di nuovo.

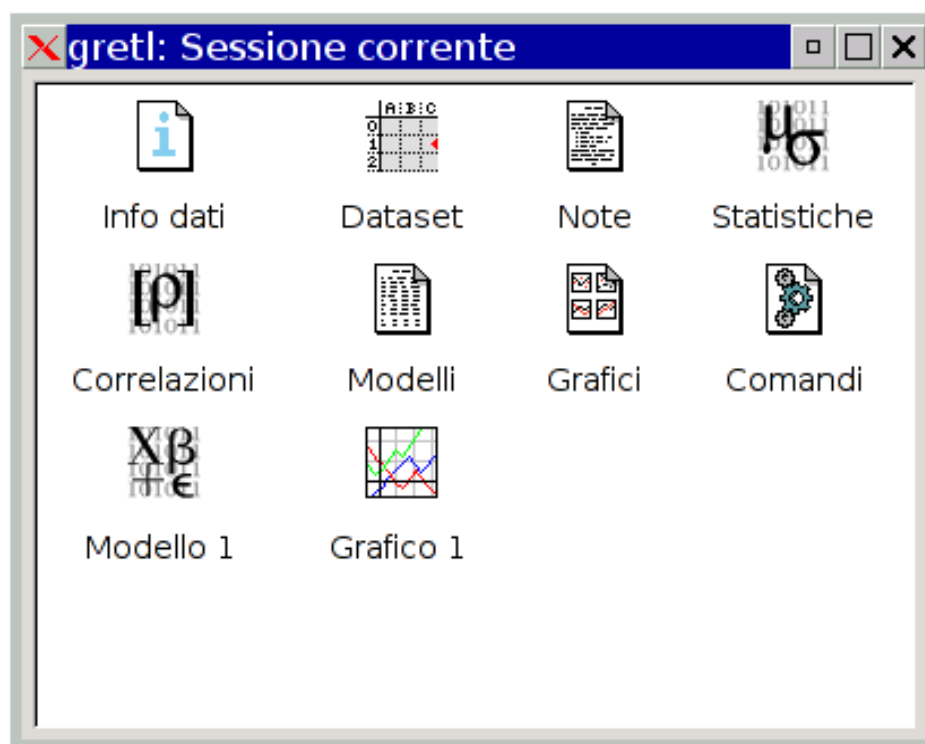
In modalità terminale, è possibile creare, visualizzare e cancellare oggetti (modelli, grafici o testo) nel modo descritto sopra per la modalità script.

Il concetto di “sessione”

Introduzione

gretl offre il concetto di “sessione” per tenere traccia del proprio lavoro e richiamarlo in un secondo momento. L'idea di base è quella di fornire uno spazio che contiene, sotto forma di icone, vari oggetti relativi alla sessione di lavoro in corso (si veda la [Figura 3-2](#)). È possibile aggiungere oggetti in questo spazio e salvarli assieme alla sessione, in modo che siano disponibili ad una successiva riapertura della sessione.

Figura 3-2. Icone di sessione: un modello e un grafico sono stati aggiunti alle icone predefinite



Avviando gretl, aprendo un dataset e selezionando Visualizza icone dal menù Sessione, è possibile visualizzare l'insieme predefinito di icone, che permettono di accedere rapidamente al log dei comandi (Comandi), alle informazioni sul dataset (se esistono), alla matrice di correlazione (Correlazioni) e alle statistiche descrittive di riepilogo (Statistiche). Tutte queste funzioni sono attivate facendo doppio clic sull'icona relativa. L'icona Dataset è un po' più complessa: un doppio clic apre i dati nel foglio di lavoro integrato, mentre facendo clic col tasto destro del mouse si ottiene un menù con altre azioni possibili.

Per aggiungere un modello alla finestra di sessione, occorre per prima cosa stimarlo usando il menù Modello, quindi aprire il menù File nella finestra del modello e selezionare Salva alla sessione come icona... o Salva come icona e chiudi. L'ultima operazione può essere eseguita semplicemente anche premendo il tasto S da dentro la finestra del modello.

Per aggiungere un grafico, occorre crearlo (dal menù Dati, Grafico delle variabili, o attraverso uno degli altri comandi gretl di generazione dei grafici). Facendo clic sulla finestra del grafico si ottiene un menù da cui si dovrà selezionare Salva alla sessione come icona.

Una volta che un modello o un grafico è stato aggiunto, la sua icona dovrebbe comparire nella finestra delle icone di sessione. Facendo doppio clic sull'icona, l'oggetto viene visualizzato di nuovo, mentre facendo clic con il tasto destro del mouse si ottiene un menù che permette di visualizzare o cancellare l'oggetto, oppure di modificarlo, se si tratta di un grafico.

La tabella modelli

Nella ricerca econometrica è prassi comune stimare vari modelli con una variabile dipendente comune, mentre i modelli differiscono tra loro per le variabili indipendenti o per lo stimatore usato. In questa situazione è comodo poter rappresentare i risultati delle regressioni sotto forma di una tabella dove ogni colonna contiene i risultati (stime dei coefficienti e errori standard) per un dato modello e ogni riga contiene le stime per una certa variabile nei differenti modelli.

Nella finestra delle icone di sessione, gretl dà la possibilità di costruire una tabella simile (e di esportarla in testo semplice, LaTeX o RTF - Rich Text Format). Ecco come fare:¹

1. Stimare un modello che si vuole includere nella tabella e selezionare, nel menù File della finestra di visualizzazione del modello, Salva alla sessione come icona o Salva come icona e chiudi.
2. Ripetere il punto 1 per gli altri modelli da includere nella tabella (fino a un massimo di sei modelli).
3. Completata la stima dei modelli, aprire l'icona della sessione di gretl (selezionando Visualizza Icone nel menù Sessione della finestra principale di gretl, o facendo clic su Icone di sessione sulla barra degli strumenti di gretl).
4. La finestra delle icone di sessione contiene un'icona chiamata **Tabella Modelli**. Per aggiungere alla tabella modelli il modello che deve apparire nella colonna più a sinistra della tabella, basta trascinare l'icona del modello sull'icona della Tabella Modelli, oppure fare clic col tasto destro sull'icona del modello e selezionare **Aggiungi alla tabella modelli** dal menù pop-up.
5. Ripetere il punto 4 per gli altri modelli da aggiungere alla tabella. Il secondo modello scelto apparirà nella seconda colonna da sinistra della tabella, e così via.
6. Ultimata la composizione della tabella, è possibile visualizzarla facendo doppio clic sulla sua icona. Per copiare la tabella negli appunti in uno dei formati supportati, basta fare clic sul menù **Modifica della finestra** in cui appare la tabella.
7. Se l'ordinamento dei modelli nella tabella non è quello voluto, fare clic col tasto destro sull'icona della tabella modelli e selezionare **Pulisci**, quindi tornare al punto 4.

Un semplice esempio di tabella modelli di gretl è mostrato in [Figura 3-3](#).

1. La tabella modelli può anche essere costruita in modo non interattivo in uno script. Per i dettagli si veda il comando [modeltab](#).

Figura 3-3. Esempio della tabella modelli

gretl: tabella modelli

stime OLS
Variabile dipendente: price

	Modello 2	Modello 3	Modello 4
const	129.1 (88.30)	121.2 (80.18)	52.35 (37.29)
sqft	0.1548** (0.03194)	0.1483** (0.02121)	0.1388** (0.01873)
bedrms	-21.59 (27.03)	-23.91 (24.64)	
baths	-12.19 (43.25)		
n	14	14	14
R**2 Corr.	0.7868	0.8046	0.8056

Errori standard tra parentesi
 * indica significatività al livello del 10 per cento
 ** indica significatività al livello del 5 per cento

Chiudi

La pagina dei grafici

L'icona Grafici della finestra di sessione offre la possibilità di riunire vari grafici da stampare su una sola pagina, se si è installato il sistema di composizione LaTeX e si è in grado di generare e visualizzare file in formato postscript²

Nella finestra della sessione, è possibile trascinare fino a otto grafici sull'icona della pagina dei grafici. Facendo doppio clic sull'icona della pagina dei grafici (o facendo clic col tasto destro e selezionando Mostra), una pagina contenente i grafici selezionati verrà composta e aperta con il proprio visualizzatore di file postscript, da cui sarà possibile stampare la pagina.

Per pulire la pagina dei grafici, fare clic col tasto destro sull'icona e selezionare Pulisci.

Su sistemi diversi da MS Windows, può essere necessario modificare l'impostazione del programma per visualizzare il postscript, attraverso la sezione Programmi della finestra di dialogo delle Preferenze di gretl (nel menù File della finestra principale). Su Windows

2. In particolare occorre avere dvips e ghostscript installati, insieme a un visualizzatore come gv, ggv o kghostview. Il visualizzatore predefinito per sistemi diversi da MS Windows è gv.

può essere necessario dover impostare le regole di associazione dei file in modo che sia usato il visualizzatore adeguato per l'azione **Apri** sui file con estensione `.ps`.

Salvare e riaprire sessioni

Se si creano modelli o grafici che si pensa di poter riutilizzare in seguito, è utile selezionare **Salva come...** dal menù **Sessione** prima di uscire da `gretl`. Per riaprire la sessione in seguito, è possibile:

- § Avviare `gretl` e riaprire il file della sessione usando il comando **Apri** dal menù **Sessione**, oppure
- § Dalla riga di comando, scrivere `gretl -r file_sessione`, dove *file_sessione* è il nome del file in cui è stata salvata la sessione.

Capitolo 4. File di dati

Formato interno

`gretl` ha un formato nativo per i file di dati. La maggior parte degli utenti probabilmente non è interessata a leggere o scrivere questi file con altri programmi che non siano `gretl`, ma in alcune occasioni potrebbe essere utile farlo: per ulteriori dettagli si veda l'[Appendice A](#).

Altri formati dei file di dati

`gretl` legge anche file in altri formati:

- § File di testo semplice (ASCII). Possono essere importati in `gretl` usando il comando “File, Apri dati, Importa ASCII...” dell’interfaccia grafica o il comando `import` dell’interfaccia a riga di comando. Per ulteriori dettagli su questo tipo di file, si veda [la Sezione Creare un file di dati da zero](#).
- § File con valori separati da virgole (CSV). Possono essere importati in `gretl` usando il comando “File, Apri dati, Importa CSV...” dell’interfaccia grafica o il comando `import` dell’interfaccia a riga di comando. Si veda anche [la Sezione Creare un file di dati da zero](#).
- § Cartelle di lavoro in formato MS Excel o Gnumeric. Possono essere importate in `gretl` con il comando “File, Apri dati, Importa”. [la Sezione Creare un file di dati da zero](#) descrive i requisiti per questo tipo di file.
- § Dati in formato BOX1. Il Data Extraction Service del Bureau of the Census statunitense offre (gratuitamente) molti dati microeconomici in questo formato. I dati BOX1 possono essere importati con il comando “File, Apri dati, Importa BOX...”, oppure con il comando `import -o`.

Quando vengono importati file in formato ASCII, CSV o BOX, `gretl` apre una finestra “diagnostica”, che informa sullo stato della lettura dei dati. Se dovessero verificarsi dei problemi a causa di dati malformattati, questa finestra mostrerà dei suggerimenti per risolverli.

Per venire incontro a chi vuole eseguire analisi più sofisticate, `gretl` offre la possibilità di salvare i dati nei formati usati dai programmi GNU R e GNU Octave (si veda l'[Appendice D](#)). Nell’interfaccia grafica, questa opzione si trova nel menù “File”, mentre nel client a riga di comando occorre usare il comando `store` con l’opzione `-r` (per R) o `-m` (per Octave).

Database binari

Per lavorare con grandi quantità di dati, `gretl` include una routine per operare su database binari. Un *database*, al contrario di un *file di dati*, non viene letto direttamente nello spazio di lavoro del programma, ma può contenere serie con frequenze e intervalli del campione diversi. È possibile aprire un database, selezionare delle serie e importarle nel dataset corrente; le serie potranno poi essere salvate in un file di dati. È possibile accedere ai database attraverso il comando “File, Consulta database”.

Per i dettagli sul formato dei database di `gretl`, si veda l'[Appendice A](#).

Accesso ai database online

Dalla versione 0.40, `gretl` è in grado di accedere ai database via internet. Alla Wake Forest University sono disponibili alcuni database, a cui è possibile accedere se il proprio computer è connesso a internet. Si veda la voce “Accesso ai database online” nel menù Aiuto di `gretl`.

Database RATS 4

Grazie a Thomas Doan di *Estima*, che mi ha fornito le specifiche del formato di database usato da RATS 4 (Regression Analysis of Time Series), gret1 è in grado di gestire anche alcuni tipi di database RATS 4: per la precisione quelli che contengono dati mensili o trimestrali. La mia università possiede il database RATS G7, che contiene dati per i sette maggiori paesi dell'OECD, e gret1 lo legge correttamente.



Per dettagli e aggiornamenti sui dati disponibili, è possibile visitare la pagina dei dati di gret1.

Creare un file di dati da zero

Ci sono cinque modi per compiere questa operazione.

1. Acquisire, o creare con un editor di testo, un file di testo semplice ed aprirlo con il comando “Importa ASCII” di gret1.
2. Usare il proprio foglio di lavoro per inserire i dati, salvarlo in formato con valori separati da virgole (Comma Separated Values) se necessario (non dovrebbe essere necessario se il programma di foglio elettronico è MS Excel o Gnumeric) e infine usare uno dei comandi “Importa” di gret1 (CSV, Excel o Gnumeric).
3. Usare il foglio elettronico contenuto in gret1.
4. Selezionare le serie di dati da un database.
5. Usare un editor di testo o altri programmi per creare un file di dati nel formato interno di gret1.

Seguono alcune note a proposito dei vari metodi presentati.

Note comuni sui dati importati

Le opzioni 1 e 2 richiedono di usare il comando “import” di gret1. Affinché i dati vengano letti correttamente, occorre che siano soddisfatte alcune condizioni:

- § La prima riga deve contenere nomi di variabile validi, ossia lunghi al massimo 8 caratteri (i nomi di variabile più lunghi verranno troncati a 8 caratteri), che iniziano con una lettera e sono composti solo da caratteri alfanumerici e dal carattere trattino basso, `_`. Precisazioni per i file ASCII o CSV: se il file non contiene righe con i nomi delle variabili, il programma userà automaticamente i nomi `v1`, `v2` e così via. Inoltre, per “prima riga” si intende la prima riga *significativa*: nel caso dei file ASCII e CSV, le righe bianche e quelle che iniziano con un carattere cancelletto, `#`, vengono ignorate. Nel caso dell'importazione di file Excel e Gnumeric, viene presentata una finestra di dialogo in cui è possibile indicare il numero di righe e/o di colonne del foglio di lavoro da ignorare.
- § I valori dei dati devono costituire un blocco rettangolare, con una variabile per colonna e un'osservazione per riga. Il numero delle variabili (colonne dei dati) deve corrispondere al numero dei nomi di variabile specificati. Si veda anche [la Sezione Valori mancanti nei dati](#). Il programma si aspetta dati di tipo numerico, ma nel caso di importazione da file ASCII/CSV, c'è un supporto limitato per dati di tipo carattere (stringa): se una colonna contiene solo dati di tipo stringa, le stringhe sono sostituite da codici numerici progressivi, e quando l'importazione si conclude, viene mostrata una tabella di corrispondenza tra stringhe e codici.
- § Date (o marcatori per le osservazioni): opzionalmente, la *prima* colonna può contenere stringhe, come date o etichette identificative per osservazioni su dati cross-section. Queste stringhe possono essere lunghe al massimo 8 caratteri (come avviene per le variabili, i nomi più lunghi verranno troncati), mentre la colonna che le ospita dovrebbe avere come nome `obs` o `date`, oppure nessun nome.

Affinché una stringa sia riconosciuta come data, deve rispettare uno dei formati seguenti: per le serie *annuali*, l'anno deve essere indicato con quattro cifre; per le serie *trimestrali* occorre indicare l'anno con quattro cifre, seguito da un separatore (punto, due punti, o la lettera Q) e da una cifra che indica il trimestre, ad esempio: 1997.1, 2002:3, 1947Q1; per le serie *mensili* occorre indicare l'anno con quattro cifre, seguito dal punto o dai due punti, e da due cifre che indicano il mese, ad esempio: 1997.01, 2002:10.

I file CSV possono usare virgole, spazi o tab come separatori fra le colonne: il separatore da usare può essere selezionato subito dopo aver eseguito il comando "Importa CSV". Se invece si usa "Importa ASCII" il programma cerca di riconoscere automaticamente il separatore usato nei dati.

Se si usa un foglio elettronico per preparare i dati, è possibile applicare varie trasformazioni ai dati "grezzi" (sommare variabili, calcolare percentuali, ecc.), ma queste elaborazioni possono essere compiute, forse più facilmente, anche in gretl, usando gli strumenti disponibili nel menù "Dati, Aggiungi variabili" e/o "Variabile, Definisci nuova variabile".

Importare dati e aggiungerli

Può essere necessario costruire un dataset di gretl a poco a poco, importando successivamente i dati da varie fonti. Questa funzionalità è fornita dai comandi del menu "File, Aggiungi dati". gretl controllerà che i nuovi dati siano compatibili con il dataset esistente e in caso positivo aggiungerà i nuovi dati. In questo modo è possibile aggiungere nuove variabili, a patto che la frequenza dei dati corrisponda a quella del dataset esistente. È anche possibile aggiungere nuove osservazioni per le serie di dati presenti nel dataset; in questo caso i nomi delle variabili devono corrispondere esattamente. Attenzione: se invece di "Aggiungi dati" si sceglie "Apri dati", il dataset corrente verrà chiuso.

Usare il foglio elettronico interno

È possibile creare un dataset con il comando "File, Crea dataset", scegliendo il tipo di dati (ad es. serie storiche trimestrali, dati cross-section), le date (o numero di osservazioni) iniziale e finale e il nome della prima variabile da creare nel dataset. Dopo aver effettuato queste scelte, viene presentato un semplice foglio elettronico in cui è possibile iniziare a inserire i valori. Facendo clic col tasto destro nella finestra del foglio elettronico, comparirà un menù che permette di aggiungere una nuova variabile (colonna), di aggiungere una nuova osservazione (aggiungere una riga in fondo al foglio), o di inserire un'osservazione nel punto indicato (i dati sottostanti saranno spostati in basso e verrà inserita una riga vuota).

Dopo aver inserito i dati nel foglio elettronico, è possibile importarli nel foglio di lavoro di gretl premendo il pulsante "Applica le modifiche" nella finestra del foglio elettronico.

Si noti che il foglio elettronico di gretl è molto semplice e non permette di inserire funzioni o formule: per trasformare i dati è possibile usare i comandi disponibili nei menù "Dati" o "Variabile" nella finestra principale di gretl.

Estrarre dati da un database

Un modo alternativo di creare un dataset consiste nel selezionare le variabili da un database. gretl include un database di serie storiche macroeconomiche relative agli USA e, come visto sopra, consente di leggere i database RATS 4.

Selezionando il comando "File, Consulta database", vengono presentate tre alternative: "gretl", "RATS 4" e "sul server di database". Selezionando "gretl", si troverà il file `fedstl.bin`, che contiene un'ampia raccolta di serie macroeconomiche USA ed è distribuito insieme al programma.

Non si troverà nulla sotto "RATS 4" a meno di non aver acquistato dei dati RATS¹. Se si

1. Si veda www.estima.com

possiedono dati RATS, occorre usare il comando “File, Preferenze, Generali...”, selezionare la finestra Database e inserire il percorso completo dei propri file RATS.

Se il proprio computer è connesso a internet è possibile accedere a vari database (alla Wake Forest University) scegliendo “sul server di database”. È possibile consultare questi database da remoto, oppure installarli sul proprio computer. La finestra dei database ha una colonna che mostra, per ogni file, lo stato di installazione e lo stato di aggiornamento della copia locale rispetto alla versione disponibile alla Wake Forest.

Dopo aver aperto un database è anche possibile importare singole serie nello spazio di lavoro di gretl usando il comando “Importa” nella finestra del database, o nel menù che compare facendo clic col tasto destro, oppure trascinando la serie nella finestra principale del programma.

Creare un file di dati nei formati interni di gretl

Se si hanno già molti dati archiviati in formato elettronico, l'approccio migliore può essere quello di creare un file di dati in uno dei formati interni di gretl, usando un editor di testo o altri programmi come awk, sed o perl. Ovviamente occorrerà studiare i formati di dati di gretl (il formato XML o quello “tradizionale”) descritti nel [Capitolo 4](#).

Nota aggiuntiva

gretl non ha problemi a compattare serie di dati ad alta frequenza (ad es. mensile) trasformandole in una frequenza più bassa (ad es. trimestrale) utilizzando uno dei 4 metodi supportati (media dei dati nel periodo, somma dei dati nel periodo, dato di inizio periodo, dato di fine periodo). Tuttavia, non c'è modo di convertire dati a bassa frequenza in alta frequenza, quindi se si intende importare serie con frequenza diversa da un database *occorre iniziare ad importare le serie con la frequenza minore*. In questo modo, il dataset di gretl verrà inizializzato con la frequenza più bassa e i dati a frequenza maggiore potranno essere importati in seguito (verranno compattati automaticamente). Se invece si inizia ad importare dati ad alta frequenza, non sarà possibile importare serie a frequenza più bassa in seguito.

Valori mancanti nei dati

I valori mancanti vengono rappresentati internamente come DBL_MAX, il più alto numero in virgola mobile rappresentabile sul sistema (che è probabile sia almeno 10 alla trecentesima potenza, e non va interpretato come un valore legittimo dei dati). Nei file di dati in formato interno vanno rappresentati come NA, mentre se si importano dati in formato CSV gretl riconosce alcuni modi comuni di rappresentare i valori mancanti: -999, la stringa NA (in maiuscolo o minuscolo), un singolo punto, o semplicemente una stringa vuota. Queste ultime, ovviamente, vanno delimitate in modo opportuno, ad es. 120.6,,5.38 indica che il valore di mezzo è mancante.

Per quanto riguarda il trattamento dei valori mancanti durante le analisi statistiche, gretl si comporta nel modo seguente:

- § Nel calcolo delle statistiche descrittive (media, deviazione standard, ecc.) con il comando `summary`, i valori mancanti sono semplicemente ignorati, e la dimensione del campione viene corretta adeguatamente.
- § Nel calcolo delle regressioni, gretl per prima cosa corregge l'inizio e la fine del campione, troncandolo dove occorre. Ad esempio, possono esserci dei valori mancanti all'inizio del campione perché la regressione comprende serie differenziate, ritardate e così via. Oppure i valori mancanti possono trovarsi alla fine del campione, a causa della compresenza di serie con diverso livello di aggiornamento, o di serie anticipate.

Se gretl trova dei valori mancanti “all'interno” dell'intervallo del campione per una regressione (che può anche essere troncato), il risultato dipende dal tipo di dataset e dallo

stimatore scelto. In molti casi, il programma eseguirà le stime saltando automaticamente le osservazioni che contengono valori mancanti, emettendo un messaggio che indica quante osservazioni sono state escluse. Tuttavia, ci sono procedure che non saltano automaticamente le osservazioni mancanti: tutti gli stimatori autoregressivi, gli stimatori di sistema (come il SUR) e i minimi quadrati non lineari. Nel caso di dati panel, l'esclusione automatica delle osservazioni mancanti avviene solo se il dataset risultante costituisce un panel "bilanciato". In tutti i casi in cui l'esclusione automatica delle osservazioni mancanti non è supportata, gretl emette un messaggio di errore e non produce stime.

In tutti i casi problematici dovuti a valori mancanti all'interno di un dataset, è possibile ricorrere alla funzione `misszero` (da usare con cautela!) del comando `genr`. Eseguendo

```
genr pippo = misszero(pluto)
```

è possibile produrre la serie `pippo`, che è identica a `pluto`, tranne per il fatto che tutti i valori mancanti sono stati trasformati in zeri. In seguito, costruendo opportunamente delle variabili dummy, sarà possibile eliminare dalla regressione le osservazioni che contengono valori mancanti, pur mantenendo lo stesso intervallo del campione².

Raccolte di file di dati

Se si usa gretl nell'attività didattica, può essere utile creare una raccolta di file di dati e/o di script di comandi, personalizzati per il proprio corso, ad uso degli studenti.

A partire dalla versione 1.2.1 di gretl, ci sono tre modi per accedere a una raccolta di file:

§ Per i file di dati: selezionare dal menù "File, Apri dati, File di esempio", o fare clic sull'icona a forma di cartella sulla barra degli strumenti di gretl.

§ Per i file di comandi: selezionare dal menù "File, Apri file comandi, File di esempio".

Quando un utente seleziona uno dei comandi visti sopra:

§ Vengono elencati automaticamente i file di dati o di comandi inclusi nella distribuzione di gretl (che comprendono i file relativi a *Introductory Econometrics* di Ramanathan e a *Econometric Analysis* di Greene).

§ Il programma cerca alcune raccolte di dati opzionali, ad esempio i file relativi ad alcuni libri di testo (Wooldridge, Gujarati, Stock e Watson) e la Penn World Table (PWT 5.6). Si veda la pagina dei dati sul sito web di gretl per ulteriori informazioni su queste raccolte. Se queste raccolte vengono trovate, vengono aggiunte all'elenco dei file disponibili.

§ Il programma infine cerca delle raccolte di dati (non necessariamente note) nei posti seguenti: la directory "di sistema" dei file di dati, la directory di sistema dei file di comandi, la directory utente e tutte le loro sotto-directory di primo livello. Valori tipici per i nomi di queste directory sono mostrati nella [Tabella 4-1.](#))

Tabella 4-1. Posizioni tipiche delle raccolte di file

	Linux	MS Windows
Directory di sistema per i dati	<code>/usr/share/gretl/data</code>	<code>c:\userdata\gretl\data</code>
Directory di sistema per i comandi	<code>/usr/share/gretl/scripts</code>	<code>c:\userdata\gretl/scripts</code>
Directory utente	<code>/home/nomeutente/gretl</code>	<code>c:\userdata\gretl\user</code>

Le raccolte trovate verranno aggiunte all'elenco dei file disponibili. In che formato deve essere una raccolta per essere riconosciuta come tale? Una raccolta può essere costituita da un gruppo di file di dati di gretl in formato XML (con l'estensione `.gdt`) o da un gruppo

2. `genr` offre anche la funzione inversa di `misszero`, ossia `zeromiss`, che sostituisce in una serie i valori zero con il codice per i valori mancanti.

di file di comandi (con l'estensione `.inp`), in entrambi i casi accompagnati da un “file principale” o catalogo. La distribuzione di `gretl` contiene vari esempi di file di catalogo, ad esempio il file `descriptions` nella sottodirectory `misc` della directory dati di `gretl` e il file `ps_descriptions` nella sottodirectory `misc` della directory dei comandi.

Se si intende aggiungere una propria raccolta, occorrerà creare dei file di catalogo, chiamati `descriptions` per i file di dati, e `ps_descriptions` per i file di comandi, nelle rispettive directory (ad es. `/usr/share/gretl/data/mydata` o `c:\userdata\gretl\data\mydata`).

La sintassi dei file di catalogo (che sono file di testo) è semplice; ecco ad esempio le prime righe del catalogo della raccolta di file di dati “misc” inclusa nella distribuzione di `gretl`:

```
# Gretl: various illustrative datafiles
"arma","artificial data for ARMA script example"
"ects_nls","Nonlinear least squares example"
"hamilton","Prices and exchange rate, U.S. and Italy"
```

La prima riga, che deve iniziare con un carattere cancelletto, contiene un nome breve, qui “Gretl”, che comparirà come etichetta identificativa per questa raccolta nella finestra di selezione dei dati, seguito da una virgola e da una descrizione breve della raccolta (opzionale).

Le righe seguenti contengono due elementi, separati da una virgola e racchiusi tra virgolette doppie. Il primo è il nome del file di dati (escludendo l'estensione `.gdt`), mentre il secondo è una breve descrizione del contenuto del file di dati. Dovrebbe esserci una riga come questa per ogni file di dati della raccolta.

I file di catalogo per le raccolte di file di comandi sono molto simili a quelli appena visti, tranne per il fatto che ogni riga del file contiene tre campi: il nome del file (senza l'estensione `.inp`), una breve descrizione del significato econometrico della serie di comandi contenuti nel file e una breve descrizione dei dati usati. Ecco un altro esempio: le prime righe del catalogo della raccolta di file di comandi “misc” inclusa nella distribuzione di `gretl`:

```
# Gretl: various sample scripts
"arma.inp","ARMA modeling","artificial data"
"ects_nls","Nonlinear least squares (Davidson)","artificial data"
"leverage","Influential observations","artificial data"
"longley","Multicollinearity","US employment"
```

La procedura per creare la propria raccolta di dati e renderla disponibile agli utenti è la seguente:

1. Assemblare i dati, nel formato più comodo.
2. Convertire i dati in formato `gretl` e salvarli come file `gdt`. Probabilmente il modo più semplice consiste nell'importare i dati nel programma come testo semplice, CSV o formato foglio elettronico (MS Excel o Gnumeric) e quindi salvarli. Può essere utile aggiungere delle descrizioni delle singole variabili (usando il comando “Variabile, Modifica attributi”) e delle informazioni sulle fonti dei dati (usando il comando “Dati, Modifica descrizione”).
3. Scrivere un file di catalogo per la raccolta, usando un editor di testi.
4. Copiare i file di dati e il file di catalogo in una sottodirectory della directory dei dati (o utente) di `gretl`.
5. Se la raccolta deve essere distribuita ad altri utenti, creare un pacchetto contenente i file di dati e il catalogo, ad esempio sotto forma di file zip.

Se la raccolta creata non contiene dati proprietari, è possibile inviarla al curatore di `gretl` in modo che venga resa disponibile a tutti gli utenti del programma come pacchetto dati opzionale.

Capitolo 5. Funzioni speciali in `genr`

Introduzione

Il comando `genr` offre un modo flessibile per definire nuove variabili. Il comando è documentato nel capitolo della guida comandi di questo manuale (si veda [genr](#)), mentre questo capitolo offre una discussione più approfondita di alcune delle funzioni speciali disponibili con `genr` e di alcune particolarità del comando.

Filtri per serie storiche

Un tipo di funzione specializzata di `genr` è il filtro per le serie storiche. Ne esistono di due tipi al momento: il filtro di Hodrick-Prescott e quello passa banda di Baxter-King. Sono utilizzabili rispettivamente con le funzioni `hpfilt()` e `bkfilt()`, che richiedono come argomento il nome della variabile da processare.

Il filtro di Hodrick-Prescott

Da scrivere.

Il filtro di Baxter e King

Si consideri la rappresentazione spettrale di una serie storica y_t :

$$y_t = \int_{-\pi}^{\pi} e^{i\omega} dZ(\omega)$$

se volessimo estrarre solo la componente di y_t che si trova tra le frequenze $\underline{\omega}$ e $\overline{\omega}$ potremmo applicare un filtro passa banda:

$$c_t^* = \int_{-\pi}^{\pi} F^*(\omega) e^{i\omega} dZ(\omega)$$

dove $F^*(\omega) = 1$ per $\underline{\omega} < |\omega| < \overline{\omega}$ e 0 altrove. Ciò implicherebbe, nel dominio temporale, applicare alla serie un filtro con un numero infinito di coefficienti, cosa non desiderabile. Il filtro passa banda di Baxter e King applica a y_t un polinomio finito nell'operatore di ritardo $A(L)$:

$$c_t = A(L)y_t$$

dove $A(L)$ è definito come

$$A(L) = \sum_{i=-k}^k a_i L^i$$

I coefficienti a_i sono scelti in modo che $F(\omega) = A(e^{i\omega})A(e^{-i\omega})$ sia la migliore approssimazione di $F^*(\omega)$ per un dato k . Chiaramente, maggiore è k , migliore è l'approssimazione, ma poiché occorre scartare $2k$ osservazioni, di solito si cerca un compromesso. Inoltre, il filtro ha altre proprietà teoriche interessanti, tra cui quella che $A(1) = 0$, quindi una serie con una sola radice unitaria è resa stazionaria con l'applicazione del filtro.

In pratica, il filtro è usato di solito con dati mensili o trimestrali per estrarne la componente di "ciclo economico", ossia la componente tra 6 e 36 trimestri. I valori usuali per k sono 8 o 12 (o forse di più per serie mensili).

I valori predefiniti per i limiti di frequenza sono 8 e 32, mentre il valore predefinito per l'ordine di approssimazione, k , è 8. È possibile impostare questi valori usando il comando `set`. La parola chiave per impostare i limiti di frequenza è `bkbp_limits`, mentre quella per k è `bkbp_k`. Quindi ad esempio, se si stanno usando dati mensili e si vuole impostare i limiti di frequenza tra 18 e 96, e k a 24, si può eseguire

```
set bkbp_limits 18 96
set bkbp_k 24
```

Questi valori resteranno in vigore per le chiamate alla funzione `bkfilt` finché non saranno modificati da un altro uso di `set`.

Ricampionamento e bootstrapping

Un'altra funzione particolare è il ricampionamento, con reimmissione, di una serie. Da scrivere.

Gestione dei valori mancanti

Sono disponibili quattro funzioni speciali per gestire i valori mancanti. La funzione booleana `missing()` richiede come unico argomento il nome di una variabile e produce una serie con valore 1 per ogni osservazione in cui la variabile indicata ha un valore mancante, 0 altrove (ossia dove la variabile indicata ha un valore valido). La funzione `ok()` è il complemento di `missing`, ossia una scorciatoia per `!missing` (dove `!` è l'operatore booleano NOT).

Ad esempio, è possibile contare i valori mancanti della variabile `x` usando

```
genr nmanc_x = sum(missing(x))
```

La funzione `zeromiss()`, che richiede anch'essa come unico argomento il nome di una serie, produce una serie in cui tutti i valori zero sono trasformati in valori mancanti. Occorre usarla con attenzione (di solito non bisogna confondere valori mancanti col valore zero), ma può essere utile in alcuni casi: ad esempio, è possibile determinare la prima osservazione valida di una variabile `x` usando

```
genr time
genr x0 = min(zeromiss(time * ok(x)))
```

La funzione `misszero()` compie l'operazione opposta di `zeromiss`, ossia converte tutti i valori mancanti in zero.

Può essere utile chiarire la propagazione dei valori mancanti all'interno delle formule di `genr`. La regola generale è che nelle operazioni aritmetiche che coinvolgono due variabili, se una delle variabili ha un valore mancante in corrispondenza dell'osservazione `t`, anche la serie risultante avrà un valore mancante in `t`. L'unica eccezione a questa regola è la moltiplicazione per zero: zero moltiplicato per un valore mancante produce sempre zero (visto che matematicamente il risultato è zero a prescindere dal valore dell'altro fattore).

