GRETL (autore: Vittorio Albertoni)

Premessa

GRETL nasce ad opera di Allin Cottrell, della Wake Forest University, grazie ad un precedente lavoro di supporto didattico di Ramachandra (Ramu) Ramanathan, un professore della University of California che associò ad una grande cultura economica una vera e propria passione per la matematica, la statistica e l'informatica.

Il primo rilascio è del 31 gennaio 2000 e da allora il team di sviluppo si è molto allargato, rimanendo sempre capitanato da Allin Cottrell, ben spalleggiato dal nostro connazionale Riccardo "Jack" Lucchetti dell'Università Politecnica delle Marche di Ancona.

Software libero a tutti gli effetti, nato su Linux è rilasciato sotto licenza GNU, come il suo stesso nome suggerisce. GRETL è infatti l'acronimo di GNU Regression, Econometrics and Time-series Library.

Inizialmente concentrato sull'analisi delle serie temporali oggi rappresenta quanto di meglio disponibile per la statistica economica e l'econometria, insieme al blasonato e costoso EViews.

Mentre quest'ultimo è disponibile solo per il sistema operativo Windows, GRETL può essere utilizzato non solo su Windows ma anche su Linux e Mac.

Al centro di tutto l'apparato di strumenti di analisi troviamo la regressione, applicata come autoregressione nell'elaborazione delle serie storiche o come modello lineare ad interpretare relazioni tra variabili, quella che quando parliamo di data science troviamo alla base del così detto machine learning.

In econometria occorrono particolari accorgimenti, e GRETL ce ne offre tanti, per evitare che i modelli regressivi, applicati acriticamente, portino a risultati erronei a causa di problemi tipicamente ricollegabili alla natura di certi dati che ricorrono in campo economico: problemi di eteroschedasticità (i dati di una stessa variabile «risparmio» hanno certamente varianze diverse se la variabile è riferita a popolazione poco abbiente o a popolazione ricca), di multicollinearità (variabili come «reddito» e «risparmio» sono chiaramente correlate tra loro e, se entrano nello stesso modello, occorre evitare che ciò porti distorsioni), ecc.

Mai come in altre discipline matematico-statistiche troviamo un'attenzione agli errori di valutazione ed agli indicatori di attendibilità che troviamo in econometria e GRETL ci aiuta molto in questo.

Nel presente manualetto a portata di dilettante troviamo una semplice introduzione allo strumento, giusto per vedere come funziona e come potersene servire per le cose più banali.

Per gli approfondimenti occorre probabilmente essere un tantino meno dilettanti e, oltre, ovviamente, ad avere una preparazione in scienza economica, bisogna conoscere l'inglese. Esiste infatti parecchia documentazione su GRETL in rete e di particolare pregio è quella che ci propone il software che installiamo.

Indice

1 Installazione

GRETL si trova all'indirizzo *http://gretl.sourceforge.net/*.

Nella home page troviamo un'ampia descrizione del software, che possiamo scorrere anche avvalendoci dell'indice sulla sinistra.

Anche per l'installazione operiamo partendo dalla home page.

Sistema Linux

Nato su Linux, GRETL non fa nulla per essere di facile installazione su Linux e ai poveri linuxiani, da sempre considerati smanettoni, per essere installato offre il codice sorgente in un bell'archivio tar.xz che va compilato: l'ultimo rilasciato si chiama `gretl-2023c.tar.xz` e vi si accede cliccando sulla voce DOWNLOAD nell'indice della home page.

In teoria l'installazione per compilazione dovrebbe avvenire con la solita serie di comandi `./configure`, `make`, `make check` e `make install`, ma subito dopo il primo comando si apre una serie di complicazioni con le dipendenze e non ci si districa più. In genere si finisce col dire «ma chi me lo fa fare!».

Il tutto può essere facilitato ricorrendo allo script `buildgretl.tar.gz` che ci viene proposto nella pagina Download.

Fortunatamente mi pare che GRETL sia presente praticamente in tutti i repository delle varie distro e, in questi casi, è installabile in un soffio attraverso i gestori di programmi.

Così facendo, tuttavia, non potremo mai avere installata l'ultima versione del software, visto che i suoi aggiornamenti sono mediamente più fitti rispetto alle cadenze con cui escono le varie distribuzioni Linux.

Sistema Windows

Nell'indice sulla sinistra della home page clicchiamo su GRETL FOR WINDOWS e apriamo una pagina dalla quale scarichiamo gli installer potendo scegliere tra sistema a 64 bit o a 32 bit.

Il file da scaricare è quello relativo alla latest release, lasciando perdere i current snapshots.

Sistema Mac OS

Nell'indice sulla sinistra della home page clicchiamo su GRETL ON MACOS e apriamo una pagina in cui siamo guidati a scaricare l'installer adatto alla versione del nostro MacOS.

* * *

Con l'installazione, nella nostra home, viene creata una directory denominata `gretl`, destinata ad accogliere tutti i lavori che facciamo con GRETL.