Recupero di variabili interne

Il comando `genr` fornisce un modo per recuperare vari valori calcolati dal programma nel corso della stima dei modelli o della verifica di ipotesi. Le variabili che possono essere richiamate in questo modo sono elencate nella voce [genr](#) della guida comandi; qui ci occupiamo in particolare delle variabili speciali `$test` e `$pvalue`.

Queste variabili contengono, rispettivamente, il valore dell'ultima statistica `test` calcolata durante l'ultimo uso esplicito di un comando di `test` e il `p-value` per quella statistica `test`. Se non è stato eseguito alcun comando di `test`, le variabili contengono il codice di valore mancante. I "comandi espliciti di `test`" che funzionano in questo modo sono i seguenti: [add](#) (test congiunto per la significatività di variabili aggiunte a un modello); [adf](#) (test di Dickey-Fuller aumentato, si veda oltre); [arch](#) (test per ARCH); [chow](#) (test Chow per break strutturale); [coeffsum](#) (test per la somma dei coefficienti specificati); [cusum](#) (la statistica t di Harvey-Collier); [kpss](#) (il test di stazionarietà KPSS, `p-value` non disponibile); [lmtest](#) (si veda oltre); [meantest](#) (test per la differenza delle medie); [omit](#) (test congiunto per la significatività delle variabili omesse da un modello); [reset](#) (test RESET di Ramsey); [restrict](#) (vincolo lineare generale); [runs](#) (test delle successioni per la casualità); [testuhat](#) (test per la

normalità dei residui) e `vartest` (test per la differenza delle varianze). Nella maggior parte dei casi, vengono salvati valori sia in `$test` che in `$pvalue`; l'eccezione è il test KPSS, per cui non è disponibile il p-value.

Un punto da tenere in considerazione a questo proposito è che le variabili interne `$test` e `$pvalue` vengono sovrascritte ogni volta che viene eseguito uno dei test elencati sopra. Se si intende referenziare questi valori durante una sequenza di comandi `gretl`, occorre farlo nel momento giusto.

Una questione correlata è che alcuni dei comandi di test generano di solito più di una statistica test e più di un p-value: in questi casi vengono salvati solo gli ultimi valori. Per controllare in modo preciso quali valori vengono recuperati da `$test` e `$pvalue` occorre formulare il comando di test in modo che il risultato non sia ambiguo. Questa nota vale in particolare per i comandi `adf` e `lmtest`.

§ Di solito, il comando `adf` genera tre varianti del test Dickey-Fuller: una basata su una regressione che include una costante, una che include costante e trend lineare, e una che include costante e trend quadratico. Se si intende estrarre valori da `$test` o `$pvalue` dopo aver usato questo comando, è possibile selezionare la variante per cui verranno salvati i valori, usando una delle opzioni `--nc`, `--c`, `--ct` o `--ctt` con il comando `adf`.

§ Di solito, il comando `lmtest` (che deve seguire una regressione OLS) esegue vari test diagnostici sulla regressione in questione. Per controllare cosa viene salvato in `$test` e `$pvalue` occorre limitare il test usando una delle opzioni `--logs`, `--autocorr`, `--squares` o `--white`.

Un aiuto all'uso dei valori immagazzinati in `$test` e `$pvalue` è dato dal fatto che il tipo di test a cui si riferiscono questi valori viene scritto nell'etichetta descrittiva della variabile generata. Per controllare di aver recuperato il valore corretto, è possibile leggere l'etichetta con il comando `label` (il cui unico argomento è il nome della variabile). La seguente sessione interattiva illustra la procedura.

```
? adf 4 x1 --c
```

```
Test Dickey-Fuller aumentati, ordine 4, per x1
ampiezza campionaria 59
ipotesi nulla di radice unitaria: a = 1
```

```
test con costante
modello: (1 - L)y = b0 + (a-1)*y(-1) + ... + e
valore stimato di (a - 1): -0.216889
statistica test: t = -1.83491
p-value asintotico 0.3638
```

```
P-value basati su MacKinnon (JAE, 1996)
? genr pv = $pvalue
Generato lo scalare pv (ID 13) = 0.363844
? label pv
pv=Dickey-Fuller pvalue (scalar)
```

Capitolo 6. Creare dei sotto-campioni

Introduzione

Questo capitolo affronta alcune questioni correlate alla creazione di sotto-campioni in un dataset.

È possibile definire un sotto-campione per un dataset in due modi diversi, che chiameremo rispettivamente “impostazione” del campione e “restrizione” del campione.

Impostazione del campione

Per “impostazione” del campione, si intende la definizione di un campione ottenuta indicando il punto iniziale e/o quello finale dell'intervallo attuale del campione. Questa modalità è usata tipicamente con serie storiche; ad esempio se si hanno dati trimestrali per l'intervallo da 1960:1 a 2003:4 e si vuole stimare una regressione usando solo i dati degli anni '70, un comando adeguato è

```
smp1 1970:1 1979:4
```

Oppure se si vuole riservare la parte finale delle osservazioni disponibili per eseguire una previsione “fuori dal campione”, si può usare il comando

```
smp1 ; 2000:4
```

dove il punto e virgola significa “mantenere inalterata l'osservazione iniziale” (e potrebbe essere usato in modo analogo al posto del secondo parametro, indicando di mantenere inalterata l'osservazione finale). Per “inalterata” in questo caso si intende inalterata relativamente all'ultima impostazione eseguita con `smp1`, o relativamente all'intero dataset, se non è ancora stato definito alcun sotto-campione in precedenza. Ad esempio, dopo

```
smp1 1970:1 2003:4
smp1 ; 2000:4
```

l'intervallo del campione sarà da 1970:1 a 2000:4.

È possibile anche impostare l'intervallo del campione in modo incrementale o relativo: in questo caso occorre indicare per il punto iniziale e finale uno spostamento relativo, sotto forma di numero preceduto dal segno più o dal segno meno (o da un punto e virgola per indicare nessuna variazione). Ad esempio

```
smp1 +1 ;
```

sposterà in avanti di un'osservazione l'inizio del campione, mantenendo inalterata la fine del campione, mentre

```
smp1 +2 -1
```

sposterà l'inizio del campione in avanti di due osservazioni e la fine del campione indietro di una.

Una caratteristica importante dell'operazione di “impostazione del campione” descritta fin qui è che il sotto-campione creato risulta sempre composto da un insieme di osservazioni contigue. La struttura del dataset rimane quindi inalterata: se si lavora su una serie trimestrale, dopo aver impostato il campione la serie rimarrà trimestrale.

Restrizione del campione

Per “restrizione” del campione si intende la definizione di un campione ottenuta selezionando le osservazioni in base a un criterio Booleano (logico), o usando un generatore di numeri casuali. Questa modalità è usata tipicamente con dati di tipo cross-section o panel.

Si supponga di avere dei dati di tipo cross-section che descrivono il genere, il reddito e altre caratteristiche di un gruppo di individui e si vogliano analizzare solo le donne presenti nel campione. Se si dispone di una variabile dummy genere, che vale 1 per gli uomini e 0 per le donne, si potrebbe ottenere questo risultato con

```
smp1 genere=0 --restrict
```

Oppure si supponga di voler limitare il campione di lavoro ai soli individui con un reddito superiore ai 50.000 euro. Si potrebbe usare

```
smp1 reddito>50000 --restrict
```

Qui sorge un problema: eseguendo in sequenza i due comandi visti sopra, cosa conterrà il sotto-campione? Tutti gli individui con reddito superiore a 50.000 euro o solo le donne con reddito superiore a 50.000 euro? La risposta corretta è la seconda: la seconda restrizione si aggiunge alla prima, ossia la restrizione finale è il prodotto logico della nuova restrizione e di tutte le restrizioni precedenti.

Se si vuole applicare una nuova restrizione indipendentemente da quelle applicate in precedenza, occorre prima re-impostare il campione alla sua lunghezza originaria, usando

```
smp1 full
```

In alternativa, è possibile aggiungere l'opzione `replace` al comando `smp1`:

```
smp1 income>50000 --restrict --replace
```

Questa opzione ha l'effetto di re-impostare automaticamente il campione completo prima di applicare la nuova restrizione.

A differenza della semplice “impostazione” del campione, la “restrizione” del campione può produrre un insieme di osservazioni non contigue nel dataset originale e può anche modificare la struttura del dataset.

Questo fenomeno può essere osservato nel caso dei dati panel: si supponga di avere un panel di cinque imprese (indicizzate dalla variabile `impresa`) osservate in ognuno degli anni identificati dalla variabile `anno`. La restrizione

```
smp1 anno=1995 --restrict
```

produce un dataset che non è più di tipo panel, ma cross-section per l'anno 1995. In modo simile

```
smp1 impresa=3 --restrict
```

produce un dataset di serie storiche per l'impresa numero 3.

Per questi motivi (possibile non-contiguità nelle osservazioni, possibile cambiamento nella struttura dei dati) `gretl` si comporta in modo diverso a seconda che si operi una “restrizione” del campione o una semplice “impostazione” di esso. Nel caso dell'impostazione, il programma memorizza semplicemente le osservazioni iniziali e finali e le usa come parametri per i vari comandi di stima dei modelli, di calcolo delle statistiche ecc. Nel caso della restrizione, il programma crea una copia ridotta del dataset e la tratta come un sem-

plice dataset di tipo cross-section non datato.¹ Se si vuole re-imporre un'interpretazione di tipo "serie storiche" o "panel" al dataset ridotto, occorre usare il comando [setobs](#), o il comando dal menù "Campione, Struttura dataset", se appropriato).

Il fatto che una "restrizione" del campione comporti la creazione di una copia ridotta del dataset originale può creare problemi quando il dataset è molto grande (nell'ordine delle migliaia di osservazioni). Se si usano simili dataset, la creazione della copia può causare l'esaurimento della memoria del sistema durante il calcolo dei risultati delle regressioni. È possibile aggirare il problema in questo modo:

1. Aprire il dataset completo e imporre la restrizione sul campione.
2. Salvare una copia del dataset ridotto su disco.
3. Chiudere il dataset completo e aprire quello ridotto.
4. Procedere con l'analisi.

Campionamento casuale

Se si usano dataset molto grandi (o se si intende studiare le proprietà di uno stimatore), può essere utile estrarre un campione casuale dal dataset completo. È possibile farlo ad esempio con

```
smp1 100 --random
```

che seleziona 100 osservazioni. Se occorre che il campione sia riproducibile, occorre per prima cosa impostare il seme del generatore di numeri casuali, usando il comando [set](#).

Questo tipo di campionamento è un esempio di "restrizione" del campione: viene infatti generata una copia ridotta del dataset.

I comandi del menù Campione

Gli esempi visti finora hanno mostrato il comando testuale [smpl](#), ma è possibile creare un sotto-campione usando i comandi del menù Campione nell'interfaccia grafica del programma.

I comandi del menù permettono di ottenere gli stessi risultati delle varianti del comando testuale `smp1`, con la seguente eccezione: se si usa il comando "Campione, Imposta in base a condizione..." e sul dataset è già stato impostato un sotto-campione, viene data la possibilità di preservare la restrizione già attiva o di sostituirla (in modo analogo a quanto avviene invocando l'opzione `replace` descritta nella [Sezione Restrizione del campione](#)).

1. Con una eccezione: se si parte da un dataset panel bilanciato e la restrizione è tale da preservare la struttura di panel bilanciato (ad esempio perché implica la cancellazione di tutte le osservazioni per una unità cross-section), allora il dataset ridotto è ancora trattato come panel.

Capitolo 7. Dati panel

Struttura panel

I dati panel possono essere visti sotto tre dimensioni, ossia le variabili, le unità cross-section e i periodi temporali. Per rappresentarli in un file testuale (e anche per poterli manipolare), queste tre dimensioni devono in qualche modo essere riportate a due. Questa procedura di “appiattimento” richiede di prendere degli “strati” di dati che appartenerebbero alla terza dimensione e di impilarli nella dimensione verticale del file.

Gret1 si aspetta sempre di trovare dati organizzati “per osservazione”, ossia in modo che ogni riga rappresenti un’osservazione (e che ogni variabile occupi esattamente una colonna). Alla luce di questo fatto, l’appiattimento dei dati panel può essere realizzato in due modi:

§ Pila di dati cross section: ognuno dei blocchi di dati disposti verticalmente contiene i valori per tutte le unità cross-section (longitudinali) in un determinato periodo.

§ Pila di serie storiche: ognuno dei blocchi di dati disposti verticalmente contiene serie storiche per una determinata unità cross-section.

È possibile usare entrambi i metodi. Il menù Campione di gret1 contiene il comando “Ristruttura panel”, che permette di passare dalla disposizione “pila di dati cross-section” a quella “pila di serie storiche” e viceversa.

Quando si importano dati panel in gret1 da un foglio di calcolo o da un file con valori separati da virgole, la struttura panel non verrà riconosciuta automaticamente (molto probabilmente i dati verranno trattati come “non datati”). Per imporre un’interpretazione panel ai dati, è possibile procedere in due modi.

1. Usando il comando dell’interfaccia grafica “Campione, Struttura dataset”. Nella prima finestra di dialogo occorre selezionare “Panel”; in quella successiva, occorre indicare se i dati sono organizzati sotto forma di pila di serie storiche o di dati cross section. Nella finestra di dialogo successiva occorre indicare il numero di unità cross section nel dataset, infine è possibile controllare se la specificazione dei dati mostrata è corretta e confermare l’operazione.
2. Usando il comando testuale `setobs`. Per i dati panel, la sintassi del comando è `setobs freq 1:1 struttura`, dove *freq* indica la “dimensione dei blocchi” di dati (ossia, il numero di periodi nel caso delle pile di serie storiche, o il numero di unità cross section nel caso di pila di dati cross section), mentre *struttura* può essere uguale a `--stacked-time-series` o `--stacked-cross-section`. Di seguito vengono mostrati due esempi: il primo per un dataset panel sotto forma di pila di serie storiche con osservazioni per 20 periodi, il secondo per un dataset panel sotto forma di pila di dati cross section, con 5 unità cross section.

```
setobs 20 1:1 --stacked-time-series
setobs 5 1:1 --stacked-cross-section
```

Variabili dummy

In uno studio panel, può nascere l’esigenza di costruire delle variabili dummy di uno dei seguenti tipi: (a) dummy che identificano ciascuna delle unità cross-section, o (b) dummy che identificano ciascuna delle unità temporali. Il primo tipo può essere usato per permettere all’intercetta della regressione di variare tra le unità cross-section, il secondo per permettere all’intercetta di variare tra i periodi.

Per creare questo tipo di dummy, è possibile usare le tre funzioni speciali del menù Dati, Aggiungi variabile, o del comando testuale `genr`.

1. “Dummy periodiche” (comando testuale `genr dummy`). Questo comando crea un insieme di variabili dummy che identificano i periodi. La variabile `dummy_1` avrà valore 1 in ogni riga dei dati che corrisponde a un’osservazione del primo periodo, e 0 altrove; `dummy_2` avrà valore 1 in ogni riga dei dati che corrisponde a un’osservazione del secondo, e così via.
2. “Dummy per unità” (comando testuale `genr unitdum`). Questo comando crea un insieme di variabili dummy che identificano le unità cross section. La variabile `du_1` avrà valore 1 in ogni riga dei dati che corrisponde a un’osservazione della prima unità cross section, e 0 altrove; `du_2` avrà valore 1 in ogni riga dei dati che corrisponde a un’osservazione della seconda unità cross section, e così via.
3. “Dummy panel” (comando testuale `genr paneldum`). Questo comando crea entrambi i tipi di dummy: sia quelle periodiche e quelle relative alle unità cross section (longitudinali). Le dummy longitudinali sono chiamate `du_1`, `du_2` e così via, mentre le dummy periodiche sono chiamate `dt_1`, `dt_2`, ecc.

Se un dataset panel contiene l’anno di ogni osservazione all’interno della variabile `ANNO`, è possibile creare una dummy periodica per un anno particolare, ad esempio con `genr dum = (ANNO=1960)`. È anche possibile creare variabili dummy periodiche usando l’operatore modulo, `%`. Ad esempio, per creare una dummy che valga 1 ogni trenta osservazioni a partire dalla prima e 0 altrove, basta eseguire

```
genr index
genr dum = ((index-1)%30) = 0
```

Ritardi e differenze con dati panel

Se i periodi temporali sono distanziati in modo uniforme, è possibile usare valori ritardati delle variabili in una regressione panel, come anche costruire differenze prime delle variabili.

Se un dataset è identificato correttamente come panel (si veda il paragrafo precedente), `gretl` gestirà correttamente la generazione di questo tipo di variabili. Ad esempio, il comando `genr x1_1 = x1(-1)` creerà una variabile che contiene il primo ritardo di `x1`, laddove è disponibile, e il codice di valore mancante, laddove il ritardo non è disponibile.

Quanto si esegue una regressione che include questo tipo di variabili, il programma escluderà automaticamente le osservazioni mancanti.

Stima pooled

Esiste un comando speciale di stima utilizzabile con i dati panel, il comando **Pooled OLS** nel menù **Modello**. Questo comando è disponibile solo se il dataset è riconosciuto come panel; per usarlo, occorre specificare un modello senza alcuna variabile dummy relativa alle unità cross-section. Il comando esegue una semplice stima OLS, trattando allo stesso modo la variazione longitudinale e quella temporale presente nei dati. Questo modello può essere appropriato o no: il menù **Test** della finestra del modello contiene il comando **Diagnosi panel**, che esegue un test del modello OLS pooled contro le principali alternative: il modello a effetti fissi e quello a effetti casuali.

Il modello a effetti fissi aggiunge una variabile dummy per tutte le unità cross section tranne una, permettendo così all’intercetta della regressione di variare per ogni unità. Viene eseguito un test *F* per la significatività congiunta di queste dummy: un basso *p*-value per questo test va contro l’ipotesi nulla (che il semplice modello pooled OLS sia adeguato) e a favore del modello a effetti fissi.

Il modello a effetti casuali, d’altra parte, scompone la varianza dei residui in due parti: una specifica all’unità cross section, o “gruppo”, e una specifica all’osservazione particolare (la stima può essere eseguita solo se il panel è abbastanza “largo”, ossia se il numero delle unità cross section nel dataset è maggiore del numero dei parametri da stimare). La

statistica LM di Breusch-Pagan testa l'ipotesi nulla (che il modello pooled OLS sia adeguato) contro l'alternativa a effetti casuali.

Può accadere che il modello pooled OLS sia rifiutato nei confronti di entrambe le alternative, a effetti fissi o casuali. Come giudicare quindi l'adeguatezza relativa dei due metodi alternativi di stima? Il test di Hausman (i cui risultati sono mostrati a patto che il modello a effetti casuali possa essere stimato) cerca di risolvere questo problema. A patto che gli errori specifici di unità o di gruppo siano non correlati con le variabili indipendenti, lo stimatore a effetti casuali è più efficiente dello stimatore a effetti fissi; nel caso contrario lo stimatore a effetti casuali non è consistente e deve essergli preferito lo stimatore a effetti fissi. L'ipotesi nulla per il test di Hausman è che l'errore specifico di gruppo non sia correlato con le variabili indipendenti (e quindi che il modello a effetti casuali sia preferibile). Un basso p-value per questo test suggerisce di rifiutare il modello a effetti casuali in favore del modello a effetti fissi.

Per una discussione rigorosa di questo argomento, si veda il capitolo 14 di Greene (2000).

Esempio: la Penn World Table

La Penn World Table (homepage a pwt.econ.upenn.edu) è un ricco dataset panel macroeconomico, che comprende 152 paesi sull'arco temporale 1950-1992. I dati sono disponibili in formato gret1: si veda la pagina dei dati di gret1 (i dati sono liberamente scaricabili, anche se non sono distribuiti nel pacchetto principale di gret1).

L'**Esempio 7-1** sottostante apre il file `pwt56_60_89.gdt`, un sottoinsieme della Penn World Table che contiene dati per 120 paesi negli anni 1960-89, su 20 variabili, senza osservazioni mancanti (il dataset completo, anch'esso compreso nel pacchetto `pwt` per gret1, contiene molte osservazioni mancanti). Viene calcolata la crescita del PIL reale sul periodo 1960-89 per ogni paese, e viene regredita sul livello del PIL reale dell'anno 1960, per analizzare l'ipotesi della "convergenza" (ossia di una crescita più veloce da parte dei paesi che partono da una situazione peggiore).

Esempio 7-1. Uso della Penn World Table

```
open pwt56_60_89.gdt
# Per l'anno 1989 (l'ultima oss.) il ritardo 29 dà 1960, la prima oss.
genr gdp60 = RGDPL(-29)
# Calcola la crescita totale del PIL reale sui 30 anni
genr gdpgro = (RGDPL - gdp60)/gdp60
# Restringi il campione a una cross-section del 1989
smpl --restrict YEAR=1989
# convergenza: i paesi che partono più indietro crescono di più?
ols gdpgro const gdp60
# risultato: no! Proviamo una relazione inversa?
genr gdp60inv = 1/gdp60
ols gdpgro const gdp60inv
# Ancora no. Proviamo a trattare l'Africa in modo speciale?
genr afdum = (CCODE = 1)
genr afslope = afdum * gdp60
ols gdpgro const afdum gdp60 afslope
```

Capitolo 8. Grafici e diagrammi

Grafici gnuplot

Opzioni generali di gnuplot

Gretl richiama un programma separato, `gnuplot`, per generare i grafici. Gnuplot è un programma di grafica molto completo, con una miriade di opzioni. È disponibile su www.gnuplot.info (ma la versione MS Windows di `gretl` comprende già `gnuplot`). `gretl` fornisce l'accesso, attraverso un'interfaccia grafica, a una parte di queste opzioni, ma è possibile anche controllare l'aspetto di un grafico in tutti i suoi dettagli, se si vuole.

Mentre un grafico viene visualizzato, facendo clic sulla finestra del grafico si aprirà un menù pop-up con le seguenti opzioni:

- § Salva come PNG: salva il grafico in formato Portable Network Graphics
- § Salva come postscript: salva in formato encapsulated postscript (EPS)
- § Salva come Windows metafile: save in formato Enhanced Metafile (EMF).
- § Salva alla sessione come icona: il grafico apparirà sotto forma di icona quando si seleziona "Visualizza Icone" dal menù Sessione
- § Ingrandisci: permette di selezionare un'area all'interno del grafico per visualizzarla da vicino
- § Stampa: permette di stampare il grafico direttamente (disponibile solo in Gnome e MS Windows)
- § Copia negli appunti: (solo in MS Windows) permette di copiare il grafico per poi incollarlo in altri programmi Windows, come ad esempio MS Word ¹
- § Modifica: apre una finestra che permette di modificare vari dettagli dell'aspetto del grafico
- § Chiudi: chiude la finestra del grafico

Mostrare le etichette dei dati

Nel caso di semplici diagrammi a dispersione X-Y (con o senza la retta di regressione), sono disponibili altre opzioni se il dataset contiene "marcatori" (ossia etichette che identificano ogni osservazione)². Quando il diagramma a dispersione è visualizzato, muovendo il puntatore del mouse su un punto dei dati, viene mostrata l'etichetta corrispondente. In modalità predefinita, queste etichette non compaiono nelle versioni stampate o copiate del grafico, e possono essere rimosse selezionando "Cancella le etichette dei dati" dal menù pop-up del grafico. Se si desidera rendere permanenti le etichette (cosicché siano visibili anche se il grafico è stampato o copiato), ci sono due opzioni:

- § Per fissare le etichette che sono mostrate in un dato momento, selezionare "Fissa le etichette dei dati" dal menù pop-up del grafico.
- § Per fissare le etichette per tutti i punti del grafico, selezionare "Modifica" dal menù pop-up e marcare la casella "Mostra tutte le etichette dei dati". Questa opzione è disponibile solo se ci sono meno di 55 punti, e produrrà i risultati migliori se i punti del grafico non sono troppo addensati, altrimenti le etichette tenderanno a sovrapporsi.

Per rimuovere le etichette che sono state fissate in uno di questi due modi, basta selezionare "Modifica" dal menù pop-up e disattivare la casella "Mostra tutte le etichette dei dati".

1. Per ottenere i risultati migliori quando si incollano grafici nelle applicazioni di MS Office, usare il comando "Modifica, Incolla speciale..." dell'applicazione e selezionare l'opzione "Immagine (Enhanced Metafile)".

2. Per un esempio di dataset simili, si veda il file di Ramanathan `data4-10`: esso contiene dati sulle iscrizioni alle scuole private per i 50 stati degli USA, incluso Washington DC; i marcatori rappresentano i codici a due lettere per i vari stati.

Opzioni avanzate

Se si conosce `gnuplot` e si desidera un controllo sull'aspetto del grafico più preciso di quello fornito dalla finestra di modifica del grafico (opzione “Modifica”), ci sono due possibilità:

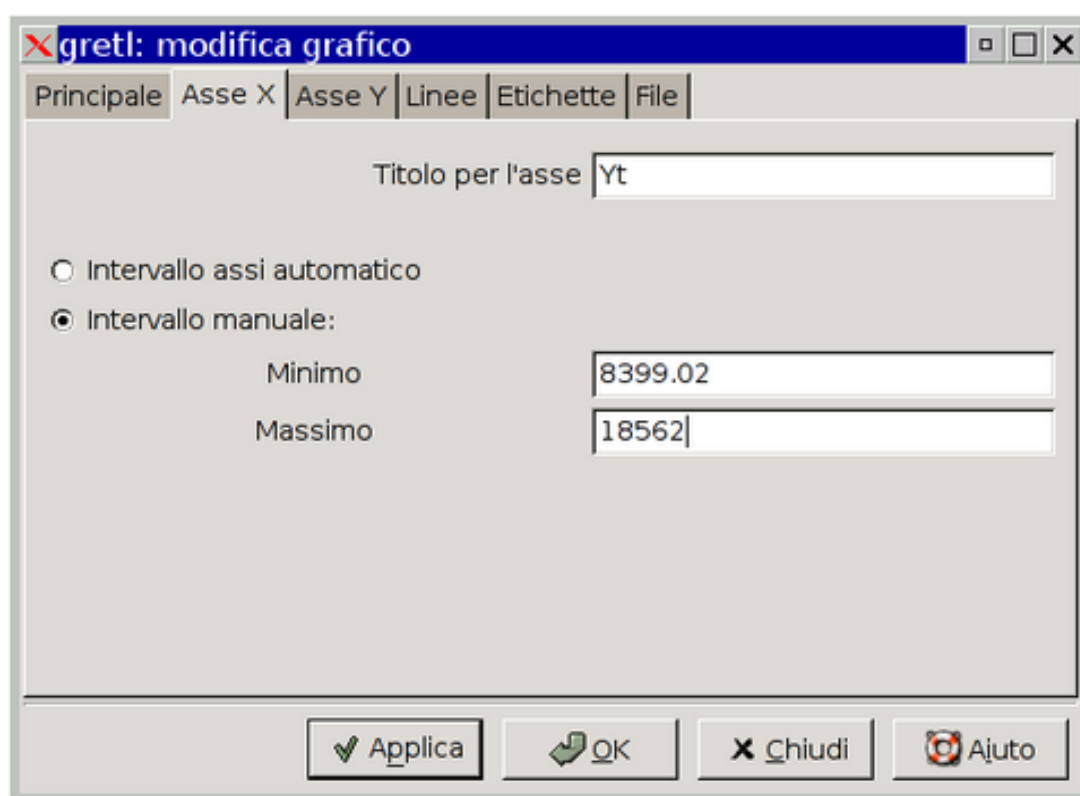
§ Una volta salvato il grafico come icona di sessione, facendo clic col tasto destro sull'icona si apre un altro menù pop-up. Una delle opzioni disponibili è “Comandi per modificare il grafico”, che apre una finestra di modifica con i comandi di `gnuplot`. È possibile modificare questi comandi e salvarli per il futuro, oppure inviarli direttamente a `gnuplot` (con il comando “File/Invia a `gnuplot`” del menù della finestra di modifica dei comandi).

§ Un altro modo per salvare i comandi del grafico (o per salvare il grafico in formati diversi da EPS o PNG) è quello di usare il comando “Modifica” nel menù pop-up del grafico per aprire la finestra di modifica del grafico, quindi fare clic su “File”: verrà visualizzato un menù a discesa con i formati in cui è possibile salvare il grafico.

Per saperne di più su `gnuplot` si veda il manuale online o www.gnuplot.info.

Si veda anche la voce `gnuplot` nel [Capitolo 14](#) e i comandi `graph` e `plot` per generare dei semplici “grafici ASCII”.

Figura 8-1. Finestra di modifica dei grafici `gnuplot` di `gretl`



Boxplot

I grafici boxplot non vengono generati con `gnuplot`, ma da `gretl` stesso.

Questi grafici (da Tukey e Chambers) mostrano la distribuzione di una variabile. La “scatola” centrale (box) racchiude il 50 per cento centrale dei dati, ossia è delimitato dal primo e terzo quartile. I “baffi” (whiskers) si estendono fino ai valori minimo e massimo. Una linea trasversale sulla scatola indica la mediana.

Nel caso dei grafici a tacca ("notches"), una tacca indica i limiti dell'intervallo di confidenza approssimato al 90 per cento per la mediana, ottenuto col metodo bootstrap (se la serie dei dati è molto lunga, potrebbe essere necessario un po' di tempo).

Facendo clic nella finestra del boxplot si ottiene un menù che permette di salvare il grafico come file encapsulated postscript (EPS) o come file postscript a piena pagina. Se si usa il sistema X Window è anche possibile salvare il grafico come file XPM, mentre in MS Windows è possibile copiarlo negli appunti in formato bitmap. Il menù dà anche la possibilità di mostrare un riepilogo in cinque numeri (minimo, primo quartile, mediana, terzo quartile, massimo) e un intervallo di confidenza per la mediana, se il boxplot è del tipo "a tacca".

Alcuni dettagli del funzionamento dei boxplot di gretl possono essere controllati attraverso un file testuale chiamato `.boxplotrc`, che viene cercato, nell'ordine, nella directory di lavoro attuale, nella directory home dell'utente (che corrisponde alla variabile d'ambiente HOME) e nella directory utente di gretl (scelta attraverso il comando "File, Preferenze, Generali..."). Tra le opzioni che possono essere specificate in questo modo ci sono: il carattere da usare per l'output in postscript (deve essere un nome di font postscript valido; il valore predefinito è Helvetica), la dimensione del carattere in punti (sempre per l'output in postscript; il valore predefinito è 12), i valori minimo e massimo per l'asse y, la larghezza e l'altezza del grafico in pixel (valori predefiniti: 560 x 448), se occorre mostrare anche i valori numerici per i quartili e la mediana (l'impostazione predefinita non li mostra) e se occorre indicare separatamente gli outlier, ossia i punti che distano più di 1.5 volte il range interquartile dalla scatola centrale (l'impostazione predefinita non li mostra). Ecco un esempio:

```
font = Times-Roman
fontsize = 16
max = 4.0
min = 0
width = 400
height = 448
numbers = %3.2f
outliers = true
```

Sulla penultima riga, il valore associato a `numbers` è una stringa di formato "printf" come quelle usate nel linguaggio di programmazione C; se viene specificata, controlla il modo in cui vengono mostrati la mediana e i quartili accanto al boxplot, altrimenti questi valori non vengono mostrati. Nell'esempio, i valori verranno mostrati usando 3 cifre in totale e 2 cifre di precisione dopo il punto decimale.

Non occorre specificare tutte le opzioni, né importa l'ordine in cui vengono specificate. Le righe che non seguono la struttura "variabile = valore" vengono ignorate, così come le righe che iniziano con il carattere cancelletto, #.

Dopo ogni variabile specificata nel comando boxplot è possibile inserire un'espressione booleana tra parentesi per limitare il campione da utilizzare per la variabile in questione, avendo cura di inserire uno spazio tra il nome (o il numero) della variabile e l'espressione. Si supponga di avere una variabile `salario` con gli stipendi di uomini e donne, e una variabile dummy `GENERE` che vale 1 per gli uomini e 0 per le donne. In questo caso è possibile disegnare dei boxplot comparativi usando il seguente comando nella finestra di dialogo:

```
salario (GENERE=1) salario (GENERE=0)
```

Capitolo 9. Minimi quadrati non lineari

Introduzione ed esempi

Dalla versione 1.0.9, `gretl` supporta i minimi quadrati non lineari (NLS - nonlinear least squares), usando una variante dell'algoritmo Levenberg-Marquandt. L'utente deve fornire la specificazione della funzione di regressione, e, prima ancora di fare questo, occorre "dichiarare" i parametri da stimare e fornire dei valori iniziali. È anche possibile indicare analiticamente delle derivate della funzione di regressione rispetto a ognuno dei parametri. La tolleranza usata per fermare la procedura iterativa di stima può essere impostata con il comando `genr`.

La sintassi per la specificazione della funzione da stimare è la stessa usata per il comando `genr`. Ecco due esempi, che includono anche le derivate.

Esempio 9-1. Funzione di consumo da Greene

```
nls C = alfa + beta * Y^gamma
deriv alfa = 1
deriv beta = Y^gamma
deriv gamma = beta * Y^gamma * log(Y)
end nls
```

Esempio 9-2. Funzione non lineare da Russell Davidson

```
nls y = alfa + beta * x1 + (1/beta) * x2
deriv alfa = 1
deriv beta = x1 - x2/(beta*beta)
end nls
```

Si notino i comandi `nls` (che indica la funzione di regressione), `deriv` (che indica la specificazione di una derivata) e `end nls`, che conclude la specificazione e avvia la stima. Se si aggiunge l'opzione `--vcv` all'ultima riga, verrà mostrata la matrice di covarianza delle stime dei parametri.

Inizializzazione dei parametri

Occorre definire dei valori iniziali per i parametri della funzione di regressione prima di eseguire il comando `nls`. Per farlo, è possibile usare il comando `genr` (o, nella versione grafica del programma, con il comando Definisci nuova variabile). In alcuni casi, in cui la funzione non lineare è una generalizzazione di un modello lineare (o di una sua forma ristretta), può essere conveniente eseguire un comando `ols` e inizializzare i parametri a partire dalle stime OLS dei coefficienti. In relazione al primo degli esempi visti sopra, si potrebbe usare:

```
ols C 0 Y
genr alfa = coeff(0)
genr beta = coeff(Y)
genr gamma = 1
```

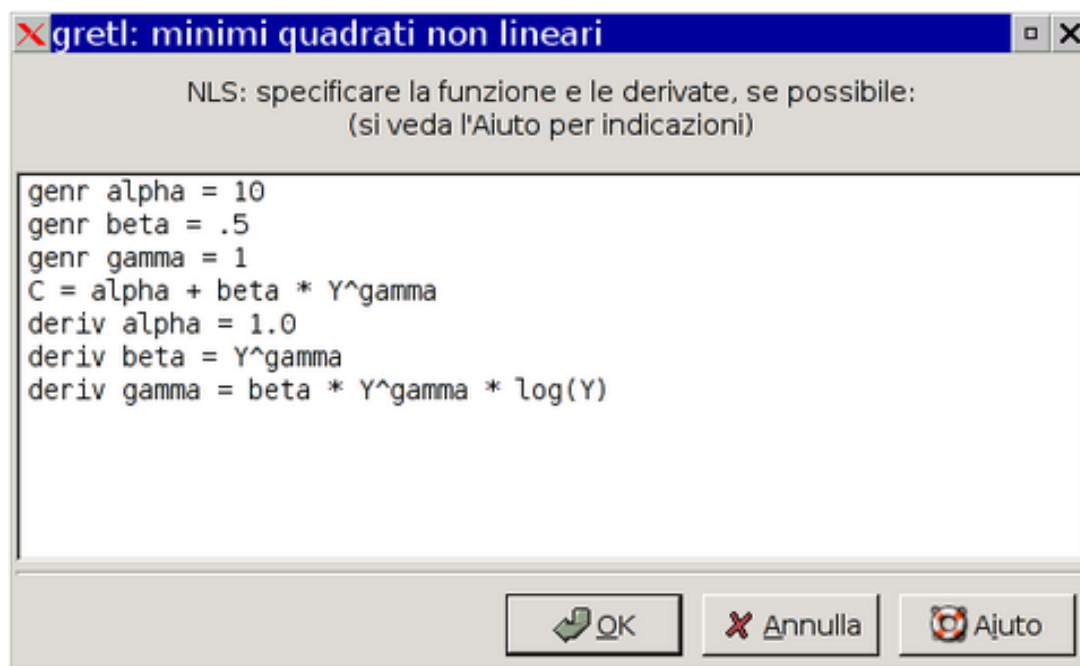
E in relazione al secondo esempio si userebbe:

```
ols y 0 x1 x2
genr alfa = coeff(0)
genr beta = coeff(x1)
```

Finestra di dialogo NLS

Proabilmente il modo più comodo di formulare i comandi per una stima NLS consiste nell'usare uno script per gretl, ma è possibile anche procedere interattivamente, selezionando il comando Minimi quadrati non lineari dal menù Modello. In questo modo, si aprirà una finestra di dialogo in cui è possibile scrivere la specificazione della funzione (opzionalmente preceduta da linee `genr` per impostare i valori iniziali dei parametri) e le derivate, se sono disponibili. Un esempio è mostrato nella [Figura 9-1](#). Si noti che in questo contesto non occorre scrivere i comandi `nls` e `end nls`.

Figura 9-1. Finestra di dialogo NLS



Derivate analitiche e numeriche

Se si è in grado di calcolare le derivate dalla funzione di regressione rispetto ai parametri, è consigliabile indicarle come mostrato negli esempi precedenti. Se ciò non è possibile, gretl calcolerà delle derivate approssimate numericamente. Le proprietà dell'algoritmo NLS in questo caso potrebbero non essere ottimali (si veda [la Sezione Accuratezza numerica](#)).

Se vengono fornite delle derivate analitiche, ne viene controllata la coerenza con la funzione non lineare data. Se esse sono chiaramente scorrette, la stima viene annullata con un messaggio di errore. Se le derivate sono "sospette", viene mostrato un messaggio di avvertimento, ma la stima prosegue. Questo avvertimento può essere causato da derivate scorrette, ma può anche essere dovuto a un alto grado di collinearità tra le derivate.

Si noti che non è possibile mischiare derivate numeriche e analitiche: se si indicano espressioni analitiche per le derivate, occorre farlo per tutte.

Arresto della procedura

La procedura di stima NLS è iterativa: l'iterazione viene arrestata si verifica una qualunque delle seguenti condizioni: viene raggiunto un criterio di convergenza, o si supera il massimo numero di iterazioni impostato. Il massimo numero di iterazioni è $100 \cdot (k+1)$ se vengono fornite derivate analitiche, o $200 \cdot (k+1)$ se vengono usate derivate numeriche, dove k indica il numero di parametri da stimare. Il criterio di convergenza consiste nel

fatto che l'errore relativo nella somma dei quadrati e/o l'errore relativo tra il vettore dei coefficienti e la soluzione sia stimato inferiore a un certo valore piccolo. Questo “valore piccolo” è impostato in modo predefinito pari alla precisione della macchina elevato alla potenza $3/4$, ma può essere impostato con il comando `genr`, usando la variabile speciale `toler`. Ad esempio

```
genr toler = .0001
```

imposterà la tolleranza a 0.0001.

Dettagli sul codice

Il motore sottostante la stima NLS è basato sulla suite di funzioni `minpack`, disponibile su netlib.org. Nello specifico, sono usate le seguenti funzioni `minpack`:

<code>lmdcr</code>	Algoritmo Levenberg-Marquandt con derivate analitiche
<code>chkder</code>	Controllo delle derivate analitiche fornite
<code>lmdif</code>	Algoritmo Levenberg-Marquandt con derivate numeriche
<code>fdjac2</code>	Calcolo del Jacobiano approssimato finale se si usano le derivate numeriche
<code>dmpar</code>	Determinazione della precisione della macchina

In caso di successo nell'iterazione Levenberg-Marquandt, viene usata una regressione Gauss-Newton per calcolare la matrice di covarianza per le stime dei parametri. Poiché i risultati NLS sono asintotici, si può discutere sulla necessità di applicare una correzione per i gradi di libertà nel calcolo dell'errore standard della regressione (e di quello delle stime dei parametri). Per confrontabilità con OLS, e seguendo il ragionamento in Davidson e MacKinnon (1993), le stime calcolate in `gretl` *usano* una correzione per i gradi di libertà.

Accuratezza numerica

La [Tabella 9-1](#) mostra i risultati dell'uso della procedura NLS di `gretl` sui 27 "Statistical Reference Dataset" forniti dal National Institute of Standards and Technology (NIST) statunitense, per il test del software di regressione non lineare ¹. Per ogni dataset, il file di test indicano due valori iniziali per i parametri, quindi il test completo riporta 54 stime. Sono stati eseguiti due test completi, uno usando derivate analitiche e uno usando approssimazioni numeriche; in entrambi casi si è usata la tolleranza predefinita²

Sulle 54 stime, `gretl` non riesce a produrre una soluzione in 4 casi, se vengono usate le derivate analitiche, e in 5 casi se vengono usate le approssimazioni numeriche. Dei quattro fallimenti in modalità derivate analitiche, due sono dovuti alla non convergenza dell'algoritmo Levenberg-Marquandt dopo il numero massimo di iterazioni (su MGH09 e Bennett5, entrambi descritti dal NIST come di “alta difficoltà”) e due sono dovuti ad errori di intervallo (valori in virgola mobile al di fuori dei limiti) occorsi durante il calcolo del Jacobiano (su BoxBOD e MGH17, rispettivamente descritti come di “alta difficoltà” e di “media difficoltà”). In modalità approssimazione numerica, l'ulteriore caso di fallimento è rappresentato da MGH10 (“alta difficoltà”, massimo numero di iterazioni raggiunto).

La tabella mostra informazioni su vari aspetti dei test: numero di fallimenti, numero medio di iterazioni richieste per produrre una soluzione, e due tipi di misura dell'accuratezza dei risultati per i parametri e per i loro errori standard.

Per ognuno dei 54 test eseguiti in ogni modalità, se è stata prodotta una soluzione, sono state confrontate le stime dei parametri ottenute da `gretl` con i valori certificati dal NIST. È stata definita la variabile “numero minimo di cifre corrette” per una data stima come il numero di cifre significative per cui la *meno accurata* delle stime di `gretl` coincide con il

1. Per una discussione dell'accuratezza di `gretl` nella stima di modelli lineari, si veda [Appendice C](#)

2. I dati mostrati nella tabella derivano da una versione di `gretl` 1.0.9, compilata con `gcc` 3.3, collegata a `glibc` 2.3.2 ed eseguita in Linux su un PC i686 (IBM ThinkPad A21m).

valore certificato. La tabella mostra i valori medio e minimo di questa variabile, calcolati sulle stime che hanno prodotto una soluzione; la stessa informazione è fornita per gli errori standard stimati ³

La seconda misura di accuratezza mostrata è la percentuale di casi, tenendo conto di tutti i parametri di tutte le stime giunte a buon fine, in cui la stima di `gret1` concorda con il valore certificato per almeno 6 cifre significative, che sono mostrate in modo predefinito nei risultati delle regressioni di `gret1`.

Tabella 9-1. Regressione non lineare: i test NIST

	Derivate analitiche	Derivate numeriche
Falimenti in 54 test	4	5
Iterazioni medie	32	127
Media del "numero minimo di cifre corrette", stima dei parametri	8.120	6.980
Valore minimo del "numero minimo di cifre corrette", stima dei parametri	4	3
Media del "numero minimo di cifre corrette", stima degli errori standard	8.000	5.673
Valore minimo del "numero minimo di cifre corrette", stima degli errori standard	5	2
Percentuale delle stime corrette a 6 cifre, stima dei parametri	96.5	91.9
Percentuale delle stime corrette a 6 cifre, stima degli errori standard	97.7	77.3

Usando derivate analitiche, i valori dei casi peggiori sia per le stime dei parametri che per gli errori standard sono stati migliorati a 6 cifre corrette restringendo il valore di tolleranza a $1.0\text{e-}14$. Usando derivate numeriche, la stessa modifica del limite di tolleranza ha innalzato la precisione dei valori peggiori a 5 cifre corrette per i parametri e 3 cifre per gli errori standard, al costo di un fallimento in più nella convergenza.

Si noti la tendenziale superiorità delle derivate analitiche: in media le soluzioni ai problemi dei test sono state ottenute con molte meno iterazioni e i risultati sono più accurati (in modo evidente per gli errori standard stimati). Si noti anche che i risultati a 6 cifre mostrati da `gret1` non sono affidabili al 100 per cento per i problemi non lineari difficili (in particolare se si usano derivate numeriche). Tenendo presente questi limiti, la percentuale dei casi in cui i risultati sono accurati alla sesta cifra o più sembra sufficiente per giustificare l'utilizzo in questa forma.

3. Per gli errori standard, dalle statistiche mostrate nella tabella, è stato escluso l'outlier costituito da `Lanczos1`, che rappresenta un caso strano, composto da dati generati con un adattamento quasi esatto; gli errori standard sono di 9 o 10 ordini di grandezza più piccoli dei coefficienti. In questo caso `gret1` riesce a riprodurre gli errori standard solo per 3 cifre (con derivate analitiche) e per 2 cifre (con derivate numeriche).

Capitolo 10. Costrutti loop

Introduzione

Il comando `loop` apre una modalità speciale, in cui `gretl` accetta un blocco di comandi da ripetere una o più volte. Questa funzionalità è destinata all'uso per le simulazioni Monte Carlo, per il bootstrapping delle statistiche test e altre procedure di stima iterativa. La forma generale di un loop, o ciclo, è:

```
loop espressione di controllo [ --progressive | --verbose ]
    corpo del loop
endloop
```

Come si spiega oltre, sono disponibili cinque tipi di *espressione di controllo*. Nel *corpo del loop* sono accettati i seguenti comandi: `genr`, `ols`, `print`, `printf`, `pvalue`, `sim`, `smpl`, `store`, `summary`, `if`, `else` e `endif`.

In modalità predefinita, il comando `genr` all'interno di un loop opera in modo silenzioso (senza mostrare informazioni sulle variabili generate). Per ottenere informazioni da `genr` è possibile specificare l'opzione `--verbose` con `loop`.

L'opzione `--progressive` di `loop` modifica il comportamento dei comandi `ols`, `print` e `store` rendendoli più comodi per l'uso in analisi di tipo Monte Carlo (si veda [la Sezione La modalità "progressiva"](#)).

Le sezioni che seguono descrivono le varie forme di espressioni di controllo e forniscono alcuni esempi di uso dei loop.



Se occorre eseguire un'analisi Monte Carlo con molte migliaia di ripetizioni, possono verificarsi problemi di memoria e di tempo di calcolo. Un modo per minimizzare l'uso delle risorse di sistema consiste nell'eseguire lo script usando il programma a riga di comando, `gretlcli`, reindirizzando i risultati su un file.

Varianti di controllo del loop

Loop limitati

Il modo più semplice di controllare un loop consiste nello specificare direttamente il numero di volte che il ciclo deve essere ripetuto, in un cosiddetto "loop limitato". Il numero di ripetizioni può essere una costante numerica, ad esempio `loop 1000`, o può essere letto da una variabile, come in `loop volte`.

Se il numero di ripetizioni è indicato da una variabile, ad es. `volte`, la variabile dovrebbe essere uno scalare intero: se si usa una serie, viene letto il primo valore, e se questo non è intero, viene troncata la sua parte decimale. Si noti che `volte` viene valutata solo una volta, quando il loop viene impostato.

Loop di tipo "while"

Un secondo tipo di espressione di controllo consiste nell'uso del comando `while` seguito da una disuguaglianza, in cui il termine a sinistra è il nome di una variabile predefinita, mentre il lato destro può essere una costante numerica o il nome di un'altra variabile predefinita. Ad esempio:

```
loop while differ > .00001
```

L'esecuzione del ciclo di comandi continuerà fintanto che la condizione specificata rimane vera. Se il termine a destra della disuguaglianza è una variabile, essa viene valutata all'inizio di ogni nuova iterazione del ciclo.

Loop con indice

Un terzo tipo di controllo di un loop utilizza la speciale variabile indice *i*. In questo caso, vengono specificati valori iniziali e finali per *i*, che viene incrementata di uno ogni volta che viene eseguito il ciclo. La sintassi è la seguente: `loop i=1..20`.

La variabile indice può essere usata all'interno del corpo del loop, in uno dei modi seguenti: è possibile accedere al valore di *i* (si veda l'Esempio 10-4), oppure è possibile usare la sua rappresentazione come stringa *\$i* (si veda l'Esempio 10-5).

I valori iniziale e finale per l'indice possono essere indicati in forma numerica, o come riferimento a variabili predefinite. Nell'ultimo caso, le variabili vengono valutate una volta, quando il loop viene impostato. Inoltre, con dataset di serie storiche è possibile indicare i valori iniziale e finale sotto forma di date, ad esempio: `loop i=1950:1..1999:4`.

Loop di tipo “for each”

Anche il terzo tipo di controllo usa la variabile interna *i*, che in questo caso può assumere valori solo all'interno di una lista specifica di stringhe. Il loop è eseguito una volta per ogni stringa presente nella lista, agevolando l'esecuzione di operazioni ripetitive su un gruppo di variabili. Ecco un esempio:

```
loop foreach i mele pere pesche
  print "$i"
endloop
```

Questo loop verrà eseguito tre volte, mostrando “mele”, “pere” e “pesche” ad ogni iterazione.

Per eseguire un loop su una lista di variabili contigue nel dataset, è possibile indicare i nomi della prima e dell'ultima variabile nella lista, separate da “..”, invece di dover indicare tutti i nomi. Ad esempio, ipotizzando di avere 50 variabili AK, AL, ..., WY, che contengono livelli di reddito per gli stati degli USA, per stimare una regressione del reddito sul tempo per ognuno degli stati, basta eseguire:

```
genr time
loop foreach i AL..WY
  ols $i const time
endloop
```

Loop di tipo “for”

L'ultimo tipo di controllo usa una forma semplificata dell'istruzione `for` del linguaggio di programmazione C. L'espressione di controllo si compone di tre parti, separate da punto e virgola. La prima parte specifica una condizione iniziale, espressa per mezzo di una variabile di controllo; la seconda parte imposta una condizione di continuazione (espressa in funzione della stessa variabile di controllo), mentre la terza parte specifica un incremento (o un decremento) per la variabile di controllo, da applicare ogni volta che il ciclo viene eseguito. L'intera espressione deve essere racchiusa tra parentesi. Ad esempio:

```
loop for (r=0.01; r<.991; r+=.01)
```

La variabile *r* assumerà i valori 0.01, 0.02, ..., 0.99 nel giro di 99 iterazioni. Si noti che a causa della precisione limitata dell'aritmetica in virgola mobile usata dal computer, può dover essere necessario usare una condizione di continuazione come quella mostrata sopra, `r<.991`, invece della più “naturale” `r<=.99` (usando numeri in doppia precisione su un processore x86, quando ci si aspetta che *r* valga 0.99, potrebbe in realtà valere 0.9900000000000001).

Altre regole per i tre componenti dell'espressione di controllo: (1) la condizione iniziale deve avere forma `X1 = Y1`, dove *Y1* deve essere una costante numerica o una variabile predefinita. Se la variabile *X1* non esiste, viene creata automaticamente. (2) La condizione di continuazione deve avere la forma `X1 operatore Y2`, dove l'*operatore* può essere `<`, `>`,

$\leq$ o $\geq$ e Y2 deve essere una costante numerica o una variabile predefinita (nel caso in cui sia una variabile, essa viene valutata ad ogni esecuzione del ciclo). (3) L'espressione che indica l'incremento o il decremento deve avere la forma $X1 \pm \text{DELTA}$, oppure $X1 \mp \text{DELTA}$, dove DELTA è una costante numerica o una variabile predefinita (nel secondo caso, essa viene valutata solo una volta, quando il loop viene impostato).

La modalità “progressiva”

Usando l'opzione `--progressive` nel comando `loop`, l'effetto dei comandi `ols`, `print` e `store` è modificato nel modo seguente.

`ols`: i risultati di ogni iterazione della regressione non vengono mostrati. Al contrario, una volta terminato il loop si ottiene un elenco dei seguenti valori: (a) il valore medio di ognuno dei coefficienti stimati, calcolato su tutte le iterazioni; (b) la deviazione standard relativa a questa media; (c) il valore medio dell'errore standard stimato per ogni coefficiente; (d) la deviazione standard degli errori standard stimati. Tutto ciò ha senso solo se ogni iterazione del loop contiene un elemento di casualità.

`print`: se si usa questo comando per mostrare il valore di una variabile, questo non viene mostrato ad ogni iterazione. Al contrario, alla fine del loop si ottiene il valore medio e la deviazione standard della variabile, calcolata su tutte le iterazioni del ciclo. Questa funzione è utile per le variabili che assumono un singolo valore per ogni iterazione, ad esempio la somma dei quadrati degli errori di una regressione.

`store`: questo comando scrive i valori delle variabili specificate, ad ogni iterazione del loop, nel file indicato, ossia, tiene traccia completa del valore delle variabili in tutte le iterazioni. Ad esempio, si potrebbero salvare le stime dei coefficienti per poi studiarne la distribuzione di frequenza. È possibile usare il comando `store` solo una volta all'interno di un loop.

Esempi di loop

Esempio di procedura Monte Carlo

Un semplice esempio di uso della modalità “progressiva” per realizzare una procedura Monte Carlo è mostrato in [Esempio 10-1](#).

Esempio 10-1. Un semplice loop di tipo Monte Carlo

```

nulldata 50
seed 547
genr x = 100 * uniform()
# Apre un loop "progressivo", da ripetere 100 volte
loop 100 --progressive
  genr u = 10 * normal()
  # Costruisce la variabile dipendente
  genr y = 10*x + u
  # Esegue la regressione OLS
  ols y const x
  # Definisce variabili per i coefficienti e R-quadro
  genr a = coeff(const)
  genr b = coeff(x)
  genr r2 = $rsq
  # Mostra le statistiche su queste variabili
  print a b r2
  # Salva i coefficienti in un file
  store coeffs.gdt a b
endloop

```

Questo loop mostrerà le statistiche di riepilogo per le stime di "a", "b" e R^2 lungo le 100 iterazioni. Dopo aver eseguito il loop, è possibile aprire con gret1 il file `coeffs.gdt`, che contiene le stime dei singoli coefficienti durante tutte le iterazioni, ed esaminare nel dettaglio la distribuzione di frequenza delle stime.

Il comando `nulldata` è utile per le procedure Monte Carlo: invece di aprire un "vero" dataset, `nulldata 50` (ad esempio) apre un finto dataset da 50 osservazioni, che contiene solo la costante e una variabile indice. Successivamente è possibile aggiungervi variabili usando il comando `genr`.

Si veda il comando `set` per informazioni su come generare numeri pseudo-casuali in modo ripetibile.

Minimi quadrati iterati

L'Esempio 10-2 usa un loop di tipo "while" per replicare la stima di una funzione di consumo non lineare nella forma $C = \alpha + \beta Y^\gamma + \epsilon$ presentata in Greene (2000, Esempio 11.3). Questo script è compreso nella distribuzione di gret1 con il nome `greene11_3.inp`; è possibile aprirlo usando il comando del menù "File, Apri file comandi, File di esempio, Greene...".

L'opzione `--print-final` per il comando `ols` fa sì che non vengano mostrati i risultati della regressione per ogni iterazione, ma solo quelli dell'ultima iterazione del loop.

Esempio 10-2. Funzione di consumo non lineare

```
open greene11_3.gdt
# Esegue la regressione OLS iniziale
ols C 0 Y
genr essbak = $ess
genr essdiff = 1
genr beta = coeff(Y)
genr gamma = 1
# Itera OLS finché la somma dei quadrati degli errori converge
loop while essdiff > .00001
  # Genera le variabili linearizzate
  genr C0 = C + gamma * beta * Y^gamma * log(Y)
  genr x1 = Y^gamma
  genr x2 = beta * Y^gamma * log(Y)
  # Esegue la regressione OLS
  ols C0 0 x1 x2 --print-final --no-df-corr --vcv
  genr beta = coeff(x1)
  genr gamma = coeff(x2)
  genr ess = $ess
  genr essdiff = abs(ess - essbak)/essbak
  genr essbak = ess
endloop
# Mostra le stime dei parametri usando i "nomi giusti"
noecho
printf "alfa = %g\n", coeff(0)
printf "beta = %g\n", beta
printf "gamma = %g\n", gamma
```

L'Esempio 10-3 (gentilmente fornito da Riccardo "Jack" Lucchetti dell'Università di Ancona) mostra come sia possibile usare un loop per stimare un modello ARMA usando la regressione "prodotto esterno del gradiente" (OPG - "outer product of the gradient") discussa da Davidson e MacKinnon nel loro *Estimation and Inference in Econometrics*.

Esempio 10-3. ARMA 1, 1

```

open armaloop.gdt

genr c = 0
genr a = 0.1
genr m = 0.1

genr e = const * 0.0
genr de_c = e
genr de_a = e
genr de_m = e

genr crit = 1
loop while crit > 1.0e-9

    # Errori di previsione "one-step"
    genr e = y - c - a*y(-1) - m*e(-1)

    # Log-verosimiglianza
    genr loglik = -0.5 * sum(e^2)
    print loglik

    # Derivate parziali degli errori di previsione rispetto a c, a e m
    genr de_c = -1 - m * de_c(-1)
    genr de_a = -y(-1) - m * de_a(-1)
    genr de_m = -e(-1) - m * de_m(-1)

    # Derivate parziali di l rispetto a c, a e m
    genr sc_c = -de_c * e
    genr sc_a = -de_a * e
    genr sc_m = -de_m * e

    # Regressione OPG
    ols const sc_c sc_a sc_m --print-final --no-df-corr --vcv

    # Aggiorna i parametri
    genr dc = coeff(sc_c)
    genr c = c + dc
    genr da = coeff(sc_a)
    genr a = a + da
    genr dm = coeff(sc_m)
    genr m = m + dm

    printf "    constant          = %.8g (gradient = %#.6g)\n", c, dc
    printf "    ar1 coefficient = %.8g (gradient = %#.6g)\n", a, da
    printf "    ma1 coefficient = %.8g (gradient = %#.6g)\n", m, dm

    genr crit = $T - $ess
    print crit
endloop

genr se_c = stderr(sc_c)
genr se_a = stderr(sc_a)
genr se_m = stderr(sc_m)

noecho
print "
printf "constant = %.8g (se = %#.6g, t = %.4f)\n", c, se_c, c/se_c
printf "ar1 term = %.8g (se = %#.6g, t = %.4f)\n", a, se_a, a/se_a
printf "ma1 term = %.8g (se = %#.6g, t = %.4f)\n", m, se_m, m/se_m

```

Esempi di loop con indice

L'Esempio 10-4 mostra un loop con indice, in cui il comando `smp1` contiene la variabile indice `i`. Si supponga di avere un dataset di tipo panel, con osservazioni su alcuni ospedali per gli anni dal 1991 al 2000 (dove l'anno dell'osservazione è indicato da una variabile chiamata `anno`). Ad ogni iterazione, restringiamo il campione a un certo anno e calcoliamo statistiche di riepilogo sulla dimensione longitudinale (cross-section) per le variabili da 1 a 4.

Esempio 10-4. Statistiche panel

```
open ospedali.gdt
loop i=1991..2000
  smp1 (annor=i) --restrict --replace
  summary 1 2 3 4
endloop
```

L'Esempio 10-5 illustra un loop indicizzato per sostituire stringhe.

Esempio 10-5. Sostituzione di stringhe

```
open bea.dat
loop i=1987..2001
  genr V = COMP$i
  genr TC = GOC$i - PBT$i
  genr C = TC - V
  ols PBT$i const TC V
endloop
```

Alla prima iterazione, la variabile `V` verrà impostata a `COMP1987` e la variabile dipendente per il comando `ols` sarà `PBT1987`. All'iterazione successiva, `V` verrà ridefinita come `COMP1988` e la variabile dipendente della regressione sarà `PBT1988`, e così via.

Capitolo 11. Funzioni definite dall'utente

Introduzione

A partire dalla versione 1.4.0, gretl contiene un nuovo meccanismo per definire funzioni all'interno di uno script.¹

Definizione di una funzione

Occorre definire una funzione prima di poterla utilizzare. La sintassi per farlo è la seguente:

```
function nome-funzione parametri
    corpo della funzione
end function
```

Il *nome-funzione* identifica la funzione in modo univoco: deve iniziare con una lettera, può essere lungo al massimo 31 caratteri (eventuali caratteri in più verranno troncati) e non può contenere spazi. Se si tenta di definire una funzione con lo stesso nome di un comando di gretl o di una funzione preesistente, si otterrà un'errore. Nel secondo caso, per evitare un errore (ossia per poter ridefinire una funzione utente), basta far precedere la definizione della funzione da:

```
function nome-funzioneclear
```

I *parametri* di una funzione (se esistono) vanno indicati sotto forma di lista separata da virgole. I parametri possono essere di tre tipi: variabili ordinarie (serie di dati), variabili scalari, o liste di variabili. Ogni elemento della lista di parametri è composto da due termini: per prima cosa un indicatore di tipo (*series*, *scalar* o *list*), quindi il nome con cui il parametro verrà riconosciuto all'interno della funzione. Ecco un esempio (le parentesi che racchiudono la lista dei parametri sono opzionali):

```
function funzione (series y, list xvars, scalar verbose)
```

Quando una funzione viene chiamata, i parametri vengono istanziati usando gli argomenti indicati nella chiamata della funzione. Vengono fatti dei controlli automatici per assicurarsi che il numero degli argomenti contenuti in una chiamata di funzione corrisponda al numero di parametri, e che i tipi degli argomenti corrispondano ai tipi specificati nella definizione della funzione; se qualcuna di queste condizioni è violata, viene segnalato un errore. Una serie può essere indicata come argomento specificando il nome della variabile in questione, oppure il suo numero identificativo. Gli scalari possono essere indicati specificando il nome di una variabile o un valore numerico (non è possibile indicare il numero identificativo della variabile). Le liste devono essere indicate per nome.

Il *corpo della funzione* è composto da comandi gretl o funzioni definite dall'utente (ossia, le funzioni possono essere nidificate). Una funzione può persino chiamare sé stessa (ossia, le funzioni possono essere ricorsive), fino a un massimo di 8 "livelli di ricorsività", ma questo valore è provvisorio, potrà essere aumentato se sembra troppo restrittivo. Se il corpo della funzione può contenere chiamate ad altre funzioni non può però contenere definizioni di altre funzioni, ossia non è possibile definire una funzione all'interno di un'altra funzione.

Le funzioni possono essere chiamate, ma non definite, all'interno di un loop (si veda il [Capitolo 10](#)).

1. Si noti che il nuovo modo di definire le funzioni presenta aspetti non compatibili con la versione 1.3.3 del programma.

Chiamare una funzione

Una funzione utente viene chiamata, o invocata, usando il suo nome, eventualmente seguito da argomenti; se si usano due o più argomenti, vanno separati da virgole. L'esempio seguente mostra una chiamata di funzione che rispetta la definizione della funzione stessa.

```
# Definizione della funzione
function ols_ess (series y, list xvars)
  ols y 0 xvars --quiet
  scalar myess = $ess
  printf "ESS = %g\n", myess
  return scalar myess
end function
# Script principale
open data4-1
list xlist = 2 3 4
# Chiamata della funzione (il valore restituito viene qui ignorato)
ols_ess price, xlist
```

La chiamata della funzione contiene due argomenti: il primo è una serie di dati specificata per nome, e il secondo è una lista di regressori. Si noti che la funzione produce la variabile `myess` come risultato, ma in questo esempio esso è ignorato.

(Una nota a margine: se si desidera che una funzione calcoli alcuni valori che hanno a che fare con una regressione, ma non si è interessati ai risultati completi della regressione, è possibile usare l'opzione `--quiet` con il comando di stima, come visto sopra.)

Un secondo esempio mostra una chiamata di funzione che assegna i valori prodotti dalla funzione a variabili:

```
# Definizione di funzione
function ess_uhat (series y, list xvars)
  ols y 0 xvars --quiet
  scalar myess = $ess
  printf "ESS = %g\n", myess
  series uh = $uhat
  return scalar myess, series uh
end function
# Script principale
open data4-1
list xlist = 2 3 4
# Chiamata di funzione
(SSR, resids) = ess_uhat price, xlist
```

Ambito delle variabili

Tutte le variabili create in una funzione sono locali a quella funzione, e vengono distrutte quando la funzione termina la sua esecuzione, a meno che esse siano rese disponibili come valori di ritorno, e che questi siano “raccolti” o assegnati nella chiamata della funzione.

Le funzioni non hanno accesso alle variabili nell’“ambito esterno” (ossia le variabili che esistono nello script da cui la funzione è chiamata), a meno che queste siano passate esplicitamente alla funzione come argomenti. Anche in questo caso, quello che viene passato alla funzione è una copia delle variabili in questione, quindi le variabili nell’ambito esterno non vengono mai modificate da una funzione, a meno che non vengano assegnate come valore prodotto da una funzione.

Valori di uscita

Le funzioni possono produrre zero o più valori di uscita: questi possono essere serie o scalari (non liste). I valori di uscita sono specificati con una dichiarazione all'interno del corpo della funzione che comincia con la parola chiave `return`, seguita da una lista separata da virgole, ogni elemento della quale è composto da un indicatore di tipo e dal nome di una variabile (in modo analogo a quanto avviene per la lista dei parametri di una funzione). All'interno di una funzione può esserci solo una di queste dichiarazioni. Ecco un esempio di dichiarazione `return` valida:

```
return scalar SSR, series resid
```

Si noti che la dichiarazione `return` *non* indica alla funzione di produrre il valore (terminare) nel punto in cui essa appare nel corpo della funzione. Piuttosto, essa specifica quali variabili sono disponibili per l'assegnazione quando la funzione terminerà, cosa che può avvenire solo quando: a) viene raggiunta la fine del codice che definisce la funzione; b) si incontra una dichiarazione `funcerr` (si veda oltre); c) `gretl` produce un errore.

La parola chiave `funcerr`, che può essere seguita da una stringa contenuta fra virgolette doppie, fa terminare una funzione con un messaggio di errore, che può essere generico o rappresentato dalla stringa indicata.

Controllo degli errori

Quando `gretl` legge e “compila” una definizione di funzione, esegue un controllo degli errori minimale: controlla che il nome della funzione sia accettabile e che non si tenti di definire una funzione all'interno di una funzione (si veda [la Sezione Definizione di una funzione](#)). Se il corpo della funzione contiene comandi non validi, verrà segnalato solo quando la funzione viene chiamata.

Capitolo 12. Cointegrazione e modelli Vector Error Correction

Il test di cointegrazione di Johansen

Il test di Johansen per la cointegrazione deve tenere conto di quali ipotesi vengono fatte a proposito dei termini deterministici, per cui si possono individuare i ben noti “cinque casi”. Una presentazione approfondita dei cinque casi richiede una certa quantità di algebra matriciale, ma è possibile dare un'intuizione del problema per mezzo di un semplice esempio.

Si consideri una serie x_t che si comporta nel modo seguente

$$x_t = m + x_{t-1} + \varepsilon_t$$

dove m è un numero reale e ε_t è un processo “white noise”. Come si può facilmente mostrare, x_t è un “random walk” che fluttua intorno a un trend deterministico con pendenza m . Nel caso particolare in cui $m = 0$, il trend deterministico scompare e x_t è un puro random walk.

Si consideri ora un altro processo y_t definito da

$$y_t = k + x_t + u_t$$

dove, ancora, k è un numero reale e u_t è un processo white noise. Poiché u_t è stazionario per definizione, x_t e y_t sono cointegrate, ossia la loro differenza $z_t = y_t - x_t = k + u_t$ è un processo stazionario. Per $k = 0$, z_t è un semplice white noise a media zero, mentre per $k \neq 0$ il processo z_t è white noise con media diversa da zero.

Dopo alcune semplici sostituzioni, le due equazioni precedenti possono essere rappresentate congiuntamente come un sistema VAR(1)

$$\begin{bmatrix} y_t \\ x_t \end{bmatrix} = \begin{bmatrix} k + m \\ m \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_{t-1} \\ x_{t-1} \end{bmatrix} + \begin{bmatrix} u_t + \varepsilon_t \\ \varepsilon_t \end{bmatrix}$$

o in forma VECM

$$\begin{aligned} \begin{bmatrix} \Delta y_t \\ \Delta x_t \end{bmatrix} &= \begin{bmatrix} k + m \\ m \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y_{t-1} \\ x_{t-1} \end{bmatrix} + \begin{bmatrix} u_t + \varepsilon_t \\ \varepsilon_t \end{bmatrix} = \\ &= \begin{bmatrix} k + m \\ m \end{bmatrix} + \begin{bmatrix} -1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} y_{t-1} \\ x_{t-1} \end{bmatrix} + \begin{bmatrix} u_t + \varepsilon_t \\ \varepsilon_t \end{bmatrix} = \\ &= \mu_0 + \alpha \beta' \begin{bmatrix} y_{t-1} \\ x_{t-1} \end{bmatrix} + \eta_t = \mu_0 + \alpha z_{t-1} + \eta_t, \end{aligned}$$

dove β è il vettore di cointegrazione e α è il vettore dei “loading” o “aggiustamenti”.