2 Come funziona

Per lanciare GRETL abbiamo a disposizione due comandi:
`gretl`,
`gretlcli`.

Il primo, che corrisponde al lanciatore con icona che si è creato quando abbiamo installato il software, apre GRETL con interfaccia grafica (GUI).

Il secondo apre GRETL a riga di comando su terminale, quello che in Linux e Mac si chiama Terminale e su Windows si chiama Prompt dei comandi.

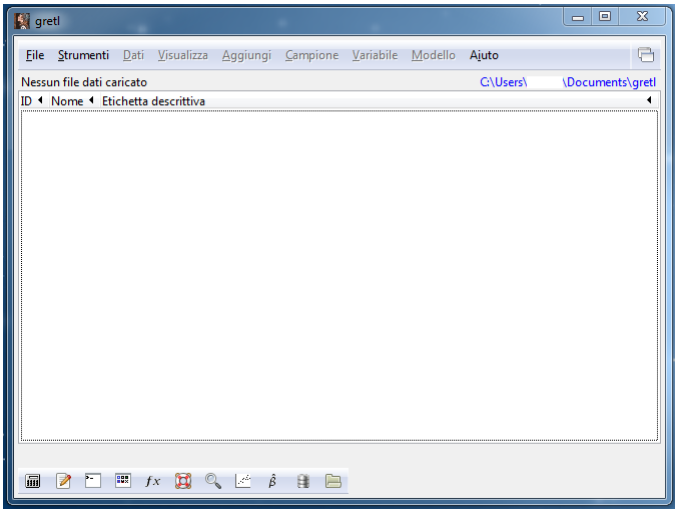
E' possibile aprire GRETL a riga di comando utilizzando l'interfaccia grafica e, in questo caso, realizziamo una comoda integrazione tra ciò che facciamo a terminale e ciò che facciamo con l'interfaccia grafica.

Quando lavoriamo con interfaccia grafica abbiamo a disposizione mouse e menu, quando lavoriamo a riga di comando abbiamo a disposizione comandi da scrivere con la tastiera.

I comandi di GRETL formano un vero e proprio linguaggio di scripting, chiamato HANSL, acronimo ricorsivo di Hansl's A Neat Scripting Language.

Ma andiamo per ordine.

L'interfaccia grafica di GRETL è questa

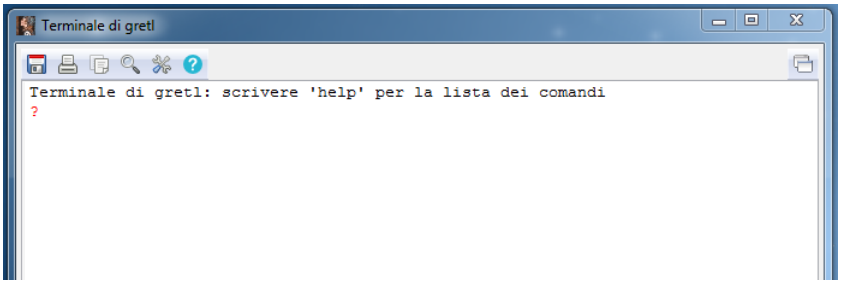


In alto abbiamo una barra dei menu, nella figura non tutti attivati in quanto, come evidenziato nella barra sottostante, nessun dato è stato ancora caricato.

Se apriamo la voce AIUTO troviamo un nutrito elenco di help e manuali su tutto ciò che si può fare con la versione di GRETL su cui siamo, compresa una guida completa all'uso di GRETL, in lingua inglese.

Nella barra inferiore abbiamo una serie di icone. Passando il mouse su ciascuna di esse vediamo a cosa servono.

Cliccando sulla terza da sinistra apriamo GRETL a riga di comando



Senza ancora aver caricato dati abbiamo a disposizione alcuni strumenti, accessibili, appunto, da menu STRUMENTI, tra i quali una simpatica serie dedicata ai test statistici più famosi con possibilità di vederne distribuzione, valori critici e p-value.

Anche qui troviamo la possibilità di aprire il terminale per lavorare a riga di comando, di aprire una finestra per lavorare direttamente con Gnuplot o per lavorare con GNU R.

Gnuplot è il software libero di cui GRETL si avvale per creare grafici. Con gli installatori di GRETL per Windows e Mac viene automaticamente installato anche questo software, mentre su Linux, dove spesso fa parte della dotazione di base del sistema operativo, deve eventualmente essere installato a parte.

GNU R è il potente software libero per elaborazioni statistiche che potremmo utilizzare a potenziamento di ciò che possiamo fare con GRETL. In ogni caso deve essere installato a parte e, per funzionare, deve avere a disposizione dati.

3 Caricamento dei dati

I dati con cui comunemente abbiamo a che fare in econometria possono essere strutturati in tre modi diversi.

Possiamo avere dati sezionali (cross section) quando siamo in presenza della rappresentazione di una o più variabili senza riguardo alle variazioni nel tempo delle osservazioni, esattamente come in un'istantanea.
Ad esempio reddito e consumo delle famiglie italiane per provincia nel 2020.