Possiamo ora considerare tre casi possibili:

1. $m \neq 0$: In questo caso x_t ha un trend, come abbiamo appena visto; ne consegue che anche y_t segue un trend lineare perché in media si mantiene a una distanza di k da x_t . Il vettore μ_0 non ha restrizioni. Questo è il caso predefinito per il comando `vecm` di `gretl`.
2. $m = 0$ e $k \neq 0$: In questo caso, x_t non ha un trend, e di conseguenza neanche y_t . Tuttavia, la distanza media tra y_t e x_t è diversa da zero. Il vettore μ_0 è dato da

$$\mu_0 = \begin{bmatrix} k \\ 0 \end{bmatrix}$$

che è non nullo, quindi il VECM mostrato sopra ha un termine costante. La costante, tuttavia è soggetta alla restrizione che il suo secondo elemento deve essere pari a 0. Più in generale, μ_0 è un multiplo del vettore α . Si noti che il VECM potrebbe essere scritto anche come

$$\begin{bmatrix} \Delta y_t \\ \Delta x_t \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & -1 & -k \end{bmatrix} \begin{bmatrix} y_{t-1} \\ x_{t-1} \\ 1 \end{bmatrix} + \begin{bmatrix} u_t + \varepsilon_t \\ \varepsilon_t \end{bmatrix}$$

che incorpora l'intercetta nel vettore di cointegrazione. Questo è il caso di “costante vincolata”; può essere specificato nel comando `vecm` di `gretl` usando l'opzione `--rc`.

3. $m = 0$ e $k = 0$: Questo caso è il più vincolante: chiaramente, né x_t né y_t hanno un trend, e la loro distanza media è zero. Anche il vettore μ_0 vale 0, quindi questo caso può essere chiamato “senza costante”. Questo caso è specificato usando l'opzione `--nc` con `vecm`.

Nella maggior parte dei casi, la scelta tra le tre possibilità si basa su un misto di osservazione empirica e di ragionamento economico. Se le variabili in esame sembrano seguire un trend lineare, è opportuno non imporre alcun vincolo all'intercetta. Altrimenti occorre chiedersi se ha senso specificare una relazione di cointegrazione che includa un'intercetta diversa da zero. Un esempio appropriato potrebbe essere la relazione tra due tassi di interesse: in generale questi non hanno un trend, ma il VAR potrebbe comunque avere un'intercetta perché la differenza tra i due (lo “spread” sui tassi d'interesse) potrebbe essere stazionaria attorno a una media diversa da zero (ad esempio per un premio di liquidità o di rischio).

L'esempio precedente può essere generalizzato in tre direzioni:

1. Se si considera un VAR di ordine maggiore di 1, l'algebra si complica, ma le conclusioni sono identiche.
2. Se il VAR include più di due variabili endogene, il rango di cointegrazione r può essere maggiore di 1. In questo caso α è una matrice con r colonne e il caso con la costante vincolata comporta che μ_0 sia una combinazione lineare delle colonne di α .
3. Se si include un trend lineare nel modello, la parte deterministica del VAR diventa $\mu_0 + \mu_1 t$. Il ragionamento è praticamente quello visto sopra, tranne per il fatto che l'attenzione è ora posta su μ_1 invece che su μ_0 . La controparte del caso con “costante vincolata” discusso sopra è un caso con “trend vincolato”, così che le relazioni di cointegrazione includono un trend, ma la differenza prima delle variabili in questione no. Nel caso di un trend non vincolato, il trend appare sia nelle relazioni di cointegrazione sia nelle differenze prime, il che corrisponde alla presenza di un trend quadratico nelle variabili (esprese in livelli). Questi due casi sono specificati rispettivamente con le opzioni `--crt` e `--ct` del comando `vecm`.

Capitolo 13. Opzioni, argomenti e percorsi

gretl

`gretl` (in MS Windows, `gretlw32.exe`)¹.

— Apre il programma e aspetta l'input dall'utente.

`gretl filedati`

— Avvia il programma caricando nello spazio di lavoro il file di dati indicato. Il file può essere nel formato interno di `gretl`, in formato CSV, o BOX1 (si veda il [Capitolo 4](#)). Il programma cercherà di riconoscere il formato del file e di trattarlo in modo appropriato. Si veda anche [la Sezione Ricerca dei percorsi](#) per quanto riguarda la ricerca dei percorsi.

`gretl --help` (o `gretl -h`)

— Mostra un breve riepilogo delle modalità di uso ed esce.

`gretl --version` (o `gretl -v`)

— Mostra informazioni sulla versione del programma ed esce.

`gretl --english` (o `gretl -e`)

— Forza l'utilizzo della versione inglese invece di quella italiana.

`gretl --run file` (o `gretl -r file`)

— Avvia il programma e apre una finestra che mostra il file di comandi indicato, pronto per essere eseguito. Si veda [la Sezione Ricerca dei percorsi](#) per quanto riguarda la ricerca dei percorsi.

`gretl --db database` (o `gretl -d database`)

— Avvia il programma e apre una finestra che mostra il database indicato. Se i file del database (il file `.bin` e il relativo file `.idx`, si veda [la Sezione Database binari nel Capitolo 4](#)) non sono nella directory predefinita dei database, occorre specificare il percorso completo dei file.

`gretl --dump` (o `gretl -c`)

— Scrive le informazioni di configurazione del programma in un file di testo (il nome del file è mostrato nello standard output). Può essere utile nella risoluzione di problemi di funzionamento.

Varie funzionalità di `gretl` sono configurabili nel menù File, Preferenze menu.

§ La directory base per i file condivisi di `gretl`.

§ La directory base dell'utente per i file relativi a `gretl`.

§ Il comando per eseguire `gnuplot`.

§ Il comando per eseguire GNU R (si veda [Appendice D](#)).

§ Il comando per visualizzare i file DVI di TeX.

§ La directory in cui iniziare a cercare i database in formato interno di `gretl`.

§ La directory in cui iniziare a cercare i database in formato RATS 4.

§ Il nome host del server dei database di `gretl` a cui connettersi.

§ L'indirizzo IP e il numero di porta del proxy HTTP da usare per contattare il server dei database, se necessario (se si è dietro a un firewall).

§ La calcolatrice e l'editor di testi da eseguire dalla barra dei comandi.

§ Il tipo di carattere a spazio fisso da usare nelle finestre dei risultati di `gretl`.

1. In Linux viene installato uno script "wrapper" chiamato `gretl`. Questo script controlla se sia impostata la variabile di ambiente `DISPLAY` e in caso positivo lancia la versione grafica del programma, altrimenti lancia la versione a riga di comando, `gretlcli`.

§ Il tipo di carattere da usare nei menù e negli altri messaggi (nota: questo elemento non è presente se `gretl` è compilato per il desktop `gnome`, visto che la scelta dei caratteri è gestita in modo centralizzato da `gnome`.)

Ci sono anche alcune caselle di testo: selezionando la casella “esperto”, vengono disabilitati alcuni messaggi di avvertimento che altrimenti verrebbero mostrati; selezionando la casella “Avvisa in caso di aggiornamenti di `gretl`”, `gretl` cercherà di connettersi al server dei database appena avviato; de-selezionando “Mostra la barra degli strumenti di `gretl`”, la barra degli strumenti sarà disabilitata. Se l'impostazione locale del proprio sistema non usa il punto (“.”) come separatore delle cifre decimali, de-selezionando “Usa le impostazioni locali per il punto decimale”, `gretl` userà comunque il carattere punto.

Infine, ci sono due scelte binarie: nella sezione “Percorso per Apri/Salva” è possibile impostare il percorso usato in modo predefinito da `gretl` quando si chiede di aprire o salvare un file: o la directory `gretl` dell'utente o la directory di lavoro attuale. Nella sezione “File di dati”, è possibile impostare l'estensione predefinita da aggiungere ai nomi dei file di dati. L'estensione standard è `.gdt`, ma è possibile impostarla a `.dat`, che era lo standard nelle versioni precedenti del programma. Se si imposta l'estensione predefinita a `.dat`, i file di dati saranno salvati nel formato “tradizionale” (si veda il [Capitolo 4](#)). Inoltre, nella sezione “File di dati” è possibile scegliere l'azione da associare alla piccola icona a forma di cartella presente sulla barra degli strumenti: se deve mostrare un elenco dei file di dati associati al libro di Ramanathan, o di quelli associati al libro di Wooldridge.

Nella sezione “Generali” è possibile scegliere l'algoritmo usato da `gretl` per calcolare le stime dei minimi quadrati. La scelta predefinita è la decomposizione di Cholesky, che è veloce, relativamente economica in termini di requisiti di memoria, e accurata per la maggior parte delle applicazioni. L'alternativa è la decomposizione QR, che è computazionalmente più impegnativa e richiede una maggiore quantità di memoria, ma è più accurata. È poco probabile dover avere bisogno dell'accuratezza ulteriore fornita dalla decomposizione QR, a meno che non si abbia a che fare con dati molto mal-condizionati e si sia interessati a valori dei coefficienti e degli errori standard precisi per più di 7 cifre decimali².

Le impostazioni scelte in questo modo sono salvate in modo diverso a seconda del sistema. In MS Windows sono scritte nel registro di Windows. Nel desktop di `gnome` sono scritte nel file `.gnome/gretl` nella home directory dell'utente. Altrimenti vengono scritte in un file chiamato `.gretlrc` nella home directory dell'utente.

gretlcli

`gretlcli`

— Apre il programma e attende l'input dall'utente.

`gretlcli filedati`

— Avvia il programma caricando nello spazio di lavoro il file di dati indicato. Il file può essere nel formato interno di `gretl`, in formato CSV, o BOX1 (si veda il [Capitolo 4](#)). Il programma cercherà di riconoscere il formato del file e di trattarlo in modo appropriato. Si veda anche [la Sezione Ricerca dei percorsi](#) per quanto riguarda la ricerca dei percorsi.

`gretlcli --help` (o `gretlcli -h`)

— Mostra un breve riepilogo delle modalità di uso ed esce.

`gretlcli --version` (o `gretlcli -v`)

— Mostra informazioni sulla versione del programma ed esce.

`gretlcli --pvalue` (o `gretlcli -p`)

— Avvia il programma in una modalità in cui è possibile determinare interattivamente i p-value per alcune statistiche comuni.

`gretlcli --english` (o `gretlcli -e`)

2. L'uso della decomposizione QR è attivabile anche impostando la variabile di ambiente `GRET_USE_QR` a un qualsiasi valore non nullo.

— Forza l'utilizzo della versione inglese invece di quella italiana.

`gretlcli --run file` (o `gretlcli -r file`)

— Esegue i comandi contenuti nel *file* e aspetta ulteriore input dalla riga di comando. Si veda [la Sezione Ricerca dei percorsi](#) per quanto riguarda la ricerca dei percorsi.

`gretlcli --batch file` (o `gretlcli -b file`)

— Esegue i comandi contenuti nel *file* ed esce; tipicamente questa opzione viene usata reindirizzando i risultati del comando a un file. Si veda [la Sezione Ricerca dei percorsi](#) per quanto riguarda la ricerca dei percorsi.

Se si usano le opzioni `--run` e `--batch`, il file richiamato deve indicare un file di dati da aprire. Ciò può essere fatto usando il comando `open` nel file, ma per compatibilità con il programma originario ESL di Ramanathan è disponibile anche un altro metodo (ESL non ha il comando `open`). Una riga del tipo:

```
(* ! miofile.gdt *)
```

farà aprire a `gretlcli` il file `miofile.gdt`, ma sarà ignorata come commento dal programma originario ESL. Si noti la sintassi: c'è esattamente uno spazio tra il marcatore di inizio commento (`*` e il carattere `!`; c'è esattamente uno spazio tra il carattere `!` e il nome del file di dati.

Un'altra funzionalità permette a `gretl` e a `gretlcli` di recuperare informazioni sui file di dati a partire dai “file di esempio” di ESL inclusi nel pacchetto `gretl`. Un file di esempio inizia tipicamente così:

```
(* PS4.1, using data file DATA4-1, for reproducing Table 4.2 *)
```

Viene usata questa regola: se un file di input inizia con il marcatore di commento, viene cercata una stringa che inizia per `DATA` (maiuscola); se viene trovata, viene estratta la parte della stringa compresa tra la `D` fino allo spazio o alla virgola successivi, viene convertita in caratteri minuscoli e la si usa come nome del file di dati da aprire.

Ricerca dei percorsi

Quando si indica il nome di un file di dati o di comandi a `gretl` o `gretlcli` sulla riga di comando (si veda [la Sezione `gretl`](#) e [la Sezione `gretlcli`](#)), il file viene cercato nel modo seguente:

1. “Esattamente come viene specificato”. Ossia, nella directory di lavoro attuale, o, se viene indicato un percorso completo, nella posizione indicata.
2. Nella directory `gretl` dell'utente (si veda la [Tabella 13-1](#) per i valori predefiniti).
3. In qualsiasi directory immediatamente contenuta nella directory `gretl` dell'utente.
4. Nel caso di un file di dati, la ricerca continua nella directory principale dei dati di `gretl`. Nel caso di un file di comandi, la ricerca continua nella directory principale dei file di comandi. Si veda la [Tabella 13-1](#) per i valori predefiniti.
5. Nel caso dei file di dati, la ricerca continua in tutte le directory immediatamente contenute nella directory principale dei dati.

Tabella 13-1. Impostazioni predefinite dei percorsi

	Linux	MS Windows
Directory <code>gretl</code> dell'utente	<code>\$HOME/gretl</code>	<code>PREFISSO\gretl\user</code>
Directory di sistema dei dati	<code>PREFISSO/share/gretl/data</code>	<code>PREFISSO\gretl\data</code>

Linux**MS Windows**

Directory di sistema dei file di comandi PREFIXSO/share/gretl/scripts PREFIXSO\gretl\scripts

Nota: PREFIXSO indica la directory base scelta al momento dell'installazione di gretl.

Non è quindi necessario specificare il percorso completo per un file di dati o di comandi, a meno che non si voglia fare a meno del meccanismo di ricerca automatico (ciò è valido anche per gretlcli quando si indica un nome di file come argomento per il comando open o il comando run)

Quando un file di comandi contiene l'istruzione di aprire un file di dati, l'ordine di ricerca per il file è quello descritto sopra, con l'eccezione che il file viene cercato anche nella directory che contiene lo script, immediatamente dopo aver tentato di cercare il file di dati "esattamente come viene specificato".

MS Windows

In MS Windows le informazioni di configurazione per gretl e gretlcli sono scritte nel registro di Windows. Quando gretl è installato per la prima volta, viene creata un'appropriata serie di voci di registro; successivamente le impostazioni possono essere modificate attraverso il menù File, Preferenze di gretl. Se occorre eseguire delle modifiche manuali dei valori, le voci di registro possono essere trovate (usando il programma standard di Windows regedit.exe) sotto Software\gretl in HKEY_CLASSES_ROOT (la directory principale di gretl e il comando per invocare gnuplot) e in HKEY_CURRENT_USER (tutte le altre variabili configurabili).

Capitolo 14. Guida ai comandi

Introduzione

I comandi descritti in questa guida possono essere eseguiti nella versione a riga di comando del programma, ma anche in quella con interfaccia grafica (GUI): possono essere inseriti in un file “script” da eseguire o nel “terminale di gretl”. Nella maggior parte dei casi, la sintassi presentata nella guida è valida anche per inserire i comandi nei campi testuali delle finestre di dialogo dell’interfaccia grafica (si veda anche l’aiuto online di gretl), ricordandosi che in quest’ultimo caso *non* occorre inserire anche il nome del comando nella finestra di dialogo: esso è già sottointeso dal contesto. Un’altra differenza è che le opzioni di alcuni comandi non possono essere inserite nei campi delle finestre di dialogo: di solito, comunque, sono disponibili delle voci di menù che ne replicano l’effetto.

La guida mostra le opzioni in “formato lungo”, costituito da una parola facilmente memorizzabile preceduta da due trattini, ma esiste anche un formato abbreviato, mostrato nella [Tabella 14-4](#).

La guida utilizza le seguenti convenzioni:

- § Il carattere a larghezza fissa viene usato per indicare ciò che l’utente deve scrivere e per i nomi delle variabili.
- § I termini in *corsivo* sono dei segnaposto da sostituire con termini specifici, ad esempio si potrebbe scrivere reddito invece del generico *variabile-x*.
- § Il costrutto [*arg*] indica che l’argomento *arg* è opzionale: è possibile fornirlo oppure no (in ogni caso, le parentesi quadre non vanno scritte).
- § La frase “comando di stima” indica un comando che genera stime per un certo modello, ad esempio `ols`, `ar` o `wls`.

I capitoli e i paragrafi citati sono quelli di Ramanathan (2002).

Comandi di gretl

add

Argomento: *lista-variabili*

Opzioni: --vcv (mostra la matrice di covarianza)
 --quiet (non mostra le stime del modello aumentato)
 --silent (non mostra nulla)

Esempi: add 5 7 9
 add xx yy zz --quiet

Va invocato dopo un comando di stima. Aggiunge al modello precedente le variabili nella *lista-variabili* e stima il nuovo modello. Viene eseguito anche un test per la significatività congiunta delle variabili aggiunte: nel caso di stime OLS, la statistica test è la F , negli altri casi è il chi-quadro asintotico di Wald. Un p-value inferiore a 0.05 indica che i coefficienti sono congiuntamente significativi al livello del 5 per cento.

Se viene usata l’opzione `--quiet` viene mostrato solo il risultato del test per la significatività congiunta delle variabili aggiunte, altrimenti vengono mostrate anche le stime per il modello aumentato. Nell’ultimo caso, l’opzione `--vcv` mostra anche la matrice di covarianza dei coefficienti. Usando l’opzione `--silent`, non viene mostrato nulla; tuttavia i risultati del test possono essere recuperati usando le variabili speciali `$test` e `$pvalue`.

Accesso dal menù: Finestra del modello, /Test/ADD - Aggiungi variabili

addobs

Argomento: *numero_osservazioni*

Esempio: `addobs 10`

Aggiunge il numero indicato di osservazioni alla fine del dataset, principalmente a scopo di previsione. I valori della maggior parte delle variabili saranno impostati a "mancante", ma alcune variabili deterministiche verranno riconosciute ed estese; ad esempio, un semplice trend lineare o delle dummy periodiche.

Il comando non è disponibile se sul dataset è impostato un sotto-campione ottenuto con una condizione Booleana.

Accesso dal menù: /Data/Aggiungi osservazioni

addto

Argomenti: *id-modello lista-variabili*

Opzione: `--quiet` (non mostra le stime del modello aumentato)

Esempio: `addto 2 5 7 9`

Funziona come il comando `add`, con l'eccezione che occorre specificare il modello precedente (usando il suo numero identificativo, che è mostrato all'inizio dei risultati del modello) da prendere come base per l'aggiunta delle variabili. L'esempio precedente aggiunge le variabili numero 5, 7 e 9 al modello 2.

Accesso dal menù: Finestra del modello, /Test/ADD - Aggiungi variabili

adf

Argomenti: *ordine nome-variabile*

Opzioni: `--nc` (test senza costante)

`--c` (solo con la costante)

`--ct` (con costante e trend)

`--ctt` (con costante, trend e trend al quadrato)

`--verbose` (mostra i risultati della regressione)

Esempi: `adf 0 y`

`adf 2 y --nc --c --ct`

Calcola le statistiche per un gruppo di test Dickey-Fuller sulla variabile specificata, assumendo come ipotesi nulla che la variabile abbia una radice unitaria.

Per impostazione predefinita, vengono mostrate tre varianti del test: una basata su una regressione che contiene solo una costante, una che include la costante e un trend lineare, una che usa la costante e un trend quadratico. È possibile controllare le tre varianti specificando una o più opzioni.

In tutti i casi, la variabile dipendente è la differenza prima della variabile specificata, y , e la variabile dipendente più importante è il ritardo (di ordine uno) di y . Il modello è costruito in modo che il coefficiente della variabile ritardata y è pari a 1 meno la radice. Ad esempio, il modello con una costante può essere scritto come

$$(1 - L)y_t = \beta_0 + (1 - \alpha)y_{t-1} + \epsilon_t$$

Se l'ordine di ritardi, k , è maggiore di 0, ai regressori di ognuna delle regressioni calcolate per il test saranno aggiunti k ritardi della variabile dipendente.

Se l'ordine di ritardi è preceduto da un segno meno, ad esempio $-k$, verrà considerato come ordine massimo, e l'ordine da usare effettivamente sarà ricavato applicando la seguente procedura di test "all'indietro":

1. Stima la regressione Dickey-Fuller con k ritardi della variabile dipendente.
2. Se questo ordine di ritardi è significativo, esegue il test con l'ordine di ritardo k . Altrimenti, prova il test con $k = k - 1$; se $k = 0$, esegue il test con ordine di ritardo 0, altrimenti va al punto 1.

Durante il punto 2 spiegato sopra, "significativo" significa che la statistica t per l'ultimo ritardo abbia un p -value asintotico a due code per la distribuzione normale pari a 0.10 o inferiore.

I p -value per questo test sono basati su MacKinnon (1996). Il codice rilevante è incluso per gentile concessione dell'autore.

Accesso dal menù: /Variabile/Test Dickey-Fuller aumentato

append

Argomento: *file-dati*

Apri un file di dati e aggiunge il suo contenuto al dataset corrente, se i nuovi dati sono compatibili. Il programma cerca di riconoscere il formato del file di dati (interno, testo semplice, CSV, o BOX1).

Accesso dal menù: /File/Aggiungi dati

ar

Argomenti: *ritardi ; variabile-dipendente variabili-indipendenti*

Opzione: *--vcv* (mostra la matrice di covarianza)

Esempio: *ar 1 3 4 ; y 0 x1 x2 x3*

Calcola le stime parametriche usando la procedura iterativa generalizzata di Cochrane-Orcutt (si veda il Capitolo 9.5 di Ramanathan). La procedura termina quando le somme dei quadrati degli errori consecutivi non differiscono per più dello 0.005 per cento, oppure dopo 20 iterazioni.

ritardi è una lista di ritardi nei residui, conclusa da un punto e virgola. Nell'esempio precedente, il termine di errore è specificato come

$$u_t = \rho_1 u_{t-1} + \rho_3 u_{t-3} + \rho_4 u_{t-4} + e_t$$

Accesso dal menù: /Modello/Serie storiche/AR - Stima autoregressiva

arch

Argomenti: *ordine variabile-dipendente variabili-indipendenti*

Esempio: *arch 4 y 0 x1 x2 x3*

Testa il modello per l'eteroschedasticità condizionale autoregressiva (ARCH: Autoregressive Conditional Heteroskedasticity) dell'ordine specificato. Se la statistica del test LM ha un p -value inferiore a 0.10, viene anche effettuata la stima ARCH. Se la varianza prevista di qualche osservazione nella regressione ausiliaria non risulta positiva, viene usato il corrispondente residuo al quadrato. Viene quindi calcolata la stima con i minimi quadrati ponderati sul modello originale.

Si veda anche [garch](#).

Accesso dal menù: Finestra del modello, /Test/ARCH

arma

Argomenti: $p\ q$; *variabile-dipendente* [*variabili-indipendenti*]

Opzioni: --native (usa il plugin interno (predefinito))
 --x-12-arma (usa X-12-ARIMA per la stima)
 --verbose (mostra i dettagli delle iterazioni)
 --vcv (mostra la matrice di covarianza)
 --nc (non include l'intercetta)

Esempi: arma 1 2 ; y
 arma 2 2 ; y 0 x1 x2 --verbose
 arma 1 1 ; 1 0 ; y 0 x1 x2

Se non viene fornita una lista di *variabili-indipendenti*, stima un modello autoregressivo a media mobile (ARMA: Autoregressive, Moving Average) univariato. I valori interi p e q rappresentano rispettivamente gli ordini AR e MA. I valori interi opzionali P e Q rappresentano gli ordini AR e MA stagionali; essi sono rilevanti solo la frequenza dei dati è superiore a 1 (ad esempio, dati trimestrali o mensili).

Nel caso univariato la scelta predefinita include un'intercetta nel modello, ma questa può essere soppressa con l'opzione --nc. Se vengono aggiunte delle *variabili-indipendenti*, il modello diventa un ARMAX: in questo caso occorre indicare esplicitamente la costante se si desidera un'intercetta (come nel secondo degli esempi proposti).

Il funzionamento predefinito utilizza la funzione ARMA "interna" di gretl; nel caso di un modello ARMA univariato, è anche possibile usare X-12-ARIMA (se è stato installato il pacchetto X-12-ARIMA per gretl).

Le opzioni mostrate in precedenza possono anche essere combinate, tenendo conto che la matrice di covarianza non è disponibile se si usa X-12-ARIMA per la stima.

L'algoritmo interno di gretl per ARMA è in gran parte dovuto a Riccardo "Jack" Lucchetti: utilizza una procedura di massima verosimiglianza condizionale, implementata attraverso la stima iterata con minimi quadrati della regressione del prodotto esterno del gradiente (OPG). Si veda l'[Esempio 10-3](#) per la logica della procedura. I coefficienti AR (e quelli per gli eventuali regressori aggiuntivi) sono inizializzati usando una auto-regressione OLS, mentre i coefficienti MA sono inizializzati a zero.

Il valore AIC mostrato nei modelli ARMA è calcolato secondo la definizione usata in X-12-ARIMA, ossia

$$AIC = -2L + 2k$$

dove L è la log-verosimiglianza e k è il numero totale di parametri stimati. Il valore di "frequenza" mostrato insieme alle radici AR e MA è il valore di λ che risolve

$$z = re^{i2\pi\lambda}$$

dove z è la radice in questione e r è il suo modulo.

Accesso dal menù: /Variabile/Modello ARMA, /Modello/Serie Storiche/ARMAX

Accesso alternativo: Menù pop-up nella finestra principale (selezione singola)

boxplot

Argomento: *lista-variabili*

Opzione: `--notches` (mostra l'intervallo di confidenza al 90 per cento per la mediana)

Questo tipo di grafici (da Tukey e Chambers) mostra la distribuzione di una variabile. La "scatola" centrale (box) racchiude il 50 per cento centrale dei dati, ossia è delimitato dal primo e terzo quartile. I "baffi" (whiskers) si estendono fino ai valori minimo e massimo. Una linea trasversale sulla scatola indica la mediana.

Nel caso dei grafici a tacca ("notches"), una tacca indica i limiti dell'intervallo di confidenza approssimato al 90 per cento per la mediana, ottenuto col metodo bootstrap.

Dopo ogni variabile specificata nel comando boxplot, è possibile aggiungere un'espressione booleana tra parentesi per limitare il campione per la variabile in questione. Occorre inserire uno spazio tra il nome o il numero della variabile e l'espressione. Si supponga di avere dati sui salari di uomini e donne e di avere una variabile dummy GENERE che vale 1 per gli uomini e 0 per le donne. In questo caso, è possibile generare dei boxplot comparativi usando la seguente *lista-variabili*:

```
salario (GENERE=1) salario (GENERE=0)
```

Alcuni dettagli del funzionamento dei boxplot di gretl possono essere controllati attraverso un file testuale chiamato `.boxplotrc`. Per ulteriori dettagli, si veda [la Sezione Boxplot nel Capitolo 8](#).

Accesso dal menù: /Dati/Grafico delle variabili/Boxplot

chow

Argomento: *osservazione*

Esempi: `chow 25`

`chow 1988:1`

Va eseguito dopo una regressione OLS. Crea una variabile dummy che vale 1 a partire dal punto di rottura specificato da *osservazione* fino alla fine del campione, 0 altrove; inoltre crea dei termini di interazione tra questa dummy e le variabili indipendenti originali, stima una regressione che include questi termini e calcola una statistica *F*, prendendo la regressione aumentata come non vincolata e la regressione originale come vincolata. La statistica è appropriata per testare l'ipotesi nulla che non esista un break strutturale in corrispondenza del punto di rottura specificato.

Accesso dal menù: Finestra del modello, /Test/CHOW

coeffsum

Argomento: *lista-variabili*

Esempio: `coeffsum xt xt_1 xr_2`

Deve essere usato dopo una regressione. Calcola la somma dei coefficienti delle variabili nella *lista-variabili* e ne mostra l'errore standard e il p-value per l'ipotesi nulla che la loro somma sia zero.

Si noti la differenza tra questo test e [omit](#), che assume come ipotesi nulla l'uguaglianza a zero di *tutti* i coefficienti di un gruppo di variabili indipendenti.

Accesso dal menù: Finestra del modello, /Test/Somma dei coefficienti

coint

Argomenti: *ordine variabile-dipendente variabili-indipendenti*
 Opzione: `--nc` (non include la costante)
 Esempio: `coint 4 y x1 x2`

Il test di cointegrazione di Engle-Granger svolge dei test Dickey-Fuller aumentati sull'ipotesi nulla che ognuna delle variabili elencate abbia una radice unitaria, usando l'ordine di ritardi specificato. Viene stimata la regressione di cointegrazione e viene eseguito un test ADF sui residui di questa regressione, mostrando anche la statistica di Durbin-Watson per la regressione di cointegrazione.

I *pvalue* per questo test si basano su MacKinnon (1996). Il codice relativo è stato incluso per gentile concessione dell'autore.

La regressione di cointegrazione di solito contiene una costante; se si vuole escluderla, basta usare l'opzione `--nc`.

Accesso dal menù: /Modello/Serie storiche/Test di cointegrazione/Engle-Granger

coint2

Argomenti: *ordine variabile-dipendente variabili-indipendenti*
 Opzioni: `--nc` (senza costante)
`--rc` (costante vincolata)
`--crt` (costante e trend vincolato)
`--ct` (costante e trend non vincolato)
`--seasonals` (include dummy stagionali centrate)
`--verbose` (mostra i dettagli delle regressioni ausiliarie)
 Esempi: `coint2 2 y x`
`coint2 4 y x1 x2 --verbose`
`coint2 3 y x1 x2 --rc`

Esegue il test di Johansen per la cointegrazione tra le variabili elencate per l'ordine specificato. I valori critici sono calcolati con l'approssimazione gamma di J. Doornik (Doornik, 1998). Per i dettagli su questo test, si veda Hamilton, *Time Series Analysis* (1994), Cap. 20.

L'inclusione di trend deterministici nel modello è controllata dalle opzioni del comando. Se non si indica alcuna opzione, viene inclusa una "costante non vincolata", che permette la presenza di un'intercetta diversa da zero nelle relazioni di cointegrazione e di un trend nei livelli delle variabili endogene. Nella letteratura originata dal lavoro di Johansen (si veda ad esempio il suo libro del 1995), si fa riferimento a questo come al "caso 3". Le prime quattro opzioni mostrate sopra, che sono mutualmente esclusive, producono rispettivamente i casi 1, 2, 4 e 5. Il significato di questi casi e i criteri per scegliere tra di essi sono spiegati in [Capitolo 12](#).

L'opzione `--seasonals`, che può accompagnare una qualsiasi delle altre opzioni, specifica l'inclusione di un gruppo di variabili dummy stagionali centrate. Questa opzione è disponibile solo per dati trimestrali o mensili.

La tabella seguente fornisce un esempio di interpretazione dei risultati del test nel caso di 3 variabili. H_0 denota l'ipotesi nulla, H_1 l'ipotesi alternativa e c il numero delle relazioni di cointegrazione.

Rango	Test traccia		Test Lmax	
	H_0	H_1	H_0	H_1
0	$c = 0$	$c = 3$	$c = 0$	$c = 1$
1	$c = 1$	$c = 3$	$c = 1$	$c = 2$
2	$c = 2$	$c = 3$	$c = 2$	$c = 3$

Si veda anche il comando [vecm](#).

Accesso dal menù: /Modello/Serie storiche/Test do cointegrazione/Johansen

corc

Argomenti: *variabile-dipendente variabili-indipendenti*

Opzione: `--vcv` (mostra la matrice di covarianza)

Esempio: `corc 1 0 2 4 6 7`

Calcola le stime dei parametri usando la procedura iterativa di Cochrane-Orcutt (si veda il Capitolo 9.4 di Ramanathan). La procedura si ferma quando le stime consecutive del coefficiente di autocorrelazione differiscono per meno di 0.001, oppure dopo 20 iterazioni.

Accesso dal menù: /Modello/Serie storiche/Cochrane-Orcutt

corr

Argomento: `[ lista-variabili ]`

Esempio: `corr y x1 x2 x3`

Mostra le coppie di coefficienti di correlazione per la variabili date nella *lista-variabili*, o per tutte le variabili del dataset se non viene specificata alcuna *lista-variabili*.

Accesso dal menù: /Dati/Matrice di correlazione

Accesso alternativo: Menù pop-up nella finestra principale (selezione multipla)

corrgm

Argomenti: *variable [max-ritardo]*

Esempio: `corrgm x 12`

Mostra i valori della funzione di autocorrelazione per la *variabile* specificata (dal nome o dal numero). Si veda Ramanathan, Capitolo 11.7. È definita come $\rho(u_t, u_{t-s})$ dove u_t è la t -esima osservazione della variabile u e s è il numero dei ritardi.

Vengono mostrate anche le autocorrelazioni parziali, ossia al netto dell'effetto dei ritardi intermedi. Il comando produce anche un grafico del correlogramma e mostra la statistica Q di Box-Pierce per testare l'ipotesi nulla che la serie sia "white noise" (priva di autocorrelazione). La statistica si distribuisce asintoticamente come chi-quadro con gradi di libertà pari al numero di ritardi specificati.

Se viene specificato un valore *max-ritardo*, la lunghezza del correlogramma viene limitata al numero di ritardi specificato, altrimenti viene scelta automaticamente.

Accesso dal menù: /Variabile/Correlogramma

Accesso alternativo: Menù pop-up nella finestra principale (selezione singola)

criteria

Argomenti: *ess T k*

Esempio: `criteria 23.45 45 8`

Calcola il criterio di informazione di Akaike (AIC) e il criterio di informazione bayesiana di Schwarz (BIC), dati *ess* (somma dei quadrati degli errori), il numero delle osservazioni (T)

e quello dei coefficienti (k). T , k e ess possono essere valori numerici o nomi di variabili definite in precedenza.

critical

Argomenti: *distribuzione param1 [param2]*

Esempi: `critical t 20`
`critical X 5`
`critical F 3 37`

Se la *distribuzione* è t , X o F , mostra i valori critici per le distribuzioni t di student, chi-quadro o F , per i valori di significatività usuali, con i gradi di libertà specificati, dati da *param1* per la t e la chi-quadro, o *param1* e *param2* per la F . Se la *distribuzione* è d , mostra i valori superiore e inferiore della statistica di Durbin-Watson al livello di significatività del 5 per cento, per il dato numero di osservazioni *param1*, e per l'intervallo da 1 a 5 variabili esplicative.

Accesso dal menù: /Utilità/Tavole statistiche

cusum

Va eseguito dopo la stima di un modello OLS. Esegue il test CUSUM per la stabilità dei parametri. Viene calcolata una serie di errori di previsione (scalati) per il periodo successivo, attraverso una serie di regressioni: la prima usa le prime k osservazioni e viene usata per generare la previsione della variabile dipendente per l'osservazione $k + 1$; la seconda usa le prime $k + 1$ osservazioni per generare una previsione per l'osservazione $k + 2$ e così via (dove k è il numero dei parametri nel modello originale). Viene mostrata, anche graficamente, la somma cumulata degli errori scalati di previsione. L'ipotesi nulla della stabilità dei parametri è rifiutata al livello di significatività del 5 per cento se la somma cumulata va al di fuori delle bande di confidenza al 95 per cento.

Viene mostrata anche la statistica t di Harvey-Collier per testare l'ipotesi nulla della stabilità dei parametri. Si veda il Capitolo 7 di *Econometric Analysis* di Greene, per i dettagli.

Accesso dal menù: Finestra del modello, /Test/CUSUM

data

Argomento: *lista-variabili*

Legge le variabili nella *lista-variabili* da un database (gretl o RATS 4.0), che deve essere stato precedentemente aperto con il comando [open](#). Inoltre, prima di eseguire questo comando, va impostata una frequenza dei dati e un intervallo del campione, usando i comandi [setobs](#) e [smp1](#). Ecco un esempio completo:

```
open macrodat.rat
setobs 4 1959:1
smp1 ; 1999:4
data GDP_JP GDP_UK
```

Questi comandi aprono un database chiamato `macrodat.rat`, impostano un dataset trimestrale che inizia nel primo trimestre del 1959 e finisce nel quarto trimestre del 1999 e infine importano le serie `GDP_JP` e `GDP_UK`.

Se le serie da leggere hanno frequenza maggiore di quella impostata nel dataset, occorre specificare un metodo di compattamento, come mostrato di seguito

```
data (compact=average) LHUR PUNEW
```

I quattro metodi di compattamento disponibili sono “average” (usa la media delle osservazioni ad alta frequenza), “last” (usa l'ultima osservazione), “first” e “sum”.

Accesso dal menù: /File/Consulta database

delete

Argomento: [*lista-variabili*]

Rimuove dal dataset le variabili elencate (specificate tramite il nome o il numero). *Usare con cautela*: non viene chiesta conferma dell'operazione e le variabili con numeri identificativi più alti vengono ri-numerate.

Se non viene specificata alcuna *lista-variabili* viene eliminata dal dataset l'ultima variabile (quella col numero identificativo più alto).

Accesso dal menù: Pop-up nella finestra principale (selezione singola)

diff

Argomento: *lista-variabili*

Calcola la differenza prima di ogni variabile nella *lista-variabili* e la salva in una nuova variabile il cui nome è prefissato con d_. Quindi `diff x y` crea le nuove variabili $d_x = x(t) - x(t-1)$ e $d_y = y(t) - y(t-1)$.

Accesso dal menù: /Dati/Aggiungi variabili/Differenze

else

Si veda [if](#).

end

Termina un blocco di comandi di qualsiasi tipo. Ad esempio, `end system` termina un [system](#) (sistema di equazioni).

endif

Si veda [if](#).

endloop

Indica la fine di un ciclo (loop) di comandi. Si veda [loop](#).

eqnprint

Argomento: [*-f nomefile*]

Opzione: `--complete` (crea un documento completo)

Va eseguito dopo la stima di un modello. Stampa il modello stimato sotto forma di equazione LaTeX. Se viene specificato un nome di file usando l'opzione `-f`, il risultato viene scritto in quel file, altrimenti viene scritto in un file il cui nome ha la forma `equation_N.tex`, dove N è il numero di modelli stimati finora nella sessione in corso. Si veda anche [tabprint](#).

Usando l'opzione `--complete`, il file LaTeX è un documento completo, pronto per essere processato; altrimenti il file va incluso in un documento.

Accesso dal menù: Finestra del modello, /LaTeX

equation

Argomenti: *variabile-dipendente variabili-indipendenti*
 Esempio: `equation y x1 x2 x3 const`

Specifica un'equazione all'interno di un sistema di equazioni (si veda [system](#)). La sintassi per specificare un'equazione in un sistema SUR è la stessa usata ad esempio in [ols](#). Per un'equazione in un sistema con minimi quadrati a tre stadi, invece è possibile usare una specificazione simile a quella usata per OLS e indicare una lista di strumenti comuni usando l'istruzione `instr` (si veda ancora [system](#)), oppure si può usare la stessa sintassi di [tsls](#).

estimate

Argomenti: *nome_sistema stimatore*
 Opzioni: `--iterate` (itera fino alla convergenza)
`--no-df-corr` (nessuna correzione per i gradi di libertà)
`--geomean` (si veda oltre)
 Esempi: `estimate "Klein Model 1" method=fiml`
`estimate Sys1 method=sur`
`estimate Sys1 method=sur --iterate`

Esegue la stima di un sistema di equazioni, che deve essere stato definito in precedenza usando il comando [system](#). Per prima cosa va indicato il nome del sistema, racchiuso tra virgolette se contiene spazi, quindi il tipo di stimatore, preceduto dalla stringa `method=`. Gli stimatori disponibili sono: `ols`, `tsls`, `sur`, `3s1s`, `fiml` o `liml`.

Se al sistema in questione sono stati imposti dei vincoli (si veda il comando [restrict](#)), la stima sarà soggetta a tali vincoli.

Se il metodo di stima è `sur` o `3s1s` e viene usata l'opzione `--iterate`, lo stimatore verrà iterato. Nel caso di SUR, se la procedura converge, i risultati saranno stime di massima verosimiglianza. Invece l'iterazione della procedura dei minimi quadrati a tre stadi non produce in genere risultati di massima verosimiglianza a informazione completa. L'opzione `--iterate` viene ignorata con gli altri metodi di stima.

Se vengono scelti gli stimatori "equazione per equazione" `ols` o `tsls`, nel calcolo degli errori standard viene applicata in modo predefinito una correzione per i gradi di libertà, che può essere disabilitata usando l'opzione `--no-df-corr`. Questa opzione non ha effetti nel caso vengano usati altri stimatori, che non prevedono correzioni per i gradi di libertà.

La formula usata in modo predefinito per calcolare gli elementi della matrice di covarianza tra equazioni è

$$\hat{\sigma}_{i,j} = \frac{\hat{u}_i' \hat{u}_j}{T}$$

Se viene usata l'opzione `--geomean`, viene applicata una correzione per i gradi di libertà secondo la formula

$$\hat{\sigma}_{i,j} = \frac{\hat{u}_i' \hat{u}_j}{\sqrt{(T - k_i)(T - k_j)}}$$

dove i k indicano il numero di parametri indipendenti in ogni equazione.

fcast

Argomenti: *[oss-iniziale oss-finale] var-stima*
 Opzioni: `--dynamic` (crea previsioni dinamiche)

```

--static (crea previsioni statiche)
Esempi:  fcast 1997:1 2001:4 f1
          fcast fit2

```

Deve seguire un comando di stima. Calcola previsioni per l'intervallo specificato (o per il più lungo intervallo possibile, se non viene specificata alcuna *oss-iniziale* e *oss-finale*) e salva i valori nella variabile *var-stima*, che può essere stampata e rappresentata graficamente o con un diagramma ASCII. Le variabili indipendenti sono quelle del modello originale, non è possibile introdurre altre variabili. Se viene specificato un processo di errore autoregressivo, la previsione incorpora la frazione prevedibile del processo di errore.

La scelta tra previsione statica e dinamica è rilevante solo nel caso di modelli dinamici, che comprendono un processo di errore autoregressivo, o che comprendono uno o più valori ritardati della variabile dipendente come regressori. Si veda [fcasterr](#) per maggiori dettagli.

Accesso dal menù: Finestra del modello, /Dati modello/Previsioni

fcasterr

```

Argomenti:  oss-iniziale oss-finale
Opzioni:    --plot (mostra il grafico)
            --dynamic (crea previsioni dinamiche)
            --static (crea previsioni statiche)

```

Dopo aver stimato un modello è possibile usare questo comando per rappresentare i valori stimati su un intervallo di osservazioni specificato, insieme agli errori standard stimati per queste previsioni (a seconda del tipo di modello e dei dati disponibili) e agli intervalli di confidenza al 95 per cento.

La scelta tra previsione statica e dinamica è rilevante solo nel caso di modelli dinamici, che comprendono un processo di errore autoregressivo, o che comprendono uno o più valori ritardati della variabile dipendente come regressori. Le previsioni statiche sono per il periodo successivo, basate sui valori effettivi nel periodo precedente, mentre quelle dinamiche usano la regola della previsione a catena. Ad esempio, se la previsione per y nel 2008 richiede come input il valore di y nel 2007, non è possibile calcolare una previsione statica se non si hanno dati per il 2007. È possibile calcolare una previsione dinamica per il 2008 se si dispone di una precedente previsione per y nel 2007.

La scelta predefinita consiste nel fornire una previsione statica per ogni porzione dell'intervallo di previsione che fa parte dell'intervallo del campione su cui il modello è stato stimato, e una previsione dinamica (se rilevante) fuori dal campione. L'opzione *dynamic* richiede di produrre previsioni dinamiche a partire dalla prima data possibile, mentre l'opzione *static* richiede di produrre previsioni statiche anche fuori dal campione.

La natura degli errori standard della previsione (se disponibili) dipende dalla natura del modello e della previsione. Per i modelli lineari statici, gli errori standard sono calcolati seguendo il metodo delineato in Davidson and MacKinnon (2004); essi incorporano sia l'incertezza dovuta al processo d'errore, sia l'incertezza dei parametri (sintetizzata dalla matrice di covarianza delle stime dei parametri). Per modelli dinamici, gli errori standard della previsione sono calcolati solo nel caso di previsione dinamica, e non incorporano incertezza dei parametri. Per modelli non lineari, al momento non sono disponibili errori standard della previsione.

Accesso dal menù: Finestra del modello, /Dati modello/Previsioni con errori standard

fit

Una scorciatoia per *fcast*, deve seguire un comando di stima. Genera valori stimati per il campione corrente basati sull'ultima regressione e li salva nella serie *autofit*. Nel caso

di modelli di serie storiche, mostra anche un grafico dei valori stimati e di quelli effettivi della variabile dipendente rispetto al tempo.

freq

Argomento: *variabile*
 Opzioni: `--quiet` (non mostra l'istogramma)
`--gamma` (test per la distribuzione gamma)

Se non vengono indicate opzioni, mostra la distribuzione di frequenza per la *variabile* (indicata con il nome o il numero) e i risultati del test chi-quadro di Doornik-Hansen per la normalità.

Se si usa l'opzione `--quiet`, l'istogramma non viene mostrato. Usando l'opzione `--gamma`, al posto del test di normalità viene eseguito il test non parametrico di Locke per l'ipotesi nulla che la variabile segua la distribuzione gamma; si veda Locke (1976), Shapiro e Chen (2001).

In modalità interattiva viene mostrato anche un grafico della distribuzione.

Accesso dal menù: /Variabile/Distribuzione di frequenza

function

Argomento: *nome_funzione*

Apri un blocco di istruzioni che definiscono una funzione. Il blocco va chiuso con `end function`. Per i dettagli, si veda [Capitolo 11](#).

garch

Argomenti: *p q ; [P Q ;] variabile-dipendente [variabili-indipendenti]*
 Opzioni: `--robust` (errori standard robusti)
`--verbose` (mostra i dettagli delle iterazioni)
`--vcv` (mostra la matrice di covarianza)
`--arma-init` (parametri di varianza iniziale da ARMA)
 Esempi: `garch 1 1 ; y`
`garch 1 1 ; y 0 x1 x2 --robust`

Stima un modello GARCH (Generalized Autoregressive Conditional Heteroskedasticity) univariato, o, se sono specificate delle *variabili-indipendenti*, includendo delle variabili esogene. I valori interi *p* e *q* rappresentano gli ordini di ritardo nell'equazione della varianza condizionale.

$$h_t = \alpha_0 + \sum_{i=1}^q \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^p \beta_j h_{t-j}$$

L'algoritmo GARCH usato da gretl è in pratica quello di Fiorentini, Calzolari e Panattoni (1996), usato per gentile concessione del Professor Fiorentini.

Sono disponibili varie stime della matrice di covarianza dei coefficienti. Il metodo predefinito è quello dell'Hessiana; se si indica l'opzione `--robust` viene usata la matrice di covarianza QML (White). Altre possibilità (ad es. la matrice di informazione, o lo stimatore di Bollerslev-Wooldridge) possono essere specificate con il comando [set](#).

In modalità predefinita, le stime dei parametri di varianza sono inizializzate usando la varianza dell'errore non condizionale, ottenuta dalla stima OLS iniziale, per la costante,

e piccoli valori positivi per i coefficienti dei valori passati dell'errore al quadrato e per la varianza dell'errore. L'opzione `--arma-init` fa in modo che i valori iniziali per questi parametri siano ricavati da un modello ARMA iniziale, sfruttando la relazione tra GARCH e ARMA mostrata nel capitolo 21 di *Time Series Analysis* di Hamilton. In alcuni casi, questo metodo può aumentare le probabilità di convergenza.

Accesso dal menù: /Modello/Serie storiche/GARCH

genr

Argomenti: *nuova-variabile = formula*

Crea nuove variabili, di solito per mezzo di trasformazioni di variabili esistenti. Si veda anche [diff](#), [logs](#), [lags](#), [ldiff](#), [multiply](#) e [square](#) per le scorciatoie. Nel contesto di una formula `genr`, le variabili esistenti devono essere referenziate per nome, non per numero identificativo. La formula dev'essere una combinazione ben definita di nomi di variabile, costanti, operatori e funzioni (descritte oltre). Ulteriori dettagli su alcuni aspetti di questo comando si possono trovare in [Capitolo 5](#).

Questo comando può produrre come risultato una serie o uno scalare. Ad esempio, la formula `x2 = x * 2` produce una serie se la variabile `x` è una serie e uno scalare se `x` è uno scalare. Le formule `x = 0` e `mx = mean(x)` producono degli scalari. In alcune circostanze, può essere utile che un risultato scalare sia espanso in una serie o in un vettore: è possibile ottenere questo risultato usando `series` come "alias" per il comando `genr`. Ad esempio, `series x = 0` produce una serie con tutti i valori pari a 0. Allo stesso modo, è possibile usare `scalar` come alias per `genr`, ma non è possibile forzare un risultato vettoriale in uno scalare: con questa parola chiave si indica che il risultato *dovrebbe essere* uno scalare; se non lo è, viene emesso un messaggio di errore.

Quando una formula produce una serie o un vettore come risultato, l'intervallo su cui essi sono definiti dipende dall'impostazione attuale del campione. È quindi possibile definire una serie a pezzi, alternando l'uso dei comandi `smpl` e `genr`.

Gli *operatori aritmetici* supportati sono, in ordine di precedenza: $\wedge$ (esponenziale); $*$, $/$ e $\%$ (modulo o resto); $+$ e $-$.

Gli *operatori Booleani* disponibili sono (ancora in ordine di precedenza): $!$ (negazione), $\&$ (AND logico), $|$ (OR logico), $>$, $<$, $=$, $>=$ (maggiore o uguale), $<=$ (minore o uguale) e $\neq$ (disuguale). Gli operatori Booleani possono essere usati per costruire variabili dummy: ad esempio `(x > 10)` produce 1 se `x > 10`, 0 altrimenti.

Le costanti predefinite sono `pi` e `NA`. L'ultima rappresenta il codice per i valori mancanti: è possibile inizializzare una variabile con valori mancanti usando `scalar x = NA`.

Le *funzioni* supportate appartengono ai gruppi seguenti:

- § Funzioni matematiche standard: `abs`, `cos`, `exp`, `int` (parte intera), `ln` (logaritmo naturale: `log` è un sinonimo), `sin`, `sqrt` (radice quadrata). Tutte queste accettano un solo argomento, che può essere una serie o uno scalare.
- § Funzioni statistiche standard, che possono accettare un solo argomento e producono un risultato scalare: `max` (valore massimo in una serie), `min` (minimo), `mean` (media aritmetica), `median`, `var` (varianza) `sd` (deviazione standard), `sst` (somma dei quadrati delle deviazioni dalla media), `sum`.
- § Funzioni statistiche che accettano una serie con argomento e producono una serie o un vettore: `sort` (ordina una serie in ordine ascendente di grandezza), `cum` (somma cumulata).
- § Funzioni statistiche che accettano due serie come argomenti e producono uno scalare: `cov` (covarianza), `corr` (coefficiente di correlazione).
- § Funzioni statistiche speciali: `pvalue` (si veda oltre), `cnorm` (funzione di ripartizione normale standard), `dnorm` (funzione di densità normale standard), `resample` (ricampiona una

serie con rimpiazzo, utile per il bootstrap), `hpfilt` (componente di “ciclo” della serie), `bkfilt` (filtro passa-banda di Baxter-King).

- § Funzioni per serie storiche: `diff` (differenza prima), `ldiff` (log-differenza, ossia la differenza prima dei logaritmi naturali), `fracdiff` (differenza frazionale). Per generare ritardi della variabile `x`, si può usare la sintassi `x(-N)`, dove `N` rappresenta la lunghezza del ritardo desiderato; per generare anticipi si usi `x(+N)`.
- § Funzioni dataset, che producono una serie: `misszero` (sostituisce con zero il codice delle osservazioni mancanti in una serie); `zeromiss` (l'operazione inversa di `misszero`); `missing` (per ogni osservazione, vale 1 se l'argomento ha un valore mancante, 0 altrimenti); `ok` (l'opposto di `missing`).
- § Funzioni dataset, che producono uno scalare: `nobs` (riporta il numero di osservazioni valide in una serie), `firstobs` (riporta il numero della prima osservazione non mancante in una serie) `lastobs` (numero dell'ultima osservazione non mancante in una serie).
- § Numeri pseudo-casuali: `uniform` e `normal`. Queste funzioni non accettano argomenti e vanno scritte con parentesi vuote: `uniform()`, `normal()`. Esse creano serie pseudo-casuali estraendo i valori dalle distribuzioni uniforme (0-1) e normale standard. Si veda anche l'opzione `seed` del comando [set](#). Le serie dalla distribuzione uniforme sono generate usando il Mersenne Twister ¹; per le serie dalla distribuzione normale, viene usato il metodo di Box e Muller (1958), prendendo l'input dal Mersenne Twister.

Tutte le funzioni viste sopra, con l'eccezione di `cov`, `corr`, `pvalue`, `fracdiff`, `uniform` e `normal` hanno come unico argomento il nome di una variabile o un'espressione che si risolve in una variabile (ad es. `ln((x1+x2)/2)`). La funzione `pvalue` richiede gli stessi argomenti del comando [pvalue](#), ma in questo caso occorre separare gli argomenti con delle virgole. Questa funzione produce un p-value a una coda, che nel caso delle distribuzioni normale e *t* è la “coda corta”. Con la normale, ad esempio, sia 1.96 sia -1.96 daranno un risultato di circa 0.025. La funzione `fracdiff` richiede due argomenti: il nome della serie e una frazione nell'intervallo da -1 a 1.

`uniform()` e `normal()`, che non richiedono argomenti, producono delle serie pseudo-casuali estratte dalle distribuzioni uniforme (0-1) e normale standard (si veda anche l'opzione `seed` del comando [set](#)). Le serie dalla distribuzione uniforme sono generate usando il Mersenne Twister ²; per le serie dalla distribuzione normale, viene usato il metodo di Box e Muller (1958), prendendo l'input dal Mersenne Twister.

Oltre agli operatori e alle funzioni mostrati, ci sono alcuni usi speciali del comando `genr`:

- § `genr time` crea una variabile trend temporale (1,2,3,...) chiamata `time`. `genr index` fa la stessa cosa, ma chiamando la variabile `index`.
- § `genr dummy` crea una serie di variabili dummy a seconda della periodicità dei dati. Ad esempio, nel caso di dati trimestrali (periodicità 4) il programma crea `dummy_1`, che vale 1 nel primo trimestre e 0 altrove, `dummy_2` che vale 1 nel secondo trimestre e 0 altrove, e così via.
- § `genr paneldum` crea una serie di variabili dummy da usare in un dataset di tipo panel — si veda [panel](#).
- § Alcune variabili interne che vengono generate durante il calcolo di una regressione possono essere recuperate usando `genr`, nel modo seguente:

<code>\$ess</code>	somma dei quadrati degli errori
<code>\$rsq</code>	<i>R</i> -quadro
<code>\$T</code>	numero delle osservazioni usate
<code>\$df</code>	gradi di libertà
<code>\$trsq</code>	<i>TR</i> -quadro (ampiezza del campione moltiplicata per <i>R</i> -quadro)

1. Si veda Matsumoto e Nishimura (1998). L'implementazione è fornita da `glib`, se è disponibile, o dal codice C scritto da Nishimura e Matsumoto.

2. Si veda Matsumoto e Nishimura (1998). L'implementazione è fornita da `glib`, se è disponibile, o dal codice C scritto da Nishimura e Matsumoto.

<code>\$sigma</code>	errore standard dei residui
<code>\$aic</code>	criterio di informazione di Akaike
<code>\$bic</code>	criterio di informazione bayesiana di Schwarz
<code>\$lnl</code>	log-verosimiglianza (dove è applicabile)
<code>\$sigma</code>	errore standard dei residui
<code>coeff(variabile)</code>	coefficiente stimato per la <i>variabile</i>
<code>stderr(variabile)</code>	errore standard stimato per la <i>variabile</i>
<code>rho(i)</code>	coefficiente di autoregressione dei residui di ordine <i>i</i> -esimo
<code>vcv(x1,x2)</code>	covarianza stimata tra i coefficienti delle variabili <i>x1</i> e <i>x2</i>

Nota: nella versione a riga di comando del programma, i comandi `genr` che estraggono dati relativi al modello si riferiscono sempre al modello stimato per ultimo. Questo vale anche per la versione grafica del programma se si usa `genr` nel “terminale di gretl” o si immette una formula usando l’opzione “Definisci nuova variabile” nel menù Variabile della finestra principale. Usando la versione grafica, però, è possibile anche estrarre i dati da qualunque modello mostrato in una finestra (anche se non è il modello più recente) usando il menù “Dati modello” nella finestra del modello.

Le serie interne `uhat` e `yhat` contengono rispettivamente i residui e i valori stimati dell’ultima regressione.

Sono disponibili tre variabili dataset “interne”: `$nobs`, che contiene il numero di osservazioni nell’intervallo del campione attuale (che non è necessariamente uguale al valore di `$T`, il numero delle osservazioni usate per stimare l’ultimo modello), `$nvars`, che contiene il numero di variabili nel dataset (inclusa la costante), e `$pd`, che contiene la frequenza o la periodicità dei dati (ad esempio 4 per dati trimestrali).

Due scalari speciali interni, `$test` e `$pvalue`, contengono rispettivamente il valore e il *p*-value della statistica *test* che è stata generata dall’ultimo comando di *test* eseguito esplicitamente (ad es. `chow`). Si veda [Capitolo 5](#) per i dettagli.

La variabile `t` serve da indice per le osservazioni. Ad esempio, `genr dum = (t=15)` crea una variabile dummy che vale 1 per l’osservazione 15 e 0 altrove. La variabile `obs` è simile ma più flessibile: è possibile usarla per isolare alcune osservazioni indicandole con una data o un nome. Ad esempio, `genr d = (obs>1986:4)` o `genr d = (obs="CA")`. L’ultima forma richiede che siano definite delle etichette, da usare tra virgolette doppie, per le osservazioni.

È possibile estrarre dei valori scalari da una serie usando una formula `genr` con la sintassi *nome-variabile*[*osservazione*]. Il valore di *osservazione* può essere specificato con un numero o una data. Esempi: `x[5]`, `CPI[1996:01]`. Per i dati giornalieri occorre usare la forma *AAAA/MM/GG*, ad esempio `ibm[1970/01/23]`.

È possibile modificare una singola osservazione in una serie usando `genr`. Per farlo, occorre aggiungere un numero di osservazione o una data valida tra parentesi quadre al nome della variabile nel lato sinistro della formula. Ad esempio: `genr x[3] = 30` o `genr x[1950:04] = 303.7`.

La [Tabella 14-1](#) mostra vari esempi di utilizzo di `genr`, con alcune note esplicative; ecco qualche esempio di utilizzo delle variabili dummy:

§ Si supponga che `x` abbia valori 1, 2, o 3 e si desiderino tre variabili dummy, `d1 = 1` se `x = 1`, e 0 altrove, `d2 = 1` se `x = 2` e così via. Per crearle, basta usare i comandi:

```
genr d1 = (x=1)
genr d2 = (x=2)
genr d3 = (x=3)
```

§ Per creare `z = max(x,y)` usare

```

genr d = x>y
genr z = (x*d)+(y*(1-d))

```

Tabella 14-1. Esempi di utilizzo del comando **genr**

Formula	Commento
$y = x1^3$	x1 al cubo
$y = \ln((x1+x2)/x3)$	
$z = x>y$	$z(t) = 1$ se $x(t) > y(t)$, 0 altrove
$y = x(-2)$	x ritardata di 2 periodi
$y = x(+2)$	x anticipata di 2 periodi
$y = \text{diff}(x)$	$y(t) = x(t) - x(t-1)$
$y = \text{ldiff}(x)$	$y(t) = \log x(t) - \log x(t-1)$, il tasso di crescita istantaneo di x
$y = \text{sort}(x)$	ordina x in senso crescente e la salva in y
$y = -\text{sort}(-x)$	ordina x in senso decrescente
$y = \text{int}(x)$	tronca x e salva il valore intero in y
$y = \text{abs}(x)$	salva il valore assoluto di x
$y = \text{sum}(x)$	somma i valori di x escludendo i valori mancanti -999
$y = \text{cum}(x)$	cumulativa: $y_t = \sum_{\tau=1}^t x_\tau$
$aa = \$ess$	imposta aa uguale alla somma dei quadrati degli errori dell'ultima regressione
$x = \text{coeff}(sqft)$	estrae il coefficiente stimato per la variabile sqft nell'ultima regressione
$\rho4 = \rho(4)$	estrae il coefficiente di autoregressione del quarto ordine dall'ultimo modello (presume un modello ar model)
$cvx1x2 = \text{vcv}(x1, x2)$	estrae il coefficiente di covarianza stimato tra le variabili x1 e x2 dall'ultimo modello
$\text{foo} = \text{uniform}()$	variabile pseudo-casuale uniforme nell'intervallo 0-1
$\text{bar} = 3 * \text{normal}()$	variabile pseudo-casuale normale con $\mu = 0$, $\sigma = 3$
$\text{samp} = \text{ok}(x)$	vale 1 per le osservazioni dove il valore di x non è mancante.

Accesso dal menù: /Variabile/Definisci nuova variabile

Accesso alternativo: Menù pop-up nella finestra principale

gnuplot

Argomenti: *variabili-y variabile-x [variabile-dummy]*

Opzioni: *--with-lines* (usa linee invece che punti)
--with-impulses (usa linee verticali)
--suppress-fitted (non mostra le stime minimi quadrati)
--dummy (si veda sotto)

Esempi: *gnuplot y1 y2 x*
gnuplot x time --with-lines

```
gnuplot wages educ gender --dummy
```

Senza l'opzione `--dummy` mostra un grafico delle *variabili-y* rispetto alla *variabile-x*. Con `--dummy`, la *variabile-y* è rappresentata rispetto alla *variabile-x* con i punti colorati diversamente a seconda del valore della *variabile-dummy* (1 o 0).

La variabile `time` si comporta in modo speciale: se non esiste già, verrà generata automaticamente.

In modalità interattiva il risultato è mostrato immediatamente. In modalità “batch”, viene scritto un file di comandi gnuplot, chiamato `gpttmpN.plt`, a partire da `N = 01`; il grafico vero e proprio può essere generato usando il programma `gnuplot` (su MS Windows: `wgnuplot`).

È disponibile un'ulteriore opzione per questo comando: dopo la specificazione delle variabili e le eventuali opzioni, è possibile aggiungere direttamente dei comandi gnuplot per modificare l'aspetto visivo del grafico (ad esempio, impostando il titolo e o gli intervalli degli assi). Questi comandi aggiuntivi vanno inclusi tra parentesi graffe e ogni comando va separato con un punto e virgola; è possibile usare una barra rovesciata (`\`) per continuare un gruppo di comandi gnuplot sulla riga successiva. Ecco un esempio della sintassi:

```
{ set title 'Il mio titolo'; set yrange [0:1000]; }
```

Accesso dal menù: /Dati/Grafico delle variabili

Accesso alternativo: Menù pop-up nella finestra principale, pulsante grafico sulla barra degli strumenti

graph

Argomenti: *variabili-y variabile-x*

Opzione: `--tall` (usa 40 righe)

Grafici ASCII. Le *variabili-y* (che possono essere definite per nome o numero) sono rappresentate rispetto alla *variabile-x* usando simboli ASCII. L'opzione `--tall` produce un grafico di 40 righe per 60 colonne, altrimenti il grafico sarà di 20 righe per 60 colonne (per la visualizzazione a schermo). Si veda anche [gnuplot](#).

hausman

Questo test è disponibile solo dopo aver stimato un modello con il comando [pooled](#) (si veda anche `panel` e `setobs`). Testa il semplice modello “pooled” (con tutte le osservazioni considerate indistintamente) contro le principali alternative: il modello a effetti fissi e quello a effetti casuali.

Il modello a effetti fissi aggiunge una variabile dummy per tutte le unità cross section tranne una, permettendo così all'intercetta della regressione di variare per ogni unità. Viene poi eseguito un test *F* per la significatività congiunta di queste dummy. Il modello a effetti casuali scompone la varianza dei residui in due parti: una specifica all'unità cross section e una specifica all'osservazione particolare (la stima può essere eseguita solo se il numero delle unità cross section nel dataset è maggiore del numero dei parametri da stimare). La statistica LM di Breusch-Pagan testa l'ipotesi nulla (che il modello pooled OLS sia adeguato) contro l'alternativa a effetti casuali.

Può accadere che il modello pooled OLS sia rifiutato nei confronti di entrambe le alternative, a effetti fissi o casuali. A patto che gli errori specifici di unità o di gruppo siano non correlati con le variabili indipendenti, lo stimatore a effetti casuali è più efficiente dello stimatore a effetti fissi; nel caso contrario lo stimatore a effetti casuali non è consistente e deve essergli preferito lo stimatore a effetti fissi. L'ipotesi nulla per il test di Hausman è che l'errore specifico di gruppo non sia correlato con le variabili indipendenti (e quindi che il modello a effetti casuali sia preferibile). Un basso p-value per questo test suggerisce di rifiutare il modello a effetti casuali in favore del modello a effetti fissi.

Accesso dal menù: Finestra del modello, /Test/HAUSMAN - Diagnosi panel

hccm

Argomenti: *variabile-dipendente variabili-indipendenti*

Opzione: --vcv (mostra la matrice di covarianza)

Matrice di covarianza coerente con l'eteroschedasticità (HCCM: Heteroskedasticity-Consistent Covariance Matrix); questo comando esegue una regressione dove i coefficienti sono stimati con la procedura OLS standard, ma gli errori standard delle stime dei coefficienti sono calcolati in modo da essere robusti rispetto all'eteroschedasticità, ossia usando la procedura "jackknife" di MacKinnon-White.

Accesso dal menù: /Modello/HCCM

help

Mostra un elenco dei comandi disponibili, *help comando* descrive il *comando* (ad es. *help smpl*). Al posto di *help* si può usare *man* se si preferisce.

Accesso dal menù: /Aiuto

hilu

Argomenti: *variabile-dipendente variabili-indipendenti*

Opzioni: --vcv (mostra la matrice di covarianza)

--no-corc (non affina i risultati con Cochrane-Orcutt)

Calcola le stime dei parametri per il modello specificato usando la procedura di ricerca di Hildreth-Lu (ottimizzata con la procedura CORC), progettata per correggere l'effetto della correlazione seriale nel termine di errore. Viene mostrato un grafico della somma dei quadrati degli errori per il modello trasformato rispetto ai valori di rho da -0.99 a 0.99.

Accesso dal menù: /Modello/Serie storiche/Hildreth-Lu

hsk

Argomenti: *variabile-dipendente variabili-indipendenti*

Opzione: --vcv (mostra la matrice di covarianza)

Stima una regressione OLS e ne salva i residui. I logaritmi dei quadrati di questi residui diventano la variabile dipendente di una regressione ausiliaria che ha come regressori le variabili indipendenti originali e i loro quadrati. I valori stimati dalla regressione ausiliaria sono quindi usati per costruire una serie di pesi, usati per ri-stimare il modello originale con la procedura dei minimi quadrati ponderati. Viene riportato il risultato finale.

La serie dei pesi è composta da $1/\sqrt{e^{y^*}}$, dove y^* denota i valori stimati dalla regressione ausiliaria.

Accesso dal menù: /Modello/WLS corretti per eteroschedasticità

hurst

Argomento: *nome-variabile*

Calcolate l'esponente di Hurst (una misura di persistenza, o di memoria lunga) per una serie storica con almeno 128 osservazioni.

L'esponente di Hurst è discusso da Mandelbrot. In termini teorici è l'esponente H nella relazione

$$\mathrm{RS}(x) = an^H$$

dove RS è l'“intervallo riscalato” della variabile x in un campione dell'ampiezza n , mentre a è una costante. L'intervallo riscalato è l'intervallo (valore massimo meno valore minimo) del valore cumulato, o somma parziale, di x sul periodo del campione (dopo aver sottratto la media campionaria), diviso per la deviazione standard campionaria.

Come punto di riferimento, se x è un rumore bianco (media zero, persistenza zero) l'intervallo dei suoi valori cumulati (che forma una passeggiata casuale), scalato per la deviazione standard, cresce come la radice quadrata dell'ampiezza campionaria, ossia ha un esponente di Hurst atteso pari a 0.5. Valori dell'esponente sensibilmente maggiori di 0.5 indicano persistenza della serie, mentre valori minori di 0.5 indicano anti-persistenza (autocorrelazione negativa). In teoria l'esponente deve essere compreso tra 0 e 1, ma in campioni finiti è possibile ottenere delle stime per l'esponente maggiori di 1.

In gretl, l'esponente è stimato usando il sotto-campionamento binario: si inizia dall'intero intervallo dei dati, quindi si usano le due metà dell'intervallo, poi i quattro quarti, e così via. Il valore RS è la media presa sui vari campioni. L'esponente è quindi stimato come il coefficiente di pendenza della regressione del logaritmo di RS sul logaritmo dell'ampiezza del campione.

Accesso dal menù: /Variable/Esponente di Hurst

if

Controllo di flusso per l'esecuzione dei comandi. La sintassi è:

```
if  condizione
    comandi1
else
    comandi2
endif
```

La *condizione* deve essere un'espressione Booleana, per la cui sintassi si veda [genr](#). Il blocco *else* è opzionale; i blocchi *if ... endif* possono essere nidificati.

import

Argomento: *nomefile*
 Opzione: --box1 (dati BOX1)

Importa dati da un file in formato valori separati da virgole (CSV: comma-separated values), che può essere facilmente prodotto da un foglio elettronico. Il file deve contenere i nomi delle variabili sulla prima riga e una matrice rettangolare di dati nelle righe rimanenti. Le variabili devono essere disposte “per osservazione” (una colonna per variabile, ogni riga rappresenta un'osservazione). Si veda il [Capitolo 4](#) per i dettagli.