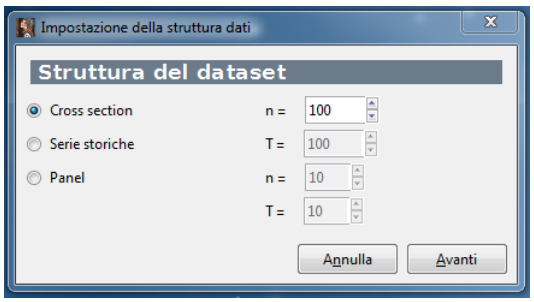
Abbiamo serie storiche quando siamo in presenza di una o più variabili ordinate rispetto al tempo.
Ad esempio reddito e consumo delle famiglie italiane dal 2001 al 2020.

Possiamo, infine, avere dati longitudinali (panel) quando siamo in presenza dell'osservazione di più variabili, ciascuna in una serie di periodi di tempo.
Ad esempio reddito e consumo delle famiglie italiane per provincia dal 2001 al 2020.

GRETL è attrezzato per trattare nel modo migliore queste strutture di dati, fornendoci anche diversi modi di caricarle per essere elaborate.

3.1 Immissione diretta

Da menu FILE ▷ NUOVO DATASET apriamo questa finestrella



nella quale vediamo che ci viene innanzi tutto chiesto di indicare, con riferimento alle tre possibilità viste prima, quale tipo di dataset vogliamo creare e quanti dati debba contenere (n come numero di righe di una tabella nel caso di dati sezionali a cross section, T come numero di periodi di tempo nel caso di serie storiche, entrambi questi elementi nel caso di dati longitudinali a panel).
In questo momento non interessa il numero delle variabili, che sarà gestito nel seguito.

Una volta fatta la scelta e cliccato su Avanti siamo guidati per l'inserimento dei valori della prima variabile lavorando in una sorta di foglio elettronico che ci viene aperto in modo commisurato al numero e al tipo di dati da inserire.

Cliccando sull'icona con il simbolo + di questa sorta di foglio elettronico possiamo inserire i dati per quante altre variabili vogliamo.

3.2 Importazione

I dati da elaborare possono essere importati da altri dataset o da file di lavoro di altri software econometrici (file `.xpt` di SAS e file `.wf1` di EViews).

Gli altri dataset possono essere file `.csv` (dove i separatori ammessi sono la virgola, lo spazio, la tabulazione e il punto e virgola), file Excel (`.xls`) o file di foglio di calcolo `.ods` (prodotti con Open Office e Libre Office).

La prima riga del dataset importato deve contenere i nomi delle variabili (ciascuno identificato con un massimo di 31 caratteri).

Se la prima colonna contiene identificativi di osservazione o date deve avere come intestazione `obs` oppure `date`.

Le date possono essere indicate in anni, trimestri o mesi.
L'anno si indica con quattro cifre.
Il trimestre si indica con le quattro cifre dell'anno, seguite da un punto (.) o da due punti (:) e da un numero di una cifra (1 2 3 o 4).
Il mese si indica con le quattro cifre dell'anno seguite da un punto (.) o da due punti (:) e da un numero di due cifre (01 02 03 ecc. fino a 12).

I file da importare è bene siano disponibili nella cartella `gretl` della nostra home.

L'importazione avviene scegliendo da menu FILE ▷ APRI DATI ▷ FILE UTENTE...

Da dataset esterni possiamo importare solo dati sezionali (cross section) o serie storiche.

Il menu FILE ▷ APRI DATI ci offre anche l'alternativa ▷ FILE DI ESEMPIO... attraverso la quale possiamo caricare dataset che GRETL mette a disposizione a scopo didattico. Tra questi vi sono anche numerosi panel.

L'importazione di dati può essere fatta da terminale con il comando `open` seguito dal nome del file che contiene i dati.

4 Accuratezza numerica

GRETL, nei calcoli, tratta i dati numerici come numeri a virgola mobile in doppia precisione (64 bit).

Quando li scrive utilizza otto cifre, virgola compresa se si tratta di numeri decimali.

Gli interi con più di otto cifre vengono scritti in notazione scientifica.

Se il numero è memorizzato in una variabile, con il comando `printf` possiamo scriverlo utilizzando la direttiva di formattazione `%_._f` in cui, prima del punto si indicano le cifre per la parte intera e dopo il punto si indicano le cifre per la parte decimale.

In questo modo possiamo scrivere quante cifre vogliamo ma quelle precise saranno 16, corrispondenti a quelle della doppia precisione.

Esempi:

Se la variabile x contiene il valore 1/3, questo viene normalmente espresso come 0,333333 (otto cifre, virgola compresa).

Con il comando

```
printf "%1.18f", x
```

il contenuto della variabile x viene espresso come 0,333333333333333370, cioè con le 18 cifre decimali richieste, di cui tuttavia solo le prime 16 precise.