Con l'opzione --box1 viene letto un file in formato BOX1, che può essere ottenuto usando il Servizio di Estrazione Dati del Bureau of the Census degli USA.

Accesso dal menù: /File/Apri dati/importa

include

Argomento: *file*

Da usare in uno script di comandi, principalmente per includere definizioni di funzioni. Esegue i comandi nel *file* e ripassa il controllo allo script principale.

info

Mostra le informazioni aggiuntive contenute nel file di dati attuale.

Accesso dal menù: /Dati/Visualizza descrizione

Accesso alternativo: Finestre di esplorazione dei dati

label

Argomenti: *nome-variabile -d descrizione -n nome-grafici*

Esempio: `label x1 -d "Descrizione di x1" -n "Nome per i grafici"`

Imposta l'etichetta descrittiva per la variabile indicata (usando l'opzione `-d` seguita da una stringa tra virgolette doppie) e/o un "nome per i grafici", usato per indicare la variabile nei grafici (usando l'opzione `-n`, seguita da una stringa tra virgolette doppie).

Se non viene usata alcuna opzione, il comando mostra l'etichetta descrittiva per la variabile specificata, se esiste.

Accesso dal menù: /Variabile/Modifica attributi

Accesso alternativo: Menù pop-up nella finestra principale

kpss

Argomenti: *ordine nome-variabile*

Opzioni: `--trend` (include un trend)

`--verbose` (mostra i risultati della regressione)

Esempi: `kpss 8 y`

`kpss 4 x1 --trend`

Calcola il test KPSS (Kwiatkowski, Phillips, Schmidt e Shin, 1992) per la stazionarietà di una variabile. L'ipotesi nulla è che la variabile in questione sia stazionaria, attorno a un valore fisso o, se è stata usata l'opzione `--trend`, attorno a un trend deterministico lineare.

L'argomento `ordine` determina la dimensione della finestra usata per il livellamento di Bartlett. Se si usa l'opzione `--verbose`, vengono mostrati anche i risultati della regressione ausiliaria, insieme alla varianza stimata della componente random walk della variabile.

Accesso dal menù: /Variabile/Test KPSS

labels

Mostra le etichette informative per le variabili generate con il comando `genr` e quelle aggiunte al dataset attraverso l'interfaccia grafica.

lad

Argomenti: *variabile-dipendente variabili-indipendenti*

Opzione: `--vcv` (mostra la matrice di covarianza)

Calcola una regressione che minimizza la somma delle deviazioni assolute dei valori stimati dai valori effettivi della variabile dipendente. Le stime dei coefficienti sono derivate usando l'algoritmo del semplice di Barrodale-Roberts; viene mostrato un messaggio di avvertimento se la soluzione non è unica.

Gli errori standard sono derivati usando la procedura bootstrap con 500 estrazioni. La matrice di covarianza per le stime dei parametri, mostrata se si usa l'opzione `--vcv`, si basa sulla stessa procedura.

Accesso dal menù: /Modello/LAD - Minime deviazioni assolute

lags

Forme alternative: `lags lista-variabili`

`lags ordine ; lista-variabili`

Esempi: `lags x y`

`lags 12 ; x y`

Crea delle nuove variabili come valori ritardati di ognuna delle variabili nella *lista-variabili*. Il numero dei ritardi può essere indicato dal primo parametro opzionale, altrimenti sarà pari alla periodicità del dataset. Ad esempio, se la periodicità è 4 (trimestrale), il comando `lags x y` crea $x_{-1} = x(t-1)$, $x_{-2} = x(t-2)$, $x_{-3} = x(t-3)$ e $x_{-4} = x(t-4)$.

Accesso dal menù: /Dati/Aggiungi variabili/Ritardi delle variabili selezionate

ldiff

Argomento: `lista-variabili`

Calcola la differenza prima del logaritmo naturale di ogni variabile della *lista-variabili* e la salva in una nuova variabile con il prefisso `ld_`. Così, `ldiff x y` crea le nuove variabili $ld_x = \ln(x_t) - \ln(x_{t-1})$ e $ld_y = \ln(y_t) - \ln(y_{t-1})$.

Accesso dal menù: /Dati/Aggiungi variabili/Differenze logaritmiche

leverage

Opzione: `--save` (salva le variabili)

Deve seguire immediatamente un comando `ols`. Calcola il "leverage" (h , compreso tra 0 e 1) di ogni osservazione nel campione su cui è stato stimato il precedente modello. Mostra il residuo (u) per ogni osservazione assieme al leverage corrispondente e a una misura della sua influenza sulla stima: $uh/(1-h)$. I "punti di leverage" per cui il valore di h supera $2k/n$ (dove k è il numero dei parametri stimati e n è l'ampiezza del campione) sono indicati con un asterisco. Per i dettagli sui concetti di leverage e influenza, si veda Davidson e MacKinnon (1993, capitolo 2).

Vengono mostrati anche i valori DFFITS: questi sono "residui studentizzati" (ossia i residui previsti, divisi per i propri errori standard) moltiplicati per $\sqrt{h/(1-h)}$. Per una discussione dei residui studentizzati e dei valori DFFITS si veda G. S. Maddala, *Introduction to Econometrics*, capitolo 12 e anche Belsley, Kuh e Welsch (1980). In breve, i "residui previsti" sono la differenza tra il valore osservato e il valore stimato della variabile dipendente all'osservazione t , ottenuti da una regressione in cui quell'osservazione è stata omessa (oppure in cui è stata aggiunta una variabile dummy che vale 1 solo per l'osservazione t); il residuo studentizzato si ottiene dividendo il residuo previsto per il proprio errore standard.

Se si usa l'opzione `--save`, il leverage, il valore di influenza e il valore DFFITS vengono aggiunti al dataset in uso.

Accesso dal menù: Finestra del modello, /Test/LEVERAGE - Osservazioni influenti

Imtest

Opzioni: --logs (non-linearità, logaritmi)
 --autocorr (correlazione seriale)
 --squares (non-linearità, quadrati)
 --white (eteroschedasticità (test di White))

Deve seguire immediatamente un comando `ols`. Esegue una combinazione dei test seguenti: test dei moltiplicatori di Lagrange per la non-linearità (logaritmi e quadrati), test di White per l'eteroschedasticità e test LMF per la correlazione seriale di ordine pari alla periodicità (si veda Kiviet, 1986), mostrando anche i coefficienti delle corrispondenti regressioni ausiliarie. Si veda Ramanathan, capitoli 7, 8 e 9 per i dettagli. Nel caso del test di White, vengono usati solo i quadrati delle variabili indipendenti, non i loro prodotti incrociati. Nel caso del test sull'autocorrelazione, se il p-value della statistica LMF è minore di 0.05 (e il modello non è stato stimato con errori standard robusti), vengono calcolati e mostrati errori standard robusti rispetto alla correlazione seriale. Per i dettagli sul calcolo di questi errori standard, si veda Wooldridge (2002, capitolo 12).

Accesso dal menù: Finestra del modello, /Test

logistic

Argomenti: *variabile-dipendente variabili-indipendenti* [*y_{max}=valore*]
 Opzione: --vcv (mostra la matrice di covarianza)
 Esempi: logistic y const x
 logistic y const x y_{max}=50

Regressione logistica: esegue una regressione OLS usando la trasformazione logistica sulla variabile dipendente:

$$\log \left(\frac{y}{y^* - y} \right)$$

La variabile dipendente dev'essere strettamente positiva. Se è una frazione decimale, compresa tra 0 e 1, il valore predefinito per y^* (il massimo asintotico della variabile dipendente) è 1. Se la variabile dipendente è una percentuale, compresa tra 0 e 100, il valore predefinito di y^* è 100. È possibile indicare un valore diverso per il massimo, usando la sintassi opzionale *y_{max}=valore*, che segue la lista dei regressori. Il valore fornito deve essere maggiore di tutti i valori osservati della variabile dipendente.

I valori stimati e i residui della regressione sono trasformati automaticamente usando

$$y = \frac{y^*}{1 + e^{-x}}$$

dove x rappresenta un valore stimato oppure un residuo della regressione OLS, usando la variabile dipendente trasformata. I valori riportati sono dunque confrontabili con la variabile dipendente originale.

Si noti che se la variabile dipendente è binaria, occorre usare il comando `logit` invece di questo comando.

Accesso dal menù: /Modello/Logistico

logit

Argomenti: *variabile-dipendente variabili-indipendenti*
 Opzioni: `--robust` (errori standard robusti)
`--vcv` (mostra la matrice di covarianza)

Regressione binomiale logit. La variabile dipendente deve essere binaria. Le stime di massima verosimiglianza dei coefficienti per le *variabili-indipendenti* sono ottenute con il metodo BRMR (“binary response model regression”) descritto in Davidson e MacKinnon (2004). Visto che il modello è nonlineare, le pendenze dipendono dai valori delle variabili indipendenti: le pendenze riportate sono valutate nelle medie di queste variabili. La statistica chi-quadro testa l'ipotesi nulla che tutti i coefficienti tranne la costante siano pari a zero.

In modalità predefinita, gli errori standard sono calcolati con l'inversa negativa dell'Hessiana. Se si usa l'opzione `--robust`, verranno calcolati gli errori standard QML o quelli di Huber-White. In questo caso, la matrice di covarianza stimata è un “sandwich” dell'inversa dell'Hessiana stimata e del prodotto esterno del gradiente. Per i dettagli, si veda Davidson e MacKinnon 2004, cap. 10.

Per condurre un'analisi delle proporzioni (dove la variabile dipendente è la proporzione dei casi che hanno una certa caratteristica in ogni osservazione, invece che una variabile binaria che indica se la caratteristica è presente o no), non bisogna usare il comando `logit`, ma occorre costruire la variabile logit (ad es. `genr lgt_p = log(p/(1 - p))`) e usare questa come variabile dipendente in una regressione OLS; si veda Ramanathan, capitolo 12.

Accesso dal menù: /Modello/Logit

logs

Argomento: *lista-variabili*

Calcola il logaritmo naturale di ognuna delle variabili della *lista-variabili* e lo salva in una nuova variabile col prefisso `l_`, ossia una “elle” seguita da un trattino basso. `logs x y` crea le nuove variabili `l_x = ln(x)` e `l_y = ln(y)`.

Accesso dal menù: /Dati/Aggiungi variabili/Logaritmi delle variabili selezionate

loop

Argomento: *controllo*
 Opzioni: `--progressive` (abilita modalità speciali di alcuni comandi)
`--verbose` (mostra i dettagli dei comandi `genr`)
 Esempi: `loop 1000`
`loop 1000 --progressive`
`loop while essdiff > .00001`
`loop for i=1991..2000`
`loop for (r=-.99; r<=.99; r+=.01)`

Il parametro *controllo* deve assumere uno dei quattro valori mostrati negli esempi: un numero di volte per cui ripetere i comandi all'interno del loop, oppure “while” seguito da una condizione numerica, oppure “for” seguito da un intervallo di valori per la variabile interna `i` che funge da indice, o ancora “for” seguito da tre espressioni tra parentesi separate da punti e virgola. Nell'ultima forma, l'espressione a sinistra inizializza una variabile, quella centrale imposta la condizione perché l'iterazione continui, e l'ultima espressione imposta un incremento o un decremento da applicare all'inizio della seconda iterazione e di quelle successive (si tratta di una forma ristretta dell'istruzione `for` nel linguaggio di programmazione C).

Questo comando apre una modalità speciale, in cui il programma accetta comandi da eseguire più volte. All'interno di un loop possono essere usati solo alcuni tipi di comandi: `genr`, `ols`, `print`, `printf`, `pvalue`, `sim`, `smpl`, `store` e `summary`, `if`, `else` e `endif`. Si esce dalla modalità loop con l'istruzione `endloop`: solo a questo punto i comandi indicati vengono eseguiti.

Si veda il [Capitolo 10](#) per ulteriori dettagli, esempi e per la spiegazione dell'opzione `--progressive` (che è destinata ad essere usata nelle simulazioni Monte Carlo).

mahal

Argomento: *lista-variabili*
 Opzioni: `--save` (salva le distanze nel dataset)
`--vcv` (mostra la matrice di covarianza)

La distanza di Mahalanobis è la distanza tra due punti in uno spazio k -dimensionale, scalata rispetto alla variazione statistica in ogni dimensione dello spazio. Ad esempio, se p e q sono due osservazioni su un insieme di k variabili con matrice di covarianza C , la distanza di Mahalanobis tra le due osservazioni è data da

$$\sqrt{(p - q)' C^{-1} (p - q)}$$

dove $(p - q)$ è un vettore a k dimensioni. Se la matrice di covarianza è la matrice identità, la distanza di Mahalanobis corrisponde alla distanza Euclidea.

Lo spazio in cui vengono calcolate le distanze è definito dalle variabili selezionate; per ogni osservazione nell'intervallo attuale viene calcolata la distanza tra l'osservazione e il centroide delle variabili selezionate. La distanza è la controparte multidimensionale di uno z-score standard e può essere usata per giudicare se una certa osservazione “appartiene” a un gruppo di altre osservazioni.

Se si usa l'opzione `--vcv`, vengono mostrate la matrice di covarianza e la sua inversa. Se si usa l'opzione `--save`, le distanze vengono salvate nel dataset con il nome `mdist` (o `mdist1`, `mdist2` e così via, se esiste già una variabile con quel nome).

Accesso dal menù: /Dati/Distanze di Mahalanobis

meantest

Argomenti: *var1 var2*
 Opzione: `--unequal-vars` (assume varianze diverse)

Calcola la statistica t per l'ipotesi nulla che le medie della popolazione siano uguali per le variabili *var1* e *var2*, mostrando il suo p-value. L'impostazione predefinita prevede di assumere che le varianze delle due variabili siano uguali, mentre usando l'opzione `--unequal-vars`, si assume che esse siano diverse. Questo è rilevante per la statistica test solo se le due variabili contengono un diverso numero di osservazioni valide (non mancanti).

Accesso dal menù: /Dati/Differenza delle medie

modeltab

Argomenti: *add o show o free*

Manipola la “tabella modelli” di gretl. Si veda il [Capitolo 3](#) per i dettagli. Le opzioni hanno i seguenti effetti: `add` aggiunge l'ultimo modello stimato alla tabella modelli, se possibile; `show` mostra la tabella modelli in una finestra; `free` pulisce la tabella.

Accesso dal menù: Finestra di sessione, Icona Tabella Modelli

mpols

Argomenti: *variabile-dipendente variabili-indipendenti*

Calcola le stime OLS per il modello indicato usando aritmetica in virgola mobile a precisione multipla. Questo comando è disponibile solo se gret1 è compilato con il supporto per la libreria Gnu Multiple Precision (GMP).

Si si vuole stimare un'interpolazione polinomiale usando l'aritmetica a precisione multipla per generare le potenze volute della variabile indipendente, si può usare la sintassi `mpols y 0 x ; 2 3 4`. Si otterrà una regressione di y su x , x quadro, x al cubo e x alla quarta potenza, ossia, i numeri (interi positivi) a destra del punto e virgola specificano le potenze di x da usare. Se si indica più di una variabile indipendente, l'ultima variabile prima del punto e virgola è quella di cui verranno calcolate le potenze.

Accesso dal menù: /Modello/MPOLS - Minimi quadrati in alta precisione

multiply

Argomenti: *x suffisso lista-variabili*

Esempi: `multiply invpop pc 3 4 5 6`
`multiply 1000 big x1 x2 x3`

Le variabili nella *lista-variabili* (indicate per nome o numero) sono moltiplicate per x , che può essere un valore numerico o il nome di una variabile già definita. I prodotti verranno chiamati con il *suffisso* specificato (massimo 3 caratteri), troncando i nomi delle variabili originali se ce n'è bisogno per motivi di spazio. Ad esempio, si supponga di voler creare la versione procapite di una serie di variabili, usando la variabile `pop` (popolazione). Lo si può fare con i comandi seguenti:

```
genr invpop = 1/pop
multiply invpop pc income
```

che creeranno `incomepc` come prodotto di `income` e `invpop`, e `expendpc` come `expend` per `invpop`.

nls

Argomenti: *funzione derivate*

Opzione: `--vcv` (mostra la matrice di covarianza)

Esegue una stima con minimi quadrati non-lineari (NLS: Nonlinear Least Squares) usando una versione modificata dell'algoritmo di Levenberg-Marquandt. Occorre fornire una specificazione di funzione e dichiarare i parametri della funzione (usando il comando `genr`) prima della stima. Opzionalmente, è anche possibile specificare le espressioni per le derivate della funzione rispetto a ognuno dei parametri; in caso contrario, viene calcolata un'approssimazione numerica del Jacobiano.

È più semplice mostrare il funzionamento con un esempio. Quello che segue è uno script completo per stimare la funzione di consumo non-lineare presentata in *Econometric Analysis* di William Greene (capitolo 11 della quarta edizione, o capitolo 9 della quinta). I numeri alla sinistra delle righe sono dei punti di riferimento e non fanno parte dei comandi. Si noti che l'opzione `--vcv` per mostrare la matrice di covarianza delle stime dei parametri, si applica al comando finale `end nls`.

```
1 open greene11_3.gdt
2 ols C 0 Y
3 genr alfa = coeff(0)
4 genr beta = coeff(Y)
5 genr gamma = 1.0
6 nls C = alfa + beta * Y^gamma
```

```

7 deriv alfa = 1
8 deriv beta = Y^gamma
9 deriv gamma = beta * Y^gamma * log(Y)
10 end nls --vcv

```

Spesso è comodo inizializzare i parametri con riferimento a un modello lineare collegato, come è mostrato nelle righe da 2 a 5. I parametri alfa, beta e gamma possono essere impostati a qualunque valore iniziale (non necessariamente sulla base di un modello stimato con OLS), ma la convergenza della procedura NLS non è garantita per qualunque punto di partenza.

I veri comandi NLS occupano le righe da 6 a 10. Sulla riga 6 viene dato il comando `nls`: viene specificata una variabile dipendente, seguita dal segno uguale, seguito da una specificazione di funzione. La sintassi per l'espressione a destra è la stessa usata per il comando `genr`. Le tre righe successive specificano le derivate della funzione di regressione rispetto a ognuno dei parametri. Ogni riga inizia con il comando `deriv`, indica il nome di un parametro, il segno di uguale e un'espressione che indica come calcolare la derivata (anche qui la sintassi è la stessa di `genr`). Queste righe `deriv` sono opzionali, ma si raccomanda di inserirle se possibile. La riga 10, `end nls`, completa il comando ed esegue la stima.

Per ulteriori dettagli sulla stima NLS si veda [Capitolo 9](#).

Accesso dal menù: /Modello/Minimi quadrati non lineari

noecho

Comando obsoleto. Si veda [set](#).

nulldata

Argomento: *lunghezza_serie*

Esempio: `nulldata 500`

Crea un dataset “vuoto”, che contiene solo una costante e una variabile indice, con periodicità 1 e il numero indicato di osservazioni. Ad esempio, è possibile creare un dataset a scopo di simulazione usando alcuni comandi `genr` (come `genr uniform()` e `genr normal()`) per generare dati di prova. Questo comando può essere usato insieme a `loop`. Si veda anche l'opzione “seed” del comando [set](#).

Accesso dal menù: /File/Crea dataset

ols

Argomenti: *variabile-dipendente variabili-indipendenti*

Opzioni: `--vcv` (mostra la matrice di covarianza)
`--robust` (errori standard robusti)
`--quiet` (non mostra i risultati)
`--no-df-corr` (sopprime la correzione per i gradi di libertà)
`--print-final` (si veda sotto)

Esempi: `ols 1 0 2 4 6 7`
`ols y 0 x1 x2 x3 --vcv`
`ols y 0 x1 x2 x3 --quiet`

Calcola le stime minimi quadrati ordinari (OLS: Ordinary Least Squares) usando la *variabile-dipendente* e la lista di *variabili-indipendenti*, che possono essere specificate per nome o numero. Il termine costante può essere indicato usando il numero 0.

Oltre alle stime dei coefficienti e agli errori standard, il programma mostra i p-value per le statistiche t (a due code) e F . Un p-value inferiore a 0.01 indica significatività al livello dell'1 per cento ed è denotato con ***. ** indica invece la significatività tra l'1 e il 5 per cento, mentre * indica un livello di significatività tra il 5 e il 10 per cento. Vengono mostrate anche le statistiche di selezione del modello (il criterio di informazione di Akaike, AIC, e il criterio di informazione bayesiana di Schwarz, BIC). La formula usata per AIC è descritta in Akaike (1974), ossia meno due volte la log-verosimiglianza massimizzata più il doppio del numero di parametri stimati.

Usando l'opzione `--no-df-corr` la correzione per i gradi di libertà non viene applicata nel calcolo della varianza stimata dell'errore (e quindi anche dell'errore standard delle stime dei parametri).

L'opzione `--print-final` è utilizzabile solo nel contesto di un [loop](#). L'effetto è quello di eseguire la regressione in modo silenzioso per tutte le iterazioni del loop tranne l'ultima. Si veda [la Sezione Esempi di loop nel Capitolo 10](#) per i dettagli.

È possibile salvare alcune variabili interne generate durante la stima, usando il comando [genr](#) subito dopo questo comando.

La formula usata per generare gli errori standard robusti (quando viene usata l'opzione `--robust`) può essere modificata con il comando [set](#).

Accesso dal menù: /Modello/OLS - Minimi quadrati ordinari

Accesso alternativo: Pulsante Beta-hat sulla barra degli strumenti

omit

Argomento: *lista-variabili*

Opzioni: `--vcv` (mostra la matrice di covarianza)
`--quiet` (non mostra le stime per il modello ridotto)
`--silent` (non mostra nulla)

Esempi: `omit 5 7 9`
`omit seasonals --quiet`

Questo comando deve seguire un comando di stima. Omette le variabili indicate dal modello precedente e stima il nuovo modello. Viene eseguito un test per la significatività congiunta delle variabili omesse; la statistica test è la F nel caso della stima OLS, altrimenti è un chi-quadro asintotico di Wald. Un p-value inferiore a 0.05 indica che i coefficienti sono congiuntamente significativi al livello del 5 per cento.

Se viene usata l'opzione `--quiet`, i risultati mostrati comprendono solo il test per la significatività congiunta delle variabili omesse, altrimenti vengono mostrate anche le stime del modello ridotto. In quest'ultimo caso, l'opzione `--vcv` mostra anche la matrice di covarianza dei coefficienti. Usando l'opzione `--silent`, non viene mostrato nulla; tuttavia i risultati del test possono essere recuperati usando le variabili speciali `$test` e `$pvalue`.

Accesso dal menù: Finestra del modello, /Test/OMIT - Ometti variabili

omitfrom

Argomenti: *id-modello lista-variabili*

Opzione: `--quiet` (non mostra le stime per il modello ridotto)

Esempio: `omitfrom 2 5 7 9`

Funziona come [omit](#), tranne per il fatto che è possibile indicare un modello precedentemente stimato, attraverso il suo numero identificativo (che è mostrato all'inizio dei risultati del modello). L'esempio precedente omette le variabili numero 5, 7 e 9 dal modello 2.

Accesso dal menù: Finestra del modello, /Test/OMIT - Ometti variabili

open

Argomento: *file-dati*

Apri un file di dati. Se è già stato aperto un file di dati, esso viene sostituito da quello selezionato. Il programma cerca di determinare il formato del file di dati (gretl, testo semplice, CSV o BOX1).

Questo comando può essere usato anche per aprire un database (gretl o RATS 4.0) per la lettura. In questo caso, dev'essere seguito dal comando [data](#) per estrarre una particolare serie dal database.

Accesso dal menù: /File/Apri dati

Accesso alternativo: Trascinare un file di dati in gretl (MS Windows o Gnome)

outfile

Argomenti: *file-output opzione*

Opzioni:
--append (aggiunge al file)
--close (chiude il file)
--write (sovrascrive il file)

Esempi:
outfile --write regress.txt
outfile --close

Scrivi i risultati sul *file-output*, fino a nuovo ordine. Usando l'opzione --append, i risultati vengono aggiunti a un file esistente, mentre --write apre un nuovo file (o ne sovrascrive uno esistente). Può essere aperto solo un file alla volta.

L'opzione --close può essere usata per chiudere un file di output aperto in precedenza, tornando a scrivere i risultati sul canale predefinito.

Nel primo degli esempi precedenti viene aperto il file regress.txt, mentre nel secondo viene chiuso. Se prima del comando --close fosse eseguito un comando `ols`, i risultati della regressione verrebbero scritti su regress.txt invece che sullo schermo.

panel

Opzioni:
--cross-section (pila di dati cross section)
--time-series (pila di dati serie storiche)

Richiede che il dataset in uso sia interpretato come un panel (unione di dati cross section e serie storiche). L'impostazione predefinita corrisponde all'opzione --time-series e prevede di interpretare il dataset come una pila di serie storiche (costituita da blocchi successivi di dati che contengono serie storiche per ogni unità cross section). Usando l'opzione --cross-section il dataset viene letto come una pila di dati cross-section (blocchi successivi di dati contengono dati cross section per ognuno dei periodi temporali). Si veda anche [setobs](#).

pca

Argomento: *lista-variabili*

Opzioni:
--save-all (salva tutte le componenti)
--save (salva le componenti principali)

Analisi delle componenti principali. Mostra gli autovalori della matrice di correlazione per le variabili nella *lista-variabili*, insieme alla proporzione della varianza comune spiegata da ogni componente. Mostra anche i corrispondenti autovettori (o “pesi della componente”).

Usando l'opzione `--save`, le componenti con autovalori maggiori di 1.0 vengono salvati nel dataset come variabili, con i nomi PC1, PC2 e così via. Queste variabili artificiali sono definite come la somma del peso della componente moltiplicato per X_i standardizzato, dove X_i denota la i -esima variabile nella *lista-variabili*.

Usando l'opzione `--save-all`, vengono salvate tutte le componenti, come descritto sopra.

Accesso dal menù: Pop-up nella finestra principale (selezione multipla)

pergm

Argomento: *nome-variabile*

Opzione: `--bartlett` (usa la finestra di Bartlett)

Calcola e mostra (graficamente se non si è in modalità batch) lo spettro della variabile specificata. Senza l'opzione `--bartlett`, viene mostrato il periodogramma nel campione, usando l'opzione, lo spettro viene stimato usando una finestra di Bartlett per i ritardi di lunghezza $2\sqrt{T}$ (dove T è l'ampiezza del campione); si veda il capitolo 18 di *Econometric Analysis* di Greene. Se viene mostrato il periodogramma del campione, viene mostrato anche il test t per l'integrazione frazionale della serie (“memoria lunga”): l'ipotesi nulla è che l'ordine di integrazione sia zero.

Accesso dal menù: /Variabile/Spettro

Accesso alternativo: Menù pop-up nella finestra principale (selezione singola)

poisson

Argomenti: *variabile-dipendente variabili-indipendenti* [; *offset*]

Opzioni: `--vcv` (mostra la matrice di covarianza)

`--verbose` (mostra i dettagli delle iterazioni)

Esempi: `poisson y 0 x1 x2`
`poisson y 0 x1 x2 ; S`

Stima una regressione di Poisson, in cui la variabile dipendente rappresenta le occorrenze di un qualche tipo di evento e può assumere solo valori interi non negativi.

Se una variabile casuale discreta Y segue la distribuzione di Poisson,

$$\Pr(Y = y) = \frac{e^{-v} v^y}{y!}$$

per $y = 0, 1, 2, \dots$. La media e la varianza della distribuzione sono entrambe uguali a v . Nel modello di regressione di Poisson, il parametro v è rappresentato da una funzione di una o più variabili indipendenti. La versione più comune del modello (e l'unica supportata da gretl) ha

$$v = \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots)$$

, ossia il logaritmo di v è una funzione lineare delle variabili indipendenti.

Opzionalmente è possibile aggiungere una variabile “offset” alla specificazione, ossia una variabile di scala, il cui logaritmo viene aggiunto alla funzione di regressione lineare (con un coefficiente implicito di 1.0). Ciò ha senso se si ipotizza che il numero di occorrenze dell'evento in questione sia proporzionale a qualche fattore noto, a parità di altre condizioni. Ad esempio, il numero di incidenti stradali può essere ipotizzato proporzionale al

volume del traffico, che potrebbe essere specificato come una variabile di “offset” in un modello di Poisson per il tasso di incidenti. La variabile di offset dev’essere strettamente positiva.

Accesso dal menù: /Modello/Poisson

plot

Argomento: *lista-variabili*

Opzione: `--one-scale` (forza una scala sola)

Disegna i valori delle variabili indicate per l'intervallo di osservazioni attuale, usando simboli ASCII. Ogni riga rappresenta un'osservazione e i valori sono disegnati orizzontalmente. Il comportamento predefinito è di scalare la variabili in modo appropriato. Si veda anche [gnuplot](#).

pooled

Argomenti: *variabile-dipendente variabili-indipendenti*

Opzioni: `--unit-weights` (stimatore feasible GLS)

`--iterate` (itera fino alla soluzione di massima verosimiglianza)

`--vcv` (mostra la matrice di covarianza)

In modalità predefinita, stima un modello con OLS (si veda [ols](#) per i dettagli sulla sintassi) e lo marca come modello pooled o panel, rendendo disponibile il comando [hausman](#).

Con l'opzione `--unit-weights`, viene usato lo stimatore feasible GLS, con i pesi costruiti a partire dalle varianze specifiche degli errori per ogni unità cross section. Questa modalità offre un guadagno di efficienza rispetto alla stima OLS nel caso in cui la varianza differisca tra le diverse unità.

Se viene usata anche l'opzione `--iterate`, lo stimatore GLS viene iterato: se la procedura converge, verranno prodotte stime di massima verosimiglianza.

Accesso dal menù: /Modello/POOLED - Pooled OLS (panel)

print

Argomenti: *lista-variabili o stringa-letterale*

Opzioni: `--byobs` (per osservazione)

`--ten` (usa 10 cifre significative)

`--no-dates` (usa i numeri delle osservazioni)

Esempi: `print x1 x2 --byobs`

`print "Questa è una stringa"`

Se viene indicata una *lista-variabili*, stampa i valori delle variabili specificate, altrimenti stampa i valori di tutte le variabili nel dataset in uso. Usando l'opzione `--byobs` i dati vengono stampati per osservazione, altrimenti sono stampati per variabile. Usando l'opzione `--ten` i dati vengono stampati per variabile con 10 cifre significative.

Se si usa l'opzione `--byobs` e i dati sono mostrati per osservazione, il comportamento predefinito è quello di mostrare la data (per serie storiche) o il marcatore (se esiste) all'inizio di ogni riga. L'opzione `--no-dates` sopprime la visualizzazione delle date o dei marcatori: viene mostrato solo un semplice numero di osservazione.

Se l'argomento di `print` è una stringa letterale (che deve iniziare con le virgolette doppie), la stringa viene stampata così come è stata indicata. Si veda anche [printf](#).

Accesso dal menù: /Dati/Mostra valori

printf

Argomenti: *formato argomenti*

Stampa valori scalari nel formato indicato da una stringa di formato (che supporta un piccolo sottoinsieme del comando `printf()` del linguaggio di programmazione C). I formati riconosciuti sono `%g` e `%f`, con i vari modificatori disponibili in C. Esempi: la stringa `%.10g` stampa un valore con 10 cifre significative; `%12.6f` stampa un valore con 6 cifre decimali e una larghezza di 12 caratteri.

La stringa di formato deve essere racchiusa tra virgolette doppie, i valori da stampare devono seguire la stringa di formato, separati da virgole. I valori possono avere tre forme: a) i nomi di variabili nel dataset; b) espressioni valide per il comando `genr`; c) le funzioni speciali `varname()` o `date()`. L'esempio seguente stampa i valori di due variabili e quello di un'espressione calcolata:

```
ols 1 0 2 3
genr b = coeff(2)
genr se_b = stderr(2)
printf "b = %.8g, standard error %.8g, t = %.4f\n", b, se_b, b/se_b
```

Le prossime righe mostrano l'uso delle funzioni `varname` e `date`, che rispettivamente mostrano il nome di una variabile dato il suo numero identificativo, e una stringa data, dato un numero di osservazione.

```
printf "Il nome della variabile %d è %s\n", i, varname(i)
printf "La data dell'osservazione %d è %s\n", j, date(j)
```

La lunghezza massima di una stringa di formato è di 127 caratteri. Vengono riconosciute le sequenze di escape `\n` (newline), `\t` (tab), `\v` (tab verticale) e `\\` (barra inversa). Per stampare un segno di percentuale, si usi `%%`.

probit

Argomenti: *variabile-dipendente variabili-indipendenti*

Opzioni: `--robust` (errori standard robusti)
`--vcv` (mostra la matrice di covarianza)

Stima un modello probit, per variabili dipendenti binarie. Le stime di massima verosimiglianza dei coefficienti delle *variabili-indipendenti* sono ottenute con il metodo “binary response model regression” (BRMR) descritto in Davidson e MacKinnon (2004). Visto che il modello è nonlineare, le pendenze dipendono dai valori delle variabili indipendenti: le pendenze riportate sono valutate nelle medie di queste variabili. La statistica chi-quadro testa l'ipotesi nulla che tutti i coefficienti tranne la costante siano pari a zero.

In modalità predefinita, gli errori standard sono calcolati con l'inversa negativa dell'Hessiana. Se si usa l'opzione `--robust`, verranno calcolati gli errori standard QML o quelli di Huber-White. In questo caso, la matrice di covarianza stimata è un “sandwich” dell'inversa dell'Hessiana stimata e del prodotto esterno del gradiente. Per i dettagli, si veda Davidson e MacKinnon 2004, cap. 10.

Il probit per l'analisi delle proporzioni non è ancora stato implementato in `gretl`.

Accesso dal menù: /Modello/Probit

pvalue

Argomenti: *distribuzione [parametri] valore-x*

Esempi: `pvalue z zscore`
`pvalue t 25 3.0`

```
pvalue X 3 5.6
pvalue F 4 58 fval
pvalue G xbar varx x
```

Calcola l'area alla destra del *valore-x* nella distribuzione indicata (z per la Gaussiana, t per la *t* di Student, X per la chi-quadro, F per la *F* e G per la gamma). Per le distribuzioni *t* e chi-quadro vanno indicati i gradi di libertà; per la *F* sono richiesti i gradi di libertà al numeratore e al denominatore; per la gamma sono richieste la media e la varianza.

Accesso dal menù: /Utilità/Calcola p-value

pwe

Argomenti: *variabile-dipendente variabili-indipendenti*
 Opzione: --vcv (mostra la matrice di covarianza)
 Esempio: pwe 1 0 2 4 6 7

Calcola le stime dei parametri usando la procedura Prais-Winsten, un'implementazione GLS sviluppata per gestire l'autocorrelazione del primo ordine nel termine di errore. La procedura viene iterata, così come in [corc](#); la differenza è che mentre Cochrane-Orcutt traslascia la prima osservazione, Prais-Winsten ne fa uso. Per i dettagli, si veda per esempio il capitolo 13 di *Econometric Analysis* (2000) di Greene.

Accesso dal menù: /Modello/Serie storiche/PWE - Prais-Winsten

quit

Esce dal programma, dando la possibilità di salvare i risultati della sessione.

Accesso dal menù: /File/Esci

rename

Argomenti: *numero-var nuovo-nome*

Modifica il nome di una variabile con numero identificativo *numero-var* in *nuovo-nome*. Il *numero-var* deve essere compreso tra 1 e il numero di variabili nel dataset. Il nuovo nome deve essere lungo al massimo 8 caratteri, deve iniziare con una lettera e deve essere composto di sole lettere, numeri e il carattere trattino basso.

Accesso dal menù: /Variabile/Modifica attributi

Accesso alternativo: Menù pop-up nella finestra principale (selezione singola)

reset

Va eseguito dopo la stima di un modello via OLS. Esegue il test RESET di Ramsey per la specificazione del modello (non-linearità), aggiungendo alla regressione il quadrato e il cubo dei valori stimati e calcolando la statistica *F* per l'ipotesi nulla che i coefficienti dei due termini aggiunti siano pari a zero.

Accesso dal menù: Finestra del modello, /Test/RESET - Ramsey

restrict

Impone un insieme di vincoli lineari sull'ultimo modello stimato o su un sistema di equazioni definito in precedenza. La sintassi del comando è leggermente diversa in ognuno dei due casi.

In entrambi i casi, l'insieme di vincoli deve essere racchiuso tra i comandi "restrict" e "end restrict". Nel caso della singola equazione, i vincoli sono applicati implicitamente all'ultimo modello e vengono valutati appena viene terminato il comando "restrict". Nel caso del sistema, il comando iniziale "restrict" deve essere seguito immediatamente dal nome di un sistema di equazioni definito in precedenza (si veda [system](#)). I vincoli vengono valutati nella successiva stima del sistema effettuata con il comando [estimate](#).

Ogni vincolo nell'insieme va indicato sotto forma di equazione con una combinazione lineare dei parametri al primo membro e un valore numerico al secondo. Nel caso della singola equazione, i parametri sono indicati con la sintassi b_N , dove N rappresenta la posizione nella lista dei regressori, a partire da zero. Ad esempio, b_1 indica il secondo parametro della regressione. Nel caso del sistema, i parametri vengono indicati con la sintassi b seguita da due numeri tra parentesi quadre. Il primo numero rappresenta la posizione dell'equazione all'interno del sistema a partire da 1, mentre il secondo indica la posizione nella lista dei regressori, a partire da zero. Ad esempio $b[2,0]$ indica il primo parametro della seconda equazione, mentre $b[3,1]$ il secondo parametro della terza equazione.

I termini b nell'equazione che rappresenta un vincolo possono essere prefissati da un moltiplicatore numerico, usando il segno $*$ per indicare la moltiplicazione, ad esempio $3.5*b_4$.

Ecco un esempio di un insieme di vincoli per un modello stimato in precedenza:

```
restrict
  b1 = 0
  b2 - b3 = 0
  b4 + 2*b5 = 1
end restrict
```

Ed ecco un esempio di un insieme di vincoli da applicare a un sistema (se il nome del sistema non contiene spazi, è possibile tralasciare le virgolette).

```
restrict "Sistema 1"
  b[1,1] = 0
  b[1,2] - b[2,2] = 0
  b[3,4] + 2*b[3,5] = 1
end restrict
```

Nel caso dell'equazione singola, i vincoli sono valutati attraverso un test F di Wald, usando la matrice di covarianza dei coefficienti del modello in questione. Nel caso del sistema, vengono presentati i risultati completi della stima del sistema soggetto ai vincoli; la statistica test dipende dallo stimatore scelto (un test del rapporto di verosimiglianza nel caso di un sistema stimato con un metodo di massima verosimiglianza, o altrimenti un test F asintotico).

Accesso dal menù: Modello, /Test/Vincoli lineari

rhodiff

Argomenti: *lista-rho ; lista-variabili*

Esempi: `rhodiff .65 ; 2 3 4`

`rhodiff r1 r2 ; x1 x2 x3`

Crea delle versioni rho-differenziate delle variabili indicate (con numero o con nome) nella *lista-variabili* e le aggiunge al dataset, usando il suffisso $\#$ per le nuove variabili. Ad esempio, data la variabile v_1 nella *lista-variabili* e i valori r_1 e r_2 nella *lista-rho*, viene creata $v_{1\#} = v_1(t) - r_1*v_1(t-1) - r_2*v_1(t-2)$. I valori nella *lista-rho* possono essere specificati sotto forma di valori numerici o di nomi di variabili definite in precedenza.

rmplot

Argomento: *nome-variabile*

Grafici Range-mean: questo comando crea un semplice grafico che aiuta a capire se una serie storica $y(t)$ ha varianza costante o no. L'intero campione $t=1,...,T$ viene diviso in piccoli sotto-campioni di dimensione arbitraria k . Il primo sotto-campione è formato da $y(1), \dots, y(k)$, il secondo da $y(k+1), \dots, y(2k)$, e così via. Per ogni sotto-campione, vengono calcolati la media e il campo di variazione (range: il valore massimo meno quello minimo) e viene costruito un grafico con le medie sull'asse orizzontale e i campi di variazione su quello verticale, in modo che ogni sotto-campione sia rappresentato da un punto sul piano. Se la varianza della serie è costante, ci si aspetta che il campo di variazione del sotto-campione sia indipendente dalla media del sotto-campione; se i punti si dispongono su una linea crescente, la varianza della serie cresce al crescere della media, viceversa se i punti si dispongono su una linea decrescente.

Oltre al grafico, gretl mostra anche le medie e i campi di variazione per ogni sotto-campione, insieme al coefficiente di pendenza della regressione OLS del campo di variazione sulla media e il p-value per l'ipotesi nulla che la pendenza sia zero. Se il coefficiente di pendenza è significativo al livello del 10 per cento, viene mostrata sul grafico la linea stimata della regressione del campo di variazione sulla media.

Accesso dal menù: /Variabile/Grafico range-mean

run

Argomento: *file-input*

Esegue i comandi nel *file-input* e restituisce il controllo al prompt interattivo.

Accesso dal menù: Icona Esegui nella finestra comandi

runs

Argomento: *nome-variabile*

Esegue il test non parametrico "delle successioni" per la casualità della variabile specificata. Ad esempio, per testare la casualità delle deviazioni dalla mediana per una variabile chiamata *x1*, con una mediana diversa da zero, eseguire i comandi seguenti:

```
genr signx1 = x1 - median(x1)
runs signx1
```

Accesso dal menù: /Variabile/Test delle successioni

scatters

Argomenti: *variabile-y ; lista-variabili-x o lista-variabili-y ;*
variabile-x

Esempi: `scatters 1 ; 2 3 4 5`
`scatters 1 2 3 4 5 6 ; 7`

Produce grafici a dispersione della *variabile-y* rispetto ad ognuna delle variabili nella *lista-variabili-x*, oppure di tutte le variabili nella *lista-variabili-y* rispetto alla *variabile-x*. Il primo esempio visto sopra assegna la variabile 1 all'asse *y* e produce quattro grafici, il primo con la variabile 2 sull'asse *x*, il secondo con la variabile 3 sull'asse *x*, e così via. Il secondo esempio rappresenta ognuna delle variabili da 1 a 6 rispetto alla variabile 7 sull'asse *x*. Questi gruppi di grafici sono utili nell'analisi esplorativa dei dati. È possibile creare fino a sei grafici alla volta, eventuali variabili in sovrappiù saranno ignorate.

Accesso dal menù: /Dati/Grafici multipli a dispersione

seed

Comando obsoleto. Si veda [set](#).

set

Argomenti: *variabile valore*
 Esempi: `set qr on`
`set csv_delim tab`
`set horizon 10`

Imposta i valori di vari parametri del programma. Il valore impostato rimane in vigore per la durata della sessione di gretl, a meno di non essere modificato da un'ulteriore esecuzione del comando `set`. I parametri che possono essere impostati in questo modo sono elencati di seguito. Si noti che le impostazioni di `hac_lag` e `hc_version` sono usate quando viene data l'opzione `--robust` al comando `ols`.

<code>echo</code>	Valori: <code>off</code> o <code>on</code> (valore predefinito). Sopprime o ripristina l'indicazione dei comandi eseguiti nell'output dei risultati.
<code>qr</code>	Valori: <code>on</code> o <code>off</code> (valore predefinito). Usa la decomposizione QR invece di quella di Cholesky nel calcolo delle stime OLS.
<code>seed</code>	valore: un intero senza segno. Imposta il seme per il generatore di numeri pseudo-casuali. Di solito il seme viene impostato a partire dall'ora di sistema, ma se si intende generare sequenze ripetibili di numeri casuali occorre impostare il seme manualmente.
<code>hac_lag</code>	Valori: <code>nw1</code> (valore predefinito) o <code>nw2</code> , o un intero. Imposta il massimo valore di ritardo, p , usato nel calcolo degli errori standard HAC (Heteroskedasticity and Autocorrelation Consistent) con l'approccio Newey-West, per le serie storiche. <code>nw1</code> e <code>nw2</code> rappresentano due varianti di calcolo automatico basate sulla dimensione del campione, T : per <code>nw1</code> , $p = 0.75 \times T^{1/3}$, e per <code>nw2</code> , $p = 4 \times (T/100)^{2/9}$.
<code>hc_version</code>	Valori: 0 (valore predefinito), 1, 2 o 3. Imposta la variante da usare nel calcolo degli errori standard HAC (Heteroskedasticity and Autocorrelation Consistent) con dati di tipo cross section. Le opzioni corrispondono alle HC0, HC1, HC2 e HC3 discusse da Davidson e MacKinnon nel capitolo 5 di <i>Econometric Theory and Methods</i> . HC0 produce quelli che di solito vengono chiamati "errori standard di White".
<code>force_hc</code>	Valori: <code>off</code> (predefinito) o <code>on</code> . Lo stimatore HAC viene usato in modo predefinito con dati serie storiche e quando si usa l'opzione <code>--robust</code> di <code>ols</code> . Impostando invece <code>force_hc</code> a "on", si forza l'uso della matrice di covarianza coerente con l'eteroschedasticità (che non tiene conto dell'autocorrelazione).
<code>garch_vcv</code>	Valori: <code>unset</code> , <code>hessian</code> , <code>im</code> (matrice di informazione), <code>op</code> (matrice dei prodotti esterni), <code>qml</code> (stimatore QML), <code>bw</code> (Bollerslev-Wooldridge). Specifica la variante da usare per stimare la matrice di covarianza dei coefficienti nei modelli GARCH. Se si usa <code>unset</code> (valore predefinito), viene usata l'Hessiana, a meno di usare l'opzione "robust" col comando <code>garch</code> , nel qual caso viene usato QML.
<code>hp_lambda</code>	Valori: <code>auto</code> (valore predefinito), o un valore numerico. Imposta il parametro di livellamento per il filtro di Hodrick-Prescott (si veda la funzione <code>hpfilt</code> sotto il comando <code>genr</code>). Il valore predefinito è 100 volte il quadrato della periodicità, ossia 100 per i dati annuali, 1600 per i dati trimestrali e così via.

<code>bkbp_limits</code>	Valori: due interi, il secondo maggiore del primo (i valori predefiniti sono 8 e 32). Imposta i limiti di frequenza per il filtro passa-banda di Baxter-King (si veda la funzione <code>bkfilt</code> nel comando <code>genr</code>).
<code>bkbp_k</code>	Valori: un intero (il valore predefinito è 8). Imposta l'ordine di approssimazione per il filtro passa-banda di Baxter-King.
<code>horizon</code>	Valori: un intero (il valore predefinito dipende dalla frequenza dei dati). Imposta l'orizzonte per le funzioni impulso-risposta e per la decomposizione della varianza nel contesto delle autoregressioni vettoriali.
<code>csv_delim</code>	Valori: comma (virgola, valore predefinito), space (spazio), o tab. Imposta il delimitatore di colonna usato nel salvataggio di dati su file in formato CSV.

setobs

Argomenti:	<i>periodicità oss-iniziale</i>
Opzioni:	--cross-section (interpreta come cross section) --time-series (interpreta come serie storiche) --stacked-cross-section (interpreta come panel) --stacked-time-series (interpreta come panel)
Esempi:	<pre>setobs 4 1990:1 --time-series setobs 12 1978:03 setobs 1 1 --cross-section setobs 20 1:1 --stacked-time-series</pre>

Forza il programma a interpretare il dataset in uso secondo la struttura specificata.

La *periodicità*, che deve essere un valore intero, nel caso delle serie storiche rappresenta la frequenza delle osservazioni (1 = annuale; 4 = trimestrale; 12 = mensile; 52 = settimanale; 5, 6, o 7 = giornaliera; 24 = oraria). Nel caso di dati panel, la periodicità è il numero di righe per ogni blocco di dati, ossia il numero di unità cross section se i dati sono organizzati come pila di dati cross section, o il numero di periodi se i dati sono organizzati come pila di serie storiche. Nel caso di semplici dati cross section, la periodicità dev'essere impostata a 1.

L'osservazione iniziale rappresenta la data iniziale nel caso delle serie storiche. Gli anni possono essere indicati con due o quattro cifre, mentre i sotto-periodi (ad esempio i trimestri o i mesi) devono essere separati dagli anni con un carattere "due punti". Nel caso di dati panel, l'osservazione iniziale va indicata come 1:1, mentre nel caso di dati cross section come 1. L'osservazione iniziale per i dati giornalieri o settimanali va indicata nella forma AA/MM/GG o AAAA/MM/GG (oppure semplicemente 1 per i dati non datati).

Se non viene usata nessuna opzione per indicare esplicitamente la struttura dei dati, il programma cercherà di riconoscerla automaticamente a partire dalle informazioni indicate.

Accesso dal menù: Campione/Struttura dataset

setmiss

Argomenti:	<i>valore [lista-variabili]</i>
Esempi:	<pre>setmiss -1 setmiss 100 x2</pre>

Imposta il programma in modo da interpretare un dato valore numerico (il primo parametro indicato al comando) come codice per i “valori mancanti” nei dati importati. Se questo valore è l’unico parametro fornito, come nel primo degli esempi precedenti, l’interpretazione verrà applicata a tutte le serie del dataset. Se *valore* è seguito da una lista di variabili, indicate per nome o numero, l’interpretazione è limitata solo alle variabili specificate. Così, nel secondo esempio, il valore 100 è interpretato come codice per “mancante”, ma solo per la variabile *x2*.

Accesso dal menù: /Campione/Imposta codice valori mancanti

shell

Argomento: *comando-shell*

Esempi: `! ls -al`
`! notepad`

Un `!` all’inizio di una riga di comando è interpretato come passaggio all’interprete di comandi (shell) usato dall’utente nel sistema operativo. In questo modo è possibile eseguire comandi shell arbitrari dall’interno di *gretl*.

sim

Argomenti: `[ oss-iniziale oss-finale ] variabile a0 a1 a2 ...`

Esempi: `sim 1979.2 1983.1 y 0 0.9`
`sim 15 25 y 10 0.8 x`

Simula valori per la *variabile* nell’intervallo del campione in uso, o per l’intervallo che va da *oss-iniziale* a *oss-finale*, se vengono dati questi parametri opzionali. La variabile *y* deve essere stata definita in precedenza con valori iniziali appropriati. La formula usata è

$$y_t = a_{0t} + a_{1t}y_{t-1} + a_{2t}y_{t-2} + \dots$$

I termini $a_i(t)$ possono essere costanti numeriche o nomi di variabili definite in precedenza e possono essere preceduti dal segno meno.

L’uso di questo comando è sconsigliato. Si consiglia di usare [genr](#).

smp1

Forme alternative: `smp1 oss-iniziale oss-finale`

`smp1 +i -j`
`smp1 variabile-dummy --dummy`
`smp1 condizione --restrict`
`smp1 --no-missing [ lista-variabili ]`
`smp1 n --random`
`smp1 full`

Esempi: `smp1 3 10`
`smp1 1960:2 1982:4`
`smp1 +1 -1`
`smp1 x > 3000 --restrict`
`smp1 y > 3000 --restrict --replace`
`smp1 100 --random`

Reimposta l'intervallo del campione. Il nuovo intervallo può essere definito in vari modi. Nel primo modo (corrispondente ai primi due esempi precedenti) *oss-iniziale* e *oss-finale* devono essere coerenti con la periodicità dei dati. Una delle due può essere sostituita da un punto e virgola per lasciare intatto il valore attuale. Nel secondo modo, gli interi *i* e *j* (che possono essere positivi o negativi e vanno indicati con il segno) sono presi come spostamenti relativi ai punti iniziale e finale del campione in uso. Nel terzo modo, *variabile-dummy* deve essere una variabile indicatrice che assume solo valori 0 o 1 e il campione verrà ristretto alle osservazioni per cui la variabile dummy vale 1. Il quarto modo, che usa `--restrict`, limita il campione alle osservazioni che soddisfano la condizione Booleana specificata secondo la sintassi del comando [genr](#).

Con la forma `--no-missing`, se viene specificata una *lista-variabili*, vengono selezionate le osservazioni per cui tutte le variabili nella *lista-variabili* hanno valori validi in corrispondenza dell'osservazione; altrimenti, se non viene indicata alcuna *lista-variabili*, vengono selezionate le osservazioni per cui *tutte* le variabili hanno valori validi (non mancanti).

Con la forma `--random`, viene estratto casualmente dal dataset il numero indicato di osservazioni. Per essere in grado di replicare questa selezione, occorre per prima cosa impostare il seme del generatore di numeri casuali (si veda il comando [set](#)).

La forma finale, `smpl full`, ripristina l'intervallo completo del campione.

Si noti che i vincoli sul campione di solito sono cumulativi: il valore di riferimento di ogni comando `smpl` è il campione attuale. Se si vuole che il comando funzioni sostituendo i vincoli esistenti, così che ogni vincolo si aggiunga a quelli già impostati, occorre usare l'opzione `--replace` alla fine del comando.

La variabile interna `obs` può essere usata con la forma `--restrict` di `smpl` per escludere particolari osservazioni dal campione. Ad esempio,

```
smpl obs!=4 --restrict
```

scarterà la quarta osservazione. Se le osservazioni sono identificate da etichette,

```
smpl obs!="USA" --restrict
```

scarterà l'osservazione a cui è associata l'etichetta "USA".

Per le forme `--dummy`, `--restrict` e `--no-missing` di `smpl`, occorre tenere presente che tutte le informazioni "strutturali" contenute nel file dei dati (a proposito della struttura di serie storiche o di panel dei dati) vengono perse. È possibile reimpostare la struttura originale con il comando [setobs](#).

Si veda il [Capitolo 6](#) per ulteriori dettagli.

Accesso dal menù: /Campione

spearman

Argomenti: `x y`

Opzione: `--verbose` (mostra i dati ordinati)

Mostra il coefficiente di correlazione di rango di Spearman per le variabili *x* e *y*. Le variabili non devono essere state ordinate manualmente in precedenza, se ne occupa la funzione.

L'ordinamento automatico è dal massimo al minimo (ossia il valore massimo nei dati assume il rango 1). Se occorre invertire l'ordinamento, creare una variabile che è il negativo della variabile originale, ad esempio:

```
genr altx = -x
spearman altx y
```

Accesso dal menù: /Modello/SPEARMAN - Correlazione di rango

square

Argomento: *lista-variabili*

Opzione: `--cross` (genera anche i prodotti incrociati, oltre ai quadrati)

Genera nuove variabili che sono i quadrati delle variabili nella *lista-variabili* (con anche i prodotti incrociati, se si usa l'opzione `--cross`). Ad esempio, `square x y` genera `sq_x = x` al quadrato, `sq_y = y` al quadrato e (opzionalmente) `x_y = x` per `y`. Se una particolare variabile è una dummy, non ne viene fatto il quadrato, visto che si otterrebbe la stessa variabile.

Accesso dal menù: /Dati/Aggiungi variabili/Quadrati delle variabili selezionate

store

Argomenti: *file-dati* [*lista-variabili*]

Opzioni: `--csv` (usa il formato CSV)

`--omit-obs` (si veda oltre, a proposito del formato CSV)

`--gnu-octave` (usa il formato GNU Octave)

`--gnu-R` (usa il formato GNU R)

`--traditional` (usa il formato tradizionale ESL)

`--gzipped` (comprime con gzip)

`--dat` (usa il formato ASCII di PcGive)

`--database` (usa il formato database di gretl)

`--overwrite` (cfr oltre, a proposito del formato dei database)

Salva l'intero dataset, o un sottoinsieme delle variabili se è stata indicata una *lista-variabili*, nel file indicato con *file-dati*.

L'impostazione predefinita è di salvare i dati nel formato "interno" di gretl, ma le opzioni del comando permettono di usare formati alternativi. I dati CSV (Comma-Separated Values, dati separati da virgole) possono essere letti dai programmi di foglio elettronico e possono essere modificati con un editor di testi. I formati Octave, R e PcGive sono destinati ad essere usati con i rispettivi programmi. La compressione con gzip può essere utile per grandi dataset. Si veda il [Capitolo 4](#) per i dettagli sui vari formati.

L'opzione `--omit-obs` è applicabile solo quando si salvano dati in formato CSV. In modalità predefinita, se i dati sono serie storiche o panel, o se il dataset include marcatori per osservazioni specifiche, il file CSV comprende una prima colonna che identifica le osservazioni (ad esempio per data). Se si usa `--omit-obs`, questa colonna verrà omessa e verranno salvati solo i dati effettivi.

Si noti che le variabili scalari non saranno salvate automaticamente: per salvarle occorre includerle esplicitamente nella *lista-variabili*.

L'opzione di salvataggio in formato database di gretl è indicata se occorre costruire dei grandi dataset di serie, magari con frequenze diverse e diversi intervalli di osservazioni. Al momento questa opzione è disponibile solo per dati annuali, trimestrali o mensili. Salvando su un file che esiste già, il comportamento predefinito è quello di accodare le nuove serie al contenuto del database preesistente. In questo contesto, se una o più delle variabili da salvare hanno lo stesso nome di una delle variabili già presenti nel database si otterrà un messaggio di errore. L'opzione `--overwrite` permette invece di sovrascrivere eventuali variabili del dataset che hanno lo stesso nome delle nuove variabili, in modo che queste ultime rimpiazzino le variabili preesistenti.

Accesso dal menù: /File/Salva dati; /File/Esporta dati

summary

Argomento: [*lista-variabili*]

Mostra le statistiche descrittive per le variabili nella *lista-variabili*, o per tutte le variabili nel dataset, se non si indica una *lista-variabili*. L'output comprende media, deviazione standard, coefficiente di variazione (= deviazione standard / media), mediana, minimo, massimo, coefficiente di asimmetria, curtosi.

Accesso dal menù: /Dati/Statistiche descrittive

Accesso alternativo: Menù pop-up nella finestra principale

system

Forme `system method=stimatore`

alternative:

`system name=nome_sistema`

Argomenti: *tipo salva-variabili*

Esempi: `system name="Klein Model 1"`

`system method=sur`

`system method=sur save=resids`

`system method=3sls save=resids,fitted`

Inizia un sistema di equazioni. Esistono due versioni del comando, a seconda che si voglia salvare il sistema per poterlo stimare in più modi diversi, oppure stimare il sistema una volta sola.

Per salvare il sistema occorre dargli un nome, come nel primo esempio proposto (se il nome contiene spazi, occorre racchiuderlo tra virgolette). In questo caso, è possibile stimare il sistema con il comando [estimate](#). Una volta che il sistema è stato salvato, è possibile imporre dei vincoli su di esso (compresi vincoli incrociati tra equazioni) usando il comando [restrict](#).

In alternativa, è possibile indicare uno stimatore per il sistema usando `method=` seguito da una stringa che identifica uno degli stimatori supportati: `ols` (ordinary least squares - minimi quadrati ordinari), `tsls` (two-stage least squares - minimi quadrati a due stadi), `sur` (seemingly unrelated regressions - regressioni apparentemente non collegate), `3sls` (three-stage least squares - minimi quadrati a tre stadi), `fiml` (full information maximum likelihood - massima verosimiglianza con informazione completa) o `liml` (limited information maximum likelihood - massima verosimiglianza con informazione limitata). In questo caso, il sistema viene stimato appena completata la sua definizione.

Un sistema di equazioni termina con la riga `end system`. All'interno del sistema possono essere definiti i quattro tipi di istruzioni seguenti.

§ [equation](#): specifica un'equazione del sistema. Occorre indicarne almeno due.

§ `instr`: per i sistemi da stimare con i minimi quadrati a tre stadi, indica la lista degli strumenti (indicati dal nome o dal numero della variabile). In alternativa, è possibile fornire questa informazione nella riga `equation` usando la stessa sintassi accettata dal comando [tsls](#).

§ `endog`: per i sistemi di equazioni simultanee, indica la lista delle variabili endogene. È indicato principalmente per la stima FIML, ma può essere usato anche nella stima minimi quadrati a tre stadi al posto dell'istruzione `instr`: in questo modo tutte le variabili non identificate come endogene verranno usate come strumenti.

§ `identity`: per la stima FIML, un'identità che collega due o più variabili del sistema. Questo tipo di istruzione è ignorata se viene usato uno stimatore diverso da FIML.

Nel campo opzionale `save=` del comando è possibile specificare se salvare i residui (`resids`) e/o i valori stimati (`fitted`).

Per esempi completi di specificazione e stima di un sistema di equazioni, si vedano gli script `klein.inp` e `greene14_2.inp` forniti con la distribuzione di `gretl`.

tabprint

Argomento: `[ -f nomefile ]`

Opzione: `--complete` (crea un documento completo)

Va eseguito dopo la stima di un modello. Stampa il modello stimato sotto forma di tabella LaTeX. Se viene specificato un nome di file dopo l'opzione `-f`, l'output viene scritto nel file, altrimenti viene scritto in un file col nome `model_N.tex`, dove `N` è il numero dei modelli stimati finora nella sessione in corso. Si veda anche [eqnprint](#).

Usando l'opzione `--complete`, il file LaTeX è un documento completo, pronto per essere processato; altrimenti il file va incluso in un documento.

Accesso dal menù: Finestra del modello, /LaTeX

testuhat

Deve seguire un comando di stima. Mostra la distribuzione di frequenza dei residui del modello, insieme a un test chi-quadro per la normalità, basato sulla procedura suggerita da Doornik e Hansen (1984).

Accesso dal menù: Finestra del modello, /Test/Normalità dei residui

tobit

Argomenti: *variabile-dipendente variabili-indipendenti*

Opzioni: `--vcv` (mostra la matrice di covarianza)

`--verbose` (mostra i dettagli delle iterazioni)

Stima un modello Tobit. Il modello può essere appropriato quando la variabile dipendente è "troncata". Ad esempio, vengono osservati valori positivi o nulli della spesa dei consumatori per beni durevoli, ma non valori negativi; tuttavia le decisioni di spesa possono essere pensate come derivanti da una propensione al consumo, sottostante e non osservata, che può anche essere negativa in alcuni casi. Per i dettagli si veda il capitolo 20 di *Econometric Analysis* di Greene.

Accesso dal menù: /Modello/Tobit

tsls

Argomenti: *variabile-dipendente variabili-indipendenti ; strumenti*

Opzioni: `--vcv` (mostra la matrice di covarianza)

`--robust` (errori standard robusti)

Esempio: `tsls y1 0 y2 y3 x1 x2 ; 0 x1 x2 x3 x4 x5 x6`

Calcola le stime minimi quadrati a due stadi (TSLS o IV), date la *variabile-dipendente* e la lista di *variabili-indipendenti* (incluse le variabili esogene) nell'equazione strutturale per cui sono richieste le stime TSLS; *strumenti* è la lista completa delle variabili esogene e predeterminate in tutte le equazioni. Se la lista degli *strumenti* non è lunga almeno quanto quella delle *variabili-indipendenti*, il modello non è identificato.

Nell'esempio precedente, le `y` sono le variabili endogene e le `x` sono le variabili esogene e predeterminate.

Accesso dal menù: /Modello/TSLs - Minimi quadrati a due stadi

var

Argomenti: *ordine lista-variabili ; lista-deterministica*

Opzioni: *--nc* (non include una costante)
--seasonals (include variabili dummy stagionali)
--robust (errori standard robusti)
--impulse-responses (mostra impulso-risposta)
--variance-decomp (mostra decomposizioni della varianza delle previsioni)

Esempi: *var 4 x1 x2 x3 ; time mydum*
var 4 x1 x2 x3 --seasonals

Imposta e stima (usando OLS) un'autoregressione vettoriale (VAR). Il primo argomento specifica l'ordine di ritardo, quindi segue l'impostazione della prima equazione. Non occorre includere i ritardi tra gli elementi della *lista-variabili*: verranno aggiunti automaticamente. Il punto e virgola separa le variabili stocastiche, per cui verrà incluso un numero di ritardi pari a *ordine*, dai termini deterministici o esogeni presenti nella *lista-deterministica*.

gretl è in grado di riconoscere le più comuni variabili deterministiche (trend temporale, variabili dummy con valori 0 o 1), quindi queste non devono necessariamente essere elencate dopo il punto e virgola. Variabili deterministiche più complesse (ad es. un trend temporale moltiplicato per una variabile dummy) devono invece essere indicate esplicitamente. Si noti che viene inclusa automaticamente una costante, a meno che non si usi l'opzione *--nc*; inoltre è possibile aggiungere variabili dummy stagionali con l'opzione *--seasonals*.

Viene calcolata una regressione separata per ognuna delle variabili nella *lista-variabili*. Il risultato di ogni equazione include i test *F* per i vincoli di uguaglianza a zero su tutti i ritardi delle variabili, un test *F* per la significatività del ritardo massimo e, se è stata usata l'opzione *--impulse-responses*, la scomposizione della varianza della previsione e le funzioni di impulso-risposta.

Le decomposizioni della varianza della previsione e le funzioni di impulso-risposta sono basate sulla decomposizione di Cholesky della matrice di covarianza contemporanea, e in questo contesto l'ordine in cui vengono date le variabili stocastiche conta. La prima variabile nella lista viene considerata come la "più esogena" all'interno del periodo. L'orizzonte per le decomposizioni della varianza e le funzioni di impulso-risposta può essere impostato usando il comando [set](#).

Accesso dal menù: /Modello/Serie storiche/VAR - Autoregressione vettoriale

varlist

Mostra un elenco delle variabili disponibili. *list* e *ls* sono sinonimi.

vartest

Argomenti: *var1 var2*

Calcola la statistica *F* per l'ipotesi nulla che le varianze della popolazione per le variabili *var1* e *var2* siano uguali e mostra il p-value.

Accesso dal menù: /Dati/Differenza delle varianze

vecm

Argomenti: *ordine rango lista-variabili*

Opzioni: *--nc* (senza costante)
--rc (costante vincolata)
--crt (costante e trend vincolato)
--ct (costante e trend non vincolato)
--seasonals (include dummy stagionali centrate)

Esempi: *vecm 4 1 Y1 Y2 Y3*
vecm 3 2 Y1 Y2 Y3 --rc

Un VECM è un tipo di autoregressione vettoriale, o VAR (si veda [var](#)), applicabile quando le variabili del modello sono individualmente integrate di ordine 1 (ossia, sono “random walk” con o senza deriva), ma esibiscono cointegrazione. Questo comando è strettamente connesso al test di Johansen per la cointegrazione (si veda [coint2](#)).

Il parametro *ordine* rappresenta l'ordine di ritardo del sistema VAR. Il numero di ritardi nel VECM (dove la variabile dipendente è data da una differenza prima) è pari a *ordine* meno uno.

Il parametro *rango* rappresenta il rango di cointegrazione, o in altre parole il numero di vettori di cointegrazione. Questo deve essere maggiore di zero e minore o uguale (in genere minore) al numero di variabili endogene contenute nella *lista variabili*.

La *lista-variabili* rappresenta l'elenco delle variabili endogene, nei livelli. L'inclusione di trend deterministici nel modello è controllata dalle opzioni del comando. Se non si indica alcuna opzione, viene inclusa una “costante non vincolata”, che permette la presenza di un'intercetta diversa da zero nelle relazioni di cointegrazione e di un trend nei livelli delle variabili endogene. Nella letteratura originata dal lavoro di Johansen (si veda ad esempio il suo libro del 1995), si fa riferimento a questo come al “caso 3”. Le prime quattro opzioni mostrate sopra, che sono mutualmente esclusive, producono rispettivamente i casi 1, 2, 4 e 5. Il significato di questi casi e i criteri per scegliere tra di essi sono spiegati in [Capitolo 12](#).

L'opzione *--seasonals*, che può accompagnare una qualsiasi delle altre opzioni, specifica l'inclusione di un gruppo di variabili dummy stagionali centrate. Questa opzione è disponibile solo per dati trimestrali o mensili.

Il primo degli esempi mostrati sopra specifica un VECM con ordine di ritardo pari a 4 e un unico vettore di cointegrazione. Le variabili endogene sono Y1, Y2 e Y3. Il secondo esempio usa le stesse variabili ma specifica un ritardo di ordine 3 e due vettori di cointegrazione, oltre a specificare una “costante vincolata”, che è appropriata se i vettori di cointegrazione possono avere un'intercetta diversa da zero, ma le variabili Y non hanno trend.

Accesso dal menù: /Model/Time series/VECM

vif

Deve seguire la stima di un modello che includa almeno due variabili indipendenti. Calcola e mostra i fattori di inflazione della varianza (Variance Inflation Factors - VIF) per i regressori. Il VIF per il regressore *j* è definito come

$$\frac{1}{1 - R_j^2}$$

dove R_j è il coefficiente di correlazione multipla tra il regressore *j* e gli altri regressori. Il fattore ha un valore minimo di 1.0 quando la variabile in questione è ortogonale alle altre variabili indipendenti. Neter, Wasserman e Kutner (1990) suggeriscono di usare il VIF maggiore come test diagnostico per la collinearità; un valore superiore a 10 è in genere considerato indice di un grado di collinearità problematico.

Accesso dal menù: Finestra del modello, /Test/collinearità

wls

Argomenti: *variabile-pesi variabile-dipendente variabili-indipendenti*

Opzione: *--vcv* (mostra la matrice di covarianza)

Calcola stime con minimi quadrati ponderati, prendendo i pesi da *variabile-pesi*. In pratica viene calcolata una regressione OLS di *variabile-pesi * variabile-dipendente* rispetto a *variabile-pesi * variabili-indipendenti*. Se la *variabile-pesi* è una variabile dummy, la procedura equivale a eliminare tutte le osservazioni per cui *variabile-pesi* vale zero.

Accesso dal menù: /Modello/WLS - Minimi quadrati ponderati

Stimatori e test: sommario

La [Tabella 14-2](#) mostra gli estimatori disponibili nel menù Modello nella finestra principale di gret1. I corrispondenti comandi (se esistono) sono mostrati tra parentesi; per i dettagli, consultare [la Sezione Comandi di gret1](#).