Così, posto che il valore di 345^8 è 20070249011187890625, calcolato in GRETL e inserito nella variabile y viene normalmente espresso come 2,00702e+20.

Con il comando

```
printf "%21.0f", y
```

il contenuto della variabile y viene espresso come 20070249011187937280, cioè le 21 cifre richieste, di cui tuttavia solo le prime 16 precise, con la sedicesima arrotondata.

5 Comandi e funzioni

Per le elaborazioni GRETL si avvale di una nutrita serie di comandi e di funzioni, richiamabili con i loro nomi se lavoriamo a riga di comando oppure attraverso voci di menu se lavoriamo con interfaccia grafica.

L'elenco dei comandi di GRETL è disponibile da menu AIUTO ▷ GUIDA COMANDI in GRETL con GUI oppure scrivendo il comando `help` in GRETL a riga di comando.

Nel primo caso cliccando sul nome di un comando ne otteniamo una esauriente descrizione in italiano con esempi.

Lo stesso avviene nel secondo caso scrivendo a terminale il comando `help` seguito dal nome del comando di cui vogliamo avere la descrizione.

Allo stesso modo l'elenco della funzioni è disponibile da menu AIUTO ▷ GUIDA FUNZIONI in GRETL con GUI oppure scrivendo il comando `help functions` in GRETL a riga di comando.

Anche la descrizione in lingua italiana con esempi si ottiene cliccando sul nome della funzione se lavoriamo con GUI o attraverso il comando `help` seguito dal nome della funzione se lavoriamo a riga di comando.

Lavorando a riga di comando, con i comandi, ma soprattutto con le funzioni, possiamo compiere praticamente qualsiasi tipo di calcolo al di là di quelle che sono le elaborazioni statistiche ed econometriche cui abbiamo accesso lavorando con interfaccia grafica.

Inoltre l'insieme dei comandi e delle funzioni forma il linguaggio di scripting di GRETL, denominato HANSL.

6 Linguaggio HANSL

In GRETL con GUI, da menu AIUTO ▷ INTRODUZIONE AD HANSL, accediamo ad una guida, in lingua inglese, che mette in condizione chiunque abbia una minima infarinatura di programmazione di programmare GRETL utilizzando HANSL.

Per muovere i primi passi, e forse per fare qualche cosa in più, propongo una miniguida in questo paragrafo.

6.1 Interattività e script

Possiamo utilizzare il linguaggio in maniera interattiva lavorando nel terminale di GRETL oppure per predisporre script che hanno il vantaggio di poter essere conservati e riutilizzati quando serve.

Nel primo caso scriviamo le istruzioni a terminale rispettando la sintassi del linguaggio e alla pressione del tasto INVIO le vediamo eseguite.

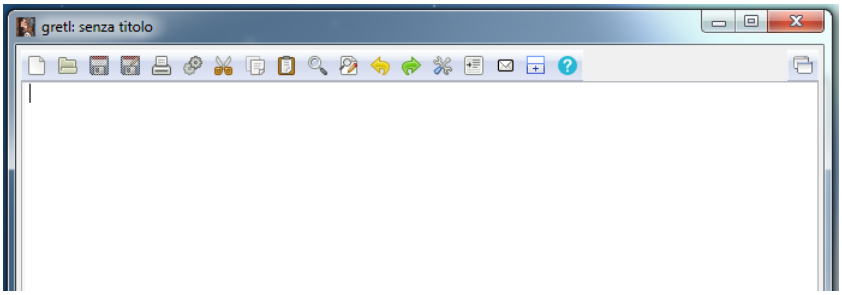
Nel secondo caso scriviamo le istruzioni in un file di testo che memorizziamo con estensione `.inp`, che può essere richiamato con il comando nel generico terminale del sistema operativo

`gretlcli -b` <nome_file>`.inp`

per essere eseguito.

Il file di testo può essere convenientemente scritto utilizzando l'editor di GRETL, che si apre cliccando sulla seconda icona (quella con rappresentata una matita) nella barra inferiore della GUI di GRETL rappresentata a pagina 4.

L'editor si presenta così



Passando il mouse sulle icone della barra degli strumenti otteniamo la descrizione di questi. Di particolare interesse il sesto da sinistra, cliccando sul quale eseguiamo lo script.

6.2 Tipi e variabili

Come in tutti i linguaggi di programmazione possiamo avvalerci di variabili, posizioni di memoria in cui teniamo a disposizione dati soggetti ad elaborazione secondo le istruzioni contenute nello script.

Il linguaggio HANSL è un linguaggio tipizzato, per cui si richiede che le variabili che prevediamo di utilizzare siano dedicate ad uno ed uno solo tipo di dato.

Le variabili possono essere create anche al momento in cui servono, non come avviene in altri linguaggi di programmazione strutturati nei quali si richiede siano preventivamente dichiarate.

L'istruzione per creare una variabile è semplicissima ed è composta dall'indicazione del tipo di dato che essa deve contenere e dal nome con cui identificarla

<tipo> <nome_variabile>

L'assegnazione di un dato alla variabile avviene con l'operatore di assegnamento = secondo la sintassi
<nome_variabile> = <valore>
dove <valore> deve essere del tipo indicato al momento della creazione della variabile vuota.

Al momento della creazione della variabile possiamo inizializzarla, assegnandole subito un valore, con la sintassi
<tipo> <nome_variabile> = <valore>

In questo caso possiamo omettere l'indicazione del tipo in quanto esso viene automaticamente dedotto ed assegnato in funzione del valore inserito, per cui possiamo creare ed inizializzare una variabile con l'istruzione abbreviata
<nome_variabile> = <valore>

6.2.1 Tipi base

I tipi base sono tre.

Stringa

Una stringa (`string`) è una successione di caratteri contenuta tra doppi apici.

Se scriviamo nel terminale di GRETL l'istruzione

```
S = "Vittorio"
```

otteniamo in risposta il messaggio

```
Generata la stringa S
```

che, in realtà, è la conferma della creazione di una variabile, chiamata S, destinata a contenere un dato di tipo string, dato che è attualmente rappresentato dal nome Vittorio.

Scalare

Uno scalare (`scalar`) è un dato numerico singolo.

Se scriviamo nel terminale di GRETL l'istruzione

```
X = 2.75
```

otteniamo in risposta il messaggio

```
Generato lo scalare X = 2,75
```

Matrice

La matrice (`matrix`) è un insieme ordinato di dati numerici.

Può essere ad una sola dimensione, nel qual caso si dovrebbe parlare più propriamente di vettore, o a due dimensioni.

L'insieme ordinato di numeri si ottiene elencando questi tra parentesi graffe e separando con una virgola (,) i numeri appartenenti alla stessa riga e con un punto e virgola (;) le diverse righe.

Sicché una sola successione di numeri separati da virgola genera un vettore riga e una sola successione di numeri separati da punto e virgola genera un vettore colonna.

Nel caso della matrice vera e propria abbiamo l'indicazione di più vettori riga separati da punto e virgola. Affinché la matrice sia coerente occorre che il numero degli elementi di ciascuna riga sia sempre lo stesso. Nel caso il numero degli elementi di ciascuna riga sia pari al numero delle righe abbiamo una matrice quadrata.

`R = {1,2,3}` inserisce nella variabile R, di tipo matrix, il vettore riga

```
1 2 3
```

`C = {1;2;3}` inserisce nella variabile C, di tipo matrix, il vettore colonna

```
1
2
```

```
   3
```
`M = {1,2,3,4;5,6,7,8}` inserisce nella variabile M, di tipo matrix, la matrice
```
   1 2 3 4
   5 6 7 8
```
`MQ = {1,2,3;4,5,6;7,8,9}` inserisce nella variabile MQ, di tipo matrix, la matrice quadrata
```
   1 2 3
   4 5 6
   7 8 9
```

6.2.2 Tipi strutturati

Sono tipi contenitori in quanto sono delle aggregazioni di elementi di un certo tipo.

Array

L'array di GRETL (`array`) può contenere zero o più elementi dello stesso tipo indicizzati con interi consecutivi a partire da 1.

Gli elementi assegnabili ad un array di GRETL devono essere di tipo `string`, o di tipo `matrix`, o di tipo `bundle` (che vedremo subito) o di tipo `list` (che vedremo successivamente).

La creazione di un array che debba ospitare un certo numero di elementi di un certo tipo avviene con la sintassi
<tipo_plurale> <nome_array> `= array(`<numero>`)`
dove <tipo_plurale> è il tipo degli elementi che l'array potrà ospitare al plurale (`strings` per il tipo string, `matrices` per il tipo matrix, `bundles` per il tipo bundle e `lists` per il tipo list) e <numero> è il numero degli elementi che potranno essere inseriti nell'array.

Con l'istruzione
```
strings S = array(4)
```
viene creato l'array denominato S predisposto per contenere 4 elementi di tipo stringa.

L'inserimento degli elementi avviene per assegnazione alla posizione identificata dall'indice
```
S[2] = "Giuseppe"
```
assegna la stringa `"Giuseppe"` come secondo elemento (indice 2) dell'array.

Ovviamente l'inserimento dell'elemento può avvenire anche richiamando la variabile che lo contiene.

Al momento della creazione dell'array possiamo inizializzarlo con la funzione `defarray` con questa sintassi
<nome_array> `= defarray(`<elemento>`,` <elemento>`,....)`
Possiamo evitare di indicare il tipo in quanto esso viene desunto da quello degli elementi inseriti, che deve essere sempre quello.

Bundle

Detto anche array associativo, il bundle (`bundle`) può contenere elementi di qualsiasi tipo previsto da GRETL, anche in modo eterogeneo, ciascuno identificato da una stringa chiave.

Per chi conosce il linguaggio Python siamo praticamente in presenza di quello che là si chiama dizionario. Per chi conosce il linguaggio Perl siamo praticamente in presenza di quello che là si chiama hash.

La creazione di un bundle avviene per semplice dichiarazione, con la sintassi
`bundle` <nome_bundle>

L'inserimento degli elementi avviene per assegnazione ad un indice, con la sintassi
<nome_bundle>`.`<indice> `=` <elemento>.

Esempio:
Con
```
bundle B
```

creiamo il bundle `B` vuoto
Con
`B.a = 12`
vi inseriamo lo scalare `12` identificato dalla chiave `"a"`
Con
`B.b = "Pippo"`
vi inseriamo la stringa `"Pippo"` identificata dalla chiave `"b"`

Al momento della creazione dell'array possiamo inizializzarlo con la funzione `defbundle` con questa sintassi
<nome_bundle> `= defbundle(`<chiave>, <elemento>, <chiave>, <elemento>, `....)`

6.2.3 Tipi legati al dataset

Se abbiamo caricato un dataset secondo quanto abbiamo visto nel Paragrafo 3 possiamo definire variabili derivanti da esso secondo due tipi.

Serie

La serie (series) è un tipo adatto a contenere diverse osservazioni di una stessa grandezza, diverse nello spazio o nel tempo, come il reddito per provincia o il reddito osservato a cadenza annuale in un decennio.

Praticamente si tratta dei dati numerici contenuti in una colonna del dataset, colonna che possiamo richiamare attraverso la stringa che ne costituisce etichetta.

Esempio:
Supponiamo di aver caricato un dataset contenente redditi e consumi in un triennio

	reddito	consumo
2015	198	135
2016	201	136
2017	212	138

Con l'istruzione
`series R = reddito`
creiamo la variabile `R` contenente la serie di valori `198 201 212`.
Con l'istruzione
`C = consumo`
creiamo la variabile `C` contenente la serie di valori `135 136 138`.
La dichiarazione del tipo può essere omessa in quanto GRETL capisce il tipo della variabile dal tipo del dato inserito.

Lista

La lista (`list`) è un array di numeri identificativi di un set di serie ordinate a partire dal numero 1.

La variabile di tipo lista si crea, questa volta obbligatoriamente dichiarandone il tipo, con la sintassi
`list` <nome> `=` <identificativo_serie>
dove <identificativo_serie> può essere indicato attraverso il nome di una variabile `series` precedentemente creata, il numero d'ordine, nel dataset, della colonna o delle colonne da includere nella lista (partendo dal numero 1 a sinistra) oppure la stringa identificativa, nel dataset, della colonna o delle colonne da includere nella lista. L'ordine con cui scriviamo gli identificativi di serie sarà rispettato nella lista e la posizione nella lista viene indicizzata a partire da 1.

Il contenuto della variabile lista si può richiamare con la sintassi
<nome>`[`<indice>`]`

Esempio:

Con riferimento al dataset del precedente esempio, con l'istruzione
`list L = 2 1`
creiamo la variabile di tipo `list L` contenente, nell'ordine, la serie relativa al consumo (la numero 2 del dataset) e la serie relativa al reddito (la numero 1 del dataset).
Allo stesso risultato saremmo pervenuti con l'istruzione
`list L = consumo reddito`
Con l'istruzione
`L[1]`
richiamiamo la prima serie inserita nella lista, cioè quella relativa al consumo.

6.3 Operatori

L'operatore è un simbolo che viene inserito tra due operandi per generare un risultato.

Gli operandi possono essere indicati al momento oppure richiamando variabili che li contengono.

L'espressione contenente operandi ed operatori va indicata in una istruzione di assegnazione se vogliamo inserire il risultato in una variabile oppure come argomento della funzione `eval` se vogliamo scrivere il risultato a terminale.

6.3.1 Operatori aritmetici

Sono previsti operatori aritmetici per ciascun tipo base degli operandi.

Operandi stringa

Il principale operatore per il tipo string è quello per la concatenazione ed è rappresentato dal simbolo della tilde (~)[1].

Esempi:
`NC = "Vittorio" ~ " " ~ "Albertoni"`
crea la variabile `string NC` con assegnata la stringa `"Vittorio Albertoni"`
`eval "Vittorio" ~ " " ~ "Albertoni"`
scrive a terminale `Vittorio Albertoni`

Operandi scalari

Per gli scalari valgono i soliti operatori aritmetici che, nell'ordine di precedenza, sono
`^` per l'elevamento a potenza
`*` per la moltiplicazione
`/` per la divisione
`%` per il modulo (resto della divisione intera)
`+` per la somma
`-` per la sottrazione

Operandi matrici

Per le matrici utilizziamo gli stessi operatori previsti per gli scalari, salvo l'operatore `%`.

Se l'elevamento a potenza, la moltiplicazione o la divisione si vogliono effettuare membro a membro occorre far precedere l'operatore `^`, `*` o `/` da un punto (`.`).

Esempio:
`eval {1,2;5,3} * {4,2;2,3}` ritorna
```
 8   8
26  19
```

[1] Il simbolo della Tilde si ottiene con AltGr+ì sulla tastiera Linux, Alt+5 o Alt+N sulla tastiera Mac o Alt+126 del tastierino numerico sulla tastiera Windows.

`eval {1,2;5,3} .* {4,2;2,3}` ritorna

```
    4  4
   10  9
```

6.3.2 Operatori di confronto

Servono per confrontare entità di tipo scalare e restituiscono un valore booleano (1 = vero, 0 = falso).
`==` uguale
`!=` non uguale
`<` minore
`>` maggiore
`<=` minore o uguale
`>=` maggiore o uguale

6.3.3 Operatori logici

Servono per la verifica di più condizioni.
`&&` AND logico
`||` OR logico
`!` NOT logico

6.4 Interattività con l'utente

Per l'interattività con l'utente non sono previste, come in altri linguaggi, istruzioni per la richiesta di input guidato da tastiera.

Sono invece previste istruzioni per l'output sullo schermo a terminale.

eval

Abbiamo già introdotto nel precedente paragrafo l'istruzione `eval`. Essa serve per scrivere a terminale il risultato di una espressione nella quale intervengano operatori.
`eval 5*3-12` scrive `33`
`eval 3^3` scrive `27`
`eval 5>6` scrive `0`

print

Con questa istruzione possiamo scrivere a terminale il contenuto di una variabile o una stringa indicata al volo.
`print "Ciao"` scrive `Ciao`
`print X` scrive il contenuto della variabile `X`

printf

Con questa istruzione possiamo scrivere a terminale una frase composta con elementi eterogenei inserendo nel testo anche valori di variabili, magari nemmeno noti al momento della stesura dello script, attraverso l'uso di segnaposti.

I segnaposti utilizzati sono
`%s` per valori di tipo stringa,
`%g` per valori di tipo numerico.

Come abbiamo già visto nel paragrafo 4, possiamo anche utilizzare direttive di formattazione per formattare dati numerici con il segnaposto
`%_._f`

dove prima del punto si indicano eventualmente le posizioni in cui allineare la parte intera del numero e dopo il punto si indica il numero delle cifre decimali da indicare, arrotondando l'ultima.

Esempio:
Data la variabile `nome`, contenente la stringa `"Luigi"` e la variabile `Y`, che nel corso dello script assumerà il valore 72,65687,
`printf "Bravo %s, hai vinto %.2f euro\n", nome, Y`
scrive
`Bravo Luigi, hai vinto 72,66 euro`

6.5 Strutture di controllo

Sono comandi per condizionare l'esecuzione di certe istruzioni al verificarsi di determinate condizioni o per l'esecuzione ripetuta di certe istruzioni.

6.5.1 Esecuzione condizionale

Il costrutto più semplice per l'esecuzione condizionale è
`if` <condizione>
    <istruzioni>
`endif`

Se la condizione è vera vengono eseguite le <istruzioni> e poi si prosegue con il resto del programma. Se la condizione è falsa non vengono eseguite le <istruzioni> e si prosegue con il resto del programma.

Nel costrutto possiamo indicare istruzioni da eseguire nel caso la prima condizione non sia vera con la parola chiave `else`
`if` <condizione>
    <istruzioni>
`else`
    <istruzioni>
`endif`

Possiamo infine gestire il verificarsi di più condizioni alternative con
`if` <condizione>
    <istruzioni>
`elif` <condizione>
    <istruzioni>
`...`
`...`
`else`
    <istruzioni>
`endif`

6.5.2 Ripetizione

Per eseguire più volte le stesse istruzioni abbiamo la parola chiave `loop` con la seguente sintassi
`loop` <espressione_di_controllo>
    <istruzioni>
`endloop`

L'espressione di controllo più comune è quella che indica il numero di volte che devono essere eseguite le istruzioni con un intero
```
loop 5
print "Ciao"
endloop
```
scrive cinque volte la parola `Ciao`.

Altre espressioni di controllo utilizzano le parole chiave `while`, `for`, `foreach` e per queste rimando al manuale.

6.6 Creazione di funzioni

Nonostante GRETL offra una nutrita serie di funzioni, al punto che si fatica a dire che cosa manchi, offre anche all'utente la possibilità di creare lui stesso funzioni che vadano ad arricchire la raccolta.

La creazione di una funzione avviene con la seguente sintassi

`function` <tipo> <nome> (<tipo> <parametro>, ...)
<istruzioni>
`end function`

dove

<tipo> è il tipo del valore restituito dalla funzione ed è uno di quelli visti nel paragrafo 6.2,
<nome> è il nome identificativo della funzione, che serve per richiamarla,
<parametri> sono i dati da indicare alla funzione affinché essa possa produrre il risultato richiesto,
<istruzioni> rappresentano ciò che deve fare la funzione per produrre il risultato. Alla fine delle istruzioni occorre indicare quale sia il risultato da restituire con la parola chiave `return`.

La funzione creata viene richiamata scrivendone il nome e, tra parentesi tonde, il valore dei parametri.

Possiamo salvare in un file con estensione `.inp` il testo della creazione della funzione e ogni volta che ci serve di utilizzare la funzione importarla, prima di richiamarla, con
`include` <nome_file>`.inp`

Esempi:

Poniamo, più per semplicità di esempio che per necessità, che ci sia comodo disporre di una funzione per raddoppiare un numero.
La possiamo produrre in questo modo:

```
function scalar raddoppia (scalar n)
doppio = n * 2
return doppio
end function
```

o, più semplicemente, in quest'altro modo abbreviato

```
function scalar raddoppia (scalar n)
return n * 2
end function
```

Dopo aver creato la funzione, o averla importata con `include`, con

```
eval raddoppia(6)
```

otteniamo e scriviamo il risultato `12`.

Un esempio meno banale, utile soprattutto per dimostrare come GRETL gestisca la ricorsività, quello di una funzione per il calcolo del fattoriale di un numero:

```
function scalar fattoriale (scalar n)
    if n == 0
       return 1
    else
       return n * fattoriale(n-1)
    endif
end function
```

Ho utilizzato le indentazioni per rendere più leggibile il codice. Si tratta di una cosa non necessaria.

Questo esempio serve anche per spiegare come mai GRETL non preveda una funzione precostituita per calcolare il fattoriale: perché sarebbe una funzione che genererebbe spesso risultati sbagliati.

Come abbiamo visto nel paragrafo 4, infatti, GRETL tratta tutti i dati numerici, interi compresi, come numeri a virgola mobile in doppia precisione.
Ciò fa si che grandi interi, che in altri linguaggi di programmazione come C, C++ e Pascal, genererebbero stack overflow, in GRETL vengono comunque trattati in notazione decimale a virgola mobile ma sono esatti solo fino alla sedicesima cifra compresa.

Per un software che tratta di econometria, dove i numeri possono essere sì molto grandi, ma vengono sempre trattati in scala ridotta (in milioni, o addirittura in miliardi), la precisione a 16 cifre è più che sufficiente, ma non è sufficiente per tutti i calcoli, come quello per il fattoriale.

Un esempio per chiarire.
Il fattoriale di 25 è 15511210043330985984000000 e lo possiamo calcolare esattamente con un linguaggio come Python o Julia, dove gli interi sono o possono essere trattati in precisione arbitraria, cioè praticamente senza limiti, se non quelli imposti dall'hardware.
Se lo calcoliamo in GRETL utilizzando la funzione costruita sopra, con stampa formattata

```
printf "%30.0f\n", fattoriale(25)
```

otteniamo il risultato 15511210043330990350270464.

Sempre un bell'intero di 26 cifre ma le cifre esatte sono solo le prime 16, con la sedicesima arrotondata.

7 Grafica

Come accennato alla fine del Paragrafo 2, GRETL si appoggia al software esterno Gnuplot per illustrare graficamente i dati in suo possesso.

Abbiamo innanzi tutto la possibilità di accedere direttamente a Gnuplot da GRETL per costruire un grafico qualsiasi, indipendentemente dai dati in possesso di GRETL, per esempio per creare il grafico di una qualsiasi funzione matematica.
Dall'interfaccia grafica, agendo con STRUMENTI ▷ GNUPLOT possiamo aprire un terminale Gnuplot in cui lavorare con il linguaggio che Gnuplot capisce oppure, agendo su STRUMENTI ▷ DISEGNA UNA CURVA possiamo lavorare con Gnuplot guidati da un interprete grafico e senza conoscere il linguaggio di Gnuplot.

Se oggetto della grafica sono dati caricati su GRETL possiamo lavorare dal terminale di GRETL con il comando `gnuplot`, che ha la seguente sintassi

`gnuplot` <variabili_y> <variabile_x> <opzioni>

dove le <variabili_y> sono la variabile o le variabili dipendenti riferite sull'asse delle ordinate e la <variabile_x> è la variabile assunta come variabile indipendente e riferita sull'asse delle ascisse.
Le <opzioni>, per la cui illustrazione rimando alla Guida comandi accessibile dalla GUI di GRETL, possono servire per indicare le caratteristiche del grafico.
A grafico creato, comunque, cliccando destro su di esso apriamo un menu grazie al quale, cliccando su MODIFICA, abbiamo la possibilità di intervenire sull'estetica del grafico, sul suo titolo, sui titoli e le etichette degli assi, ecc.

La creazione dei grafici diventa più semplice utilizzando dalla GUI di GRETL il menu VISUALIZZA ▷ GRAFICO o VISUALIZZA ▷ GRAFICI MULTIPLI.

8 Esempi di calcoli ed elaborazioni

Quanto indicato nei precedenti paragrafi, sorretto dalla consultazione delle Guide ai comandi e alle funzioni, ci mette in grado di utilizzare GRETL per fare