Tabella 14-2. Stimatori

Stimatore	Commento
Minimi quadrati ordinari (ols)	Lo stimatore principale
Minimi quadrati ponderati (wls)	Eteroschedasticità, esclusione di osservazioni scelte
HCCM (hccm)	Matrice di covarianza corretta per l'eteroschedasticità
Stime corrette per l'eteroschedasticità (hsk)	Minimi quadrati ponderati basati sulla varianza prevista dell'errore
Cochrane-Orcutt (corc)	Autocorrelazione del prim'ordine
Hildreth-Lu (h1lu)	Autocorrelazione del prim'ordine
Stima autoregressiva (ar)	Autocorrelazione di ordine superiore (Cochrane-Orcutt generalizzato)
Autoregressione vettoriale (var)	Sistemi di equazioni di serie storiche
Test di cointegrazione (coint)	Relazioni di lungo periodo tra serie
Minimi quadrati a due stadi (tsls)	Equazioni simultanee
Minimi quadrati non-lineari (nls)	Modelli non-lineari
Logit (logit)	Variabile dipendente binaria (distribuzione logistica)
Probit (probit)	Variabile dipendente binaria (distribuzione normale)
Minime deviazioni assolute (lad)	Alternativa ai minimi quadrati
Correlazione di rango (spearman)	Correlazione con dati ordinali
Pooled OLS (pooled)	Stima OLS per dati cross-section e serie storiche mischiati
OLS a precisione multipla (mpols)	Stima OLS usando aritmetica a precisione multipla

La [Tabella 14-3](#) mostra i test disponibili nel menù Test della finestra dei modelli stimati.

Tabella 14-3. Test per i modelli

Test	Comando corrispondente
Variabili omesse (test F con OLS)	omit
Variabili aggiunte (test F con OLS)	add
Nonlinearità (quadrati)	lmtest --squares
Nonlinearità (logaritmi)	lmtest --logs
Nonlinearità (RESET di Ramsey)	reset
Eteroschedasticità (test di White)	lmtest --white
Osservazioni influenti	leverage
Autocorrelazione (ordine pari alla frequenza)	lmtest --autocorr
Chow (break strutturale)	chow
CUSUM (stabilità dei parametri)	cusum
ARCH (eteroschedasticità condizionale)	arch
Normalità dei residui	testuhat
Diagnosi panel	hausman

La [Tabella 14-4](#) mostra la corrispondenza tra le opzioni dei comandi in formato lungo e quelle in formato corto.

Tabella 14-4. Opzioni in forma lunga e corta

Comando	Opzione	Effetto	Forma corta
arma	--x-12-arima	stima con X-12-ARIMA	-x
	--verbose	mostra i dettagli delle iterazioni	-v
coint2	--verbose	mostra i dettagli dei VAR	-v
eqnprint, tabprint	--complete	documento TeX completo	-o
fcasterr	--plot	mostra il grafico	-o
gnuplot	--with-lines	grafico con linee	-o
	--with-impulses	grafico con impulsi	-m
	--suppress-fitted	non mostrare la retta stimata	-s
	--dummy	separazione dei fattori	-z
import	--box1	importa dati BOX1	-o
leverage	--save	salva i valori nel dataset	-s
lmtest	--logs	non-linearità (logaritmi)	-l
	--squares	non-linearità (quadrati)	-s
	--white	test di White (eteroschedasticità)	-w
	--autocorr	correlazione seriale	-m
meantest	--unequal-vars	assume varianze diverse	-o
leverage	--save	salva i valori nel dataset	-s
ols	--robust	errori standard robusti	-r
	--vcv	mostra la matrice di covarianza	-o
	--quiet	non mostra i risultati	-q
outfile	--write	apre il file in scrittura	-w
	--append	aggiunge al file esistente	-a

Comando	Opzione	Effetto	Forma corta
	--close	chiude il file	-c
panel	--time-series	pila di dati serie storiche	-s
	--cross-section	pila di dati cross-section	-c
pca	--save	salva gli autovettori utili	-s
	--save-all	salva tutti gli autovettori	-a
pergm	--bartlett	usa la finestra dei ritardi di Bartlett	-o
plot	--one-scale	non scala le variabili	-o
print	--byobs	variabili per colonna	-o
	--ten	usa 10 cifre significative	-t
smp1	--dummy	restringe usando una variabile dummy	-o
	--no-missing	usa solo le osservazioni complete	-m
	--restrict	usa una condizione booleana	-r
spearman	--verbose	mostra i dati originali ordinati	-o
square	--cross	genera i prodotti incrociati	-o
store	--csv	valori separati da virgola	-c
	--traditional	formato ESL tradizionale	-t
	--gnu-octave	formato GNU Octave	-m
	--gnu-R	formato GNU R	-r
	--gzipped	compressione gzip	-z
var	--quiet	non mostra tutti i risultati	-q
Comandi modello	--vcv	mostra la matrice di covarianza	-o

Capitolo 15. Risoluzione dei problemi

Segnalazione dei bug

Le segnalazioni dei bug sono benvenute. È difficile trovare errori di calcolo in `gretl` (ma questa affermazione non costituisce alcuna sorta di garanzia), ma è possibile imbattersi in bug o stranezze nel comportamento dell'interfaccia grafica. Si tenga presente che l'utilità delle segnalazioni aumenta quanto più si è precisi nella descrizione: cosa *esattamente* non funziona, in che condizioni, con quale sistema operativo? Se si ricevono messaggi di errore, cosa dicono esattamente?

Programmi ausiliari

Come detto in precedenza, `gretl` richiama alcuni altri programmi per eseguire alcune operazioni (`gnuplot` per i grafici, `LaTeX` per la stampa ad alta qualità dei risultati delle regressioni, `GNU R`). Se succede qualche problema durante questi collegamenti esterni, non è sempre facile fornire un messaggio di errore abbastanza informativo. Se il problema si verifica durante l'uso di `gretl` con l'interfaccia grafica, può essere utile avviare `gretl` dal prompt dei comandi (ad es. da una finestra di `xterm` nel sistema X window, o da un "Prompt di MS-DOS" in MS Windows, nel qual caso occorre digitare `gretlw32.exe`), invece che da un menù o da un'icona: la finestra del terminale potrà quindi contenere messaggi di errore aggiuntivi.

Si tenga anche presente che nella maggior parte dei casi `gretl` assume che i programmi in questione siano disponibili nel "percorso di esecuzione" (path) dell'utente, ossia che possano essere invocati semplicemente con il nome del programma, senza indicare il percorso completo ¹. Quindi se un certo programma non si avvia, conviene provare ad eseguirlo da un prompt dei comandi, come descritto qui sotto.

Sistema	Grafica	Stampa	GNU R
Sistema X window	<code>gnuplot</code>	<code>latex, xdv</code>	<code>R</code>
MS Windows	<code>wgnuplot.exe</code>	<code>latex, windvi</code>	<code>RGui.exe</code>

Se il programma non si avvia dal prompt, non è un problema di `gretl`: probabilmente la directory principale del programma non è contenuta nel path dell'utente, oppure il programma non è stato installato correttamente. Per i dettagli su come modificare il path dell'utente, si veda la documentazione o l'aiuto online per il proprio sistema operativo.

1. L'eccezione a questa regola è costituita dall'invocazione di `gnuplot` in MS Windows, dove occorre indicare il percorso completo del programma.

Capitolo 16. L'interfaccia a riga di comando

Gretl sul terminale

Il pacchetto `gretl` include il programma a riga di comando `gretlcli`. Su Linux è possibile eseguirlo in un terminale testuale o in un `xterm` (o simili), mentre su MS Windows può essere eseguito in una finestra di terminale (quella che di solito viene chiamata “prompt di MS-DOS”). `gretlcli` ha un file di aiuto, a cui si accede inserendo il comando “help” al prompt. Inoltre, può essere usato in modalità batch, inviando i risultati direttamente a un file (si veda [la Sezione `gretlcli` nel Capitolo 13](#)).

Se `gretlcli` è linkato alla libreria `readline` (ciò avviene automaticamente nella versione MS Windows, si veda anche l'[Appendice B](#)), è possibile richiamare e modificare i comandi già eseguiti, inoltre è disponibile il completamento automatico dei comandi. Per muoversi nella lista dei comandi già eseguiti è possibile usare i tasti Freccia Su/Giù, mentre per muoversi sulla riga di comando è possibile usare i tasti Freccia a destra/sinistra e le scorciatoie in stile Emacs¹. Le scorciatoie più usate sono:

Scorciatoia	Effetto
<code>Ctrl-a</code>	Va all'inizio della riga
<code>Ctrl-e</code>	Va alla fine della riga
<code>Ctrl-d</code>	Cancella il carattere a destra

dove “`Ctrl-a`” significa premere il tasto “a” mentre si tiene premuto il tasto “`Ctrl`”. Quindi se si vuole correggere qualcosa all'inizio della riga di comando, *non occorre* usare il tasto di cancellazione all'indietro su tutta la riga, basta saltare all'inizio della riga e aggiungere o cancellare i caratteri desiderati.

Inserendo le prime lettere del nome di un comando e premendo il tasto `Tab`, `readline` cercherà di completare automaticamente il nome del comando. Se esiste un solo modo di completare le lettere inserite, verrà inserito il comando corrispondente, altrimenti premendo `Tab` una seconda volta verrà visualizzata una lista delle alternative possibili.

Differenze con ESL di Ramanathan

`gretlcli` ha ereditato la sintassi di base dei comandi dal programma ESL di Ramu Ramanathan e gli script di comandi sviluppati per ESL dovrebbero essere utilizzabili con con pochi cambiamenti: le uniche cose a cui stare attenti sono i comandi multilinea e il comando `freq`.

§ In ESL viene usato un punto e virgola come terminatore per molti comandi. In `gretlcli` ho deciso di rimuovere questa caratteristica: il carattere punto e virgola è semplicemente ignorato, a parte in alcuni casi speciali in cui ha un significato particolare: come separatore per due liste nei comandi `ar` e `tsls`, e come segnaposto per l'osservazione iniziale o finale immutata nel comando `smp1`. In ESL il punto e virgola dà la possibilità di spezzare le righe di comando su più righe dello schermo; in `gretlcli` si ottiene questo risultato inserendo una barra rovesciata `\` alla fine della riga da spezzare.

§ Per quanto riguarda `freq`, al momento non è possibile specificare intervalli personalizzati, come avviene in ESL. Inoltre, ai risultati del comando è stato aggiunto un test chi-quadro di normalità.

Si noti anche che i comandi per usare la modalità batch sono stati semplificati. In ESL si usava, ad esempio:

```
esl -b filedati < fileinput > fileoutput
```

1. In realtà quelli mostrati sono i valori predefiniti, ma possono essere modificati: si veda il manuale di `readline`.

mentre in `gretlcli` si usa:

```
gretlcli -b fileinput > fileoutput
```

Il file di input viene trattato come un argomento del programma, mentre il file di dati su cui operare va indicato all'interno del file di input, usando il comando `open datafile` o il commento speciale `(* ! filedati *)`.

Appendice A. Dettagli sui file di dati

Formato interno di base

Nel formato dati interno di gret1 un dataset è immagazzinato in XML (extensible mark-up language). I file di dati sono conformi al semplice DTD (document type definition) contenuto nel file `gret1data.dtd`, fornito con la distribuzione di gret1 e installato nella directory dei dati di sistema (ad es. `/usr/share/gret1/data` su Linux). I file di dati possono essere non compressi o compressi con gzip; oltre ai valori dei dati, contengono altre informazioni aggiuntive, come i nomi e le descrizioni delle variabili, la frequenza dei dati e così via.

La maggior parte degli utenti probabilmente non avrà bisogno di leggere o scrivere questi file, se non usando gret1 stesso, ma se si vuole manipolarli usando altri strumenti, può essere utile esaminare il DTD e qualcuno dei file di esempio forniti: il file `data4-1.gdt` dà un semplice esempio, il file `data4-10.gdt` invece contiene anche delle etichette per le osservazioni.

Formato tradizionale di ESL

Per compatibilità all'indietro, gret1 può elaborare anche file di dati nel formato “tradizionale” ereditato dal programma ESL di Ramanathan. In questo formato (che era quello predefinito nelle versioni di gret1 precedenti alla 0.98) un dataset è rappresentato da due file: uno contiene i dati veri e propri, l'altro contiene la descrizione dei dati e le modalità per la loro lettura. Per essere più specifici:

1. *Dati veri e propri*: una matrice rettangolare di numeri separati da spazio vuoto. Ogni colonna rappresenta una variabile, ogni riga un'osservazione per ognuna delle variabili (come in un foglio elettronico). Le colonne dei dati possono essere separate da spazi o caratteri tab. Il nome del file deve avere l'estensione `.gdt` e il file di solito è di tipo ASCII (testo semplice), ma può essere anche compresso con gzip, per occupare meno spazio su disco. È possibile inserire commenti in un file di dati: se una riga comincia con un carattere cancelletto (`#`), l'intera riga viene ignorata (questo comportamento è coerente con i file di dati di gnuplot e di octave).
2. *Descrizione*: il file di dati deve essere accompagnato da un file di descrizioni, che ha lo stesso nome di base del file di dati, ma l'estensione `.hdr`. Questo file contiene, nell'ordine:
 - (Opzionale) *commenti* sui dati, introdotti dalla stringa di apertura (`*` e conclusi dalla stringa `*`); ognuna di queste stringhe deve essere presente da sola su una riga.
 - (Richiesta) lista dei *nomi delle variabili* presenti nel file, separati da spazio bianco. I nomi sono limitati a 8 caratteri, devono iniziare con una lettera e possono contenere solo caratteri alfanumerici e il carattere trattino basso, `_`. La lista può continuare su più righe e deve essere chiusa da un punto e virgola `;`.
 - (Richiesta) riga *osservazioni*, nel formato `1 1 85`. Il primo elemento indica la frequenza dei dati (1 per dati non datati o annuali, 4 per trimestrali, 12 per mensili). Gli altri due elementi indicano le osservazioni iniziale e finale: di solito questi saranno pari a 1 e al numero delle osservazioni, per i dati non datati. Per le serie storiche, è possibile usare date nella forma `1959.1` (trimestrale, una cifra dopo il punto) o `1967.03` (mensile, due cifre dopo il punto). Si veda il [Capitolo 7](#) per l'uso speciale di questa riga nel caso dei dati panel.
 - La parola chiave `BYOBS`.

Ecco un esempio di un file di descrizione dei dati ben scritto.

```
(*
DATA9-6:
Dati su log(moneta), log(reddito) e tasso di interesse USA.
```

Fonte: Stock and Watson (1993) Econometrica
(dati grezzi) Il periodo è 1900-1989 (dati annuali).
Dati composti da Graham Elliott.
*)
lmoneta lreddito tassint ;
1 1900 1989 BYOBS

Il corrispondente file di dati contiene tre colonne di dati, ognuna con 90 osservazioni.

Ci sono altre tre caratteristiche del formato dati “tradizionale” degne di nota.

1. Se la parola chiave BYOBS è sostituita da BYVAR ed è seguita dalla parola chiave BINARY, significa che il corrispondente file di dati è in formato binario. Questo tipo di file di dati può essere scritto da gretlcli usando il comando store con l'opzione -s (precisione singola) o l'opzione -o (precisione doppia).
2. Se BYOBS è seguita dalla parola chiave MARKERS, gretl si aspetta un file di dati in cui la *prima colonna* contiene stringhe (lunghe al massimo 8 caratteri) usate per identificare le osservazioni. Queste possono essere utili nel caso dei dati di tipo cross-section in cui le unità di osservazione sono identificabili: regioni, stati, città ecc. Un altro caso è quello delle serie storiche irregolari, come ad esempio i dati giornalieri delle quotazioni azionarie, in cui mancano i giorni in cui non avvengono contrattazioni: in questo caso, le osservazioni possono essere marcate con una stringa che rappresenta la data, come ad esempio 10/01/98 (si ricordi il limite di 8 caratteri). Si noti che le opzioni BINARY e MARKERS sono mutualmente esclusive, e si noti anche che i “marcatori” non sono considerati come una variabile: questa colonna non compare nell'elenco delle variabili nel file di descrizione dei dati.
3. Se esiste un file con lo stesso nome di base del file di dati e di quello delle descrizioni, ma con l'estensione .lbl, esso viene letto e usato per riempire le etichette descrittive per le serie di dati. Il formato del file di etichette è semplice: ogni riga contiene il nome di una variabile (indicata nel file di descrizione) seguito da uno o più spazi, seguito dall'etichetta descrittiva. Ecco un esempio: prezzo Indice dei prezzi delle auto nuove, anno base 1982

Se si vuole salvare i dati nel formato tradizionale, occorre usare l'opzione -t con il comando store, nella versione a riga di comando del programma o nella finestra del terminale della versione grafica del programma.

Dettagli sui database binari

Un database di gretl consiste di due parti: un file indice ASCII (con l'estensione .idx) che contiene informazioni sulle serie, e un file binario (con estensione .bin) che contiene i dati veri e propri. Ecco due esempi di voci contenute in un file idx:

```
GOM910  Indice composto da 11 indicatori principali (1987=100)
M 1948.01 - 1995.11  n = 575
currba1 Bilancia dei pagamenti: parte corrente; corretta per la stagionalità
Q 1960.1 - 1999.4  n = 160
```

Il primo campo è il nome della serie, il secondo è una descrizione della serie (al massimo di 128 caratteri). Sulla seconda riga, il primo campo è un codice di frequenza: M per mensile, Q per trimestrale, A per annuale, B per giornalieri-lavorativi (giornalieri con cinque giorni a settimana) e D per giornalieri (sette giorni a settimana). Al momento non vengono accettati altri codici di frequenza. Quindi segue la data iniziale (con due cifre che seguono il punto nei dati mensili, una per i dati trimestrali, nessuna per quelli annuali), uno spazio, un trattino, un altro spazio, la data finale, la stringa “n = ” e il numero delle osservazioni. Nel caso di dati giornalieri, le date iniziale e finale devono essere indicate nella forma YYYY/MM/DD. Questo formato deve essere rispettato alla lettera.

Opzionalmente, la prima riga del file indice può contenere un breve commento (al massimo 64 caratteri) a proposito della fonte e del tipo dei dati, che segue un carattere cancellato. Ad esempio:

Banca Centrale Europea (tassi di interesse)

Il corrispondente database binario contiene i valori dei dati, rappresentati come “float”, ossia come numeri in virgola mobile a precisione singola, che tipicamente occupano quattro byte ciascuno. I numeri sono immagazzinati “per variabile”, così che i primi n numeri sono le osservazioni della variabile 1, i successivi m sono le osservazioni per la variabile 2 e così via.

Appendice B. Note tecniche

Gretl è scritto nel linguaggio di programmazione C. Ho aderito nel modo più stretto possibile allo standard C ISO/ANSI (C89), anche se l'interfaccia grafica e alcune altre componenti devono fare uso necessariamente di estensioni specifiche per certe piattaforme.

gretl è stato sviluppato sotto Linux. La libreria condivisa e il client a riga di comando dovrebbero essere compilabili su qualsiasi piattaforma che (a) supporti l'ISO/ANSI C e (b) abbia installate le seguenti librerie: zlib (per la compressione dei dati), libxml (manipolazione dell'XML) e LAPACK (supporto per l'algebra lineare). La home page di zlib si trova a info-zip.org, libxml si trova a xmlsoft.org, mentre LAPACK è a netlib.org. Se il sistema ospita anche la libreria GNU readline, essa sarà usata per `gretcli`, fornendo una riga di comando molto più comoda da utilizzare. Si veda la home page di readline.

Il client grafico dovrebbe essere compilabile e utilizzabile su ogni sistema che, oltre ai requisiti visti sopra, offra la libreria GTK nella versione 1.2.3 o superiore (si veda gtk.org). Al momento della scrittura di questo manuale esistono due varianti principali delle librerie GTK: la serie 1,2 e la serie 2.0, lanciata nell'estate 2002. Poiché queste varianti sono mutualmente incompatibili, `gretl` può essere compilato usando una sola di esse; il pacchetto sorgente contiene due directory: `gui` per GTK 1.2 e `gui2` per GTK 2.0. Raccomando l'uso di GTK 2.0 se è disponibile, visto che offre molti vantaggi rispetto a GTK 1.2.

`gretl` chiama `gnuplot` per produrre grafici. È possibile trovare `gnuplot` a gnuplot.info. Al momento della scrittura di questo manuale, la versione ufficiale più recente di `gnuplot` è la 4.0 (dell'Aprile 2004). La versione MS Windows di `gretl` comprende la versione 4.0 di `gnuplot` per Windows; sul sito web di `gretl` è possibile trovare un pacchetto rpm di `gnuplot` 3.8j0 per sistemi Linux x86.

Alcune funzionalità di `gretl` fanno uso della libreria `gtkextra` di Adrian Feguin. È possibile trovare `gtkextra` a gtkextra.sourceforge.net. Le parti rilevanti di questo pacchetto sono incluse (in forma leggermente modificata) con la distribuzione sorgente di `gretl`.

Una versione binaria del programma è disponibile per la piattaforma Microsoft Windows (versione 32-bit, ossia Windows 95 o superiore). Questa versione è cross-compilata su Linux usando `mingw` (il compilatore GNU C, `gcc`, portato per l'uso con `win32`) e collegata alla libreria C di Microsoft C, `msvcrt.dll`. Utilizza il port di GTK 2.0 su `win32` di Tor Lillqvist. L'installatore per Windows (libero, open-source) è a cura di Jordan Russell (jrsoftware.org).

Spero che gli utenti con conoscenze di programmazione possano considerare `gretl` abbastanza interessante e degno di essere migliorato ed esteso. La documentazione dell'API `libgretl` non è ancora completa, ma è possibile trovare alcuni dettagli seguendo il link "Libgretl API docs" sul sito web di `gretl`.

Appendice C. Accuratezza numerica

`gretl` usa aritmetica a doppia precisione, ad eccezione del plugin per precisione multipla invocabile con il comando del menù “Modello, Minimi quadrati in alta precisione” che rappresenta i valori in virgola mobile usando un numero di bit indicato dalla variabile di ambiente `GRET_MP_BITS` (valore predefinito: 256). Le equazioni normali dei minimi quadrati sono risolte in modo predefinito usando la decomposizione di Cholesky, che è abbastanza accurata per la maggior parte delle applicazioni (c'è anche l'opzione di usare la decomposizione QR). Il programma è stato testato abbastanza approfonditamente con i dataset di riferimento forniti dal NIST (il National Institute of Standards and Technology statunitense) e un rapporto completo dei risultati si trova sul sito web di `gretl` (seguendo il link “Accuratezza numerica”).

Nell'Ottobre del 2002 ho avuto un utile scambio di idee con Giovanni Baiocchi e Walter Distaso, che stavano scrivendo una recensione di `gretl` per il *Journal of Applied Econometrics* e con James MacKinnon, responsabile delle recensioni di software per il giornale¹ e sono grato a Baiocchi e Distaso per il loro accurato esame del programma, che ha suggerito le seguenti modifiche.

1. I recensori hanno trovato un bug nel “calcolatore di p-value” di `gretl`, perchè il programma mostrava il complemento della probabilità corretta per i valori negativi di z . Il bug è stato corretto nella versione 0.998 del programma (pubblicata il 9 Luglio 2002).
2. È stato anche notato che il calcolatore di p-value produceva risultati non accurati per valori estremi di x (ad es. valori da 8 a 10 nella distribuzione t con 100 gradi di libertà). Anche questo problema è stato risolto nella versione 0.998 di `gretl`, inserendo codice più accurato per la distribuzione di probabilità.
3. I recensori hanno notato un difetto nella presentazione dei coefficienti di regressione in `gretl`, perchè alcuni coefficienti venivano mostrati con un numero troppo basso di cifre significative. Il difetto è stato corretto nella versione 0.999 (pubblicata il 25 Agosto 2002): ora tutte le statistiche associate con la regressione sono mostrate con 6 cifre significative.
4. I test eseguiti dai recensori indicavano che l'accuratezza numerica di `gretl` su MS Windows è minore che su Linux, dove io avevo svolto i miei test. Ad esempio, i dati Longley (un dataset notoriamente “mal-condizionato” usato spesso per testare i programmi econometrici) la versione Windows di `gretl` produceva alcuni coefficienti con la settima cifra errata, mentre gli stessi coefficienti risultavano corretti su Linux. Questa anomalia è stata corretta nella versione 1.0pre3 di `gretl` (pubblicata il 10 Ottobre 2002).

La versione attuale di `gretl` comprende un “plugin” che esegue la suite NIST di test della regressione lineare. È possibile trovarlo nel menù “Utilità” della finestra principale. Quando viene eseguito questo test, un'introduzione spiega qual è il risultato atteso: se si esegue il test e si ottengono risultati diversi da quelli attesi, si prega di inviare un bug report a cottrell@wfu.edu.

Come detto sopra, tutte le statistiche di regressione sono mostrate con 6 cifre significative nella versione attuale di `gretl` (tranne quando viene usato il plugin per precisione multipla, e allora i risultati sono mostrati con 12 cifre). Se occorre esaminare un valore più da vicino, basta per prima cosa salvarlo (si veda il comando [genr](#)) e poi visualizzarlo usando il comando `print --ten` (si veda il [Capitolo 14](#)): verrà mostrato con 10 cifre significative.

1. La recensione è poi stata pubblicata; si veda Baiocchi e Distaso (2003).

Appendice D. Analisi econometrica avanzata con software libero

Come descritto nel manuale, `gretl` offre una raccolta abbastanza completa di stimatori basati sui minimi quadrati e alcuni stimatori aggiuntivi, come `logit`, `probit` e le minime deviazioni assolute, ma gli utenti avanzati possono trovare questa lista troppo ristretta.

Senza dubbio, alcuni utenti avanzati preferiranno scrivere da soli il proprio codice statistico usando qualche linguaggio di programmazione come C, C++ o Fortran. Un'altra opzione è quella di usare un linguaggio di livello più alto che offre funzioni per la manipolazione di matrici e altre routine statistiche incluse o disponibili come pacchetti aggiuntivi. Se si è attratti da quest'ultima opzione e si intende usare software libero, raccomando di dare un'occhiata a GNU R (r-project.org) o GNU Octave (www.octave.org). Questi programmi sono molto simili rispettivamente ai programmi commerciali S e Matlab.

Come già detto nel manuale, `gretl` offre la possibilità di esportare i dati nei formati di Octave e R. Nel caso di Octave, il dataset di `gretl` viene salvato come matrice singola `X`: una volta importati i dati in Octave, se si vuole, è possibile eliminare la matrice `X`, si veda il manuale di Octave per i dettagli. Per quanto riguarda R, il file dei dati esportati conserva qualsiasi struttura di serie storiche riconoscibile da `gretl`. Le serie sono salvate come strutture individuali: i dati vanno importati in R con il comando `source()`.

Di questi due programmi, R è forse quello di più immediato interesse per gli econometrici, visto che offre più routine statistiche (ad es. modelli lineari generalizzati, stima di massima verosimiglianza, metodi per serie storiche), quindi ho inserito in `gretl` una funzione per trasferire velocemente dati in R. Il menù Sessione di `gretl` contiene il comando `Avvia GNU R`, che salva una versione per R del dataset in uso (come file `Rdata.tmp` nella directory `gretl` dell'utente) e lo carica in una nuova sessione di R. Di seguito sono forniti alcuni dettagli su questa operazione.

Per prima cosa, i dati vengono importati in R scrivendo una versione temporanea di `.Rprofile` nella directory di lavoro corrente (se esiste un file di questo tipo, viene letto da R all'avvio). Se la directory contiene una versione personale di `.Rprofile`, il file originale viene rinominato temporaneamente come `.Rprofile.gretltmp` e viene ripristinato all'uscita da `gretl` (se qualcuno ha un'idea migliore per questa operazione, mi farebbe piacere ascoltarla).

Inoltre, il modo esatto in cui R viene avviato dipende dalla variabile interna `Rcommand` di `gretl`, il cui valore deve essere impostato nel menù File, Preferenze. Il comando predefinito è `RGui.exe` in MS Windows, mentre in X è `R --gui=gnome` se al momento della compilazione è stata trovata un'installazione del desktop Gnome (gnome.org), oppure `xterm -e R` se Gnome non è disponibile. Si noti che questa stringa può contenere al massimo tre elementi separati da spazi: ulteriori elementi saranno ignorati.

Appendice E. Elenco degli URL

Quello che segue è un elenco degli URL citati nel testo.

Census Bureau, Data Extraction Service

<http://www.census.gov/ftp/pub/DES/www/welcome.html>

Estima (RATS)

<http://www.estima.com/>

Home page del desktop Gnome

<http://www.gnome.org/>

Libreria GNU Multiple Precision (GMP)

<http://swox.com/gmp/>

Home page di GNU Octave

<http://www.octave.org/>

Home page di GNU R

<http://www.r-project.org/>

Manuale di GNU R

<http://cran.r-project.org/doc/manuals/R-intro.pdf>

Home page di Gnuplot

<http://www.gnuplot.info/>

Manuale di Gnuplot

<http://ricardo.ecn.wfu.edu/gnuplot.html>

Pagina dei dati di Gretl (versione italiana)

http://gretl.sourceforge.net/gretl_data_it.html

Home page di Gretl (versione italiana)

http://gretl.sourceforge.net/gretl_italiano.html

Home page di GTK+

<http://www.gtk.org/>

Home page del port di GTK+ per win32

<http://www.gimp.org/~tml/gimp/win32/>

Home page di Gtkextra

<http://gtkextra.sourceforge.net/>

Home page di InfoZip

<http://www.info-zip.org/pub/infozip/zlib/>

JRSoftware

<http://www.jrsoftware.org/>

Home page di Mingw (gcc per win32)

<http://www.mingw.org/>

Minpack

<http://www.netlib.org/minpack/>

Penn World Table

<http://pwt.econ.upenn.edu/>

Home page di Readline

<http://cnswww.cns.cwru.edu/~chet/readline/r1top.html>

Manuale di Readline

<http://cnswww.cns.cwru.edu/~chet/readline/readline.html>

Home page di Xmlsoft

<http://xmlsoft.org/>

Bibliografia

- Akaike, H. (1974) "A New Look at the Statistical Model Identification", *IEEE Transactions on Automatic Control*, AC-19, pp. 716-23.
- Baiocchi, G. and Distaso, W. (2003) "GRET: Econometric software for the GNU generation", *Journal of Applied Econometrics*, 18, pp. 105-10.
- Baxter, M. and King, R. G. (1995) "Measuring Business Cycles: Approximate Band-Pass Filters for Economic Time Series", National Bureau of Economic Research, Working Paper No. 5022.
- Belsley, D., Kuh, E. and Welsch, R. (1980) *Regression Diagnostics*, New York: Wiley.
- Berndt, E., Hall, B., Hall, R. and Hausman, J. (1974) "Estimation and Inference in Nonlinear Structural Models", *Annals of Economic and Social Measurement*, 3/4, pp. 653-65.
- Box, G. E. P. and Muller, M. E. (1958) "A Note on the Generation of Random Normal Deviates", *Annals of Mathematical Statistics*, 29, pp. 610-11.
- Davidson, R. and MacKinnon, J. G. (1993) *Estimation and Inference in Econometrics*, New York: Oxford University Press.
- Davidson, R. and MacKinnon, J. G. (2004) *Econometric Theory and Methods*, New York: Oxford University Press.
- Doornik, J. A. and Hansen, H. (1994) "An Omnibus Test for Univariate and Multivariate Normality", working paper, Nuffield College, Oxford.
- Doornik, J. A. (1998) "Approximations to the Asymptotic Distribution of Cointegration Tests", *Journal of Economic Surveys*, 12, pp. 573-93. Reprinted with corrections in M. McAleer and L. Oxley (1999) *Practical Issues in Cointegration Analysis*, Oxford: Blackwell.
- Fiorentini, G., Calzolari, G. and Panattoni, L. (1996) "Analytic Derivatives and the Computation of GARCH Estimates", *Journal of Applied Econometrics*, 11, pp. 399-417.
- Greene, William H. (2000) *Econometric Analysis*, 4th edition, Upper Saddle River, NJ: Prentice-Hall.
- Gujarati, Damodar N. (2003) *Basic Econometrics*, 4th edition, Boston, MA: McGraw-Hill.
- Hamilton, James D. (1994) *Time Series Analysis*, Princeton, NJ: Princeton University Press. (trad. it. *Econometria delle serie storiche*, Monduzzi, Bologna, 1995)
- Johansen, Søren (1995) *Likelihood-Based Inference in Cointegrated Vector Autoregressive Models*, Oxford: Oxford University Press.
- Kiviet, J. F. (1986) "On the Rigour of Some Misspecification Tests for Modelling Dynamic Relationships", *Review of Economic Studies*, 53, pp. 241-261.
- Kwiatkowski, D., Phillips, P.C.B., Schmidt, P. and Shin, Y. (1992) "Testing the Null of Stationarity Against the Alternative of a Unit Root: How Sure Are We That Economic Time Series Have a Unit Root?", *Journal of Econometrics*, 54, pp. 159-178.
- Locke, C. (1976) "A Test for the Composite Hypothesis that a Population has a Gamma Distribution", *Communications in Statistics — Theory and Methods*, A5(4), pp. 351-364.
- MacKinnon, J. G. (1996) "Numerical Distribution Functions for Unit Root and Cointegration Tests", *Journal of Applied Econometrics*, 11, pp. 601-618.
- MacKinnon, J. G. and White, H. (1985) "Some Heteroskedasticity-Consistent Covariance Matrix Estimators with Improved Finite Sample Properties", *Journal of Econometrics*, 29, pp. 305-25.
- Maddala, G. S. (1992) *Introduction to Econometrics*, 2nd edition, Englewood Cliffs, NJ: Prentice-Hall.

- Matsumoto, M. and Nishimura, T. (1998) "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator", *ACM Transactions on Modeling and Computer Simulation*, 8: 1.
- Neter, J. Wasserman, W. and Kutner, M. H. (1990) *Applied Linear Statistical Models*, 3rd edition, Boston, MA: Irwin.
- R Core Development Team (2000) *An Introduction to R*, version 1.1.1.
- Ramanathan, Ramu (2002) *Introductory Econometrics with Applications*, 5th edition, Fort Worth: Harcourt.
- Shapiro, S. and Chen, L. (2001) "Composite Tests for the Gamma Distribution", *Journal of Quality Technology*, 33, pp. 47-59.
- Silverman, B. W. (1986) *Density Estimation for Statistics and Data Analysis*, London: Chapman and Hall.
- Stock, James H. and Watson, Mark W. (2003) *Introduction to Econometrics*, Boston, MA: Addison-Wesley.
- Wooldridge, Jeffrey M. (2002) *Introductory Econometrics, A Modern Approach*, 2nd edition, Mason, Ohio: South-Western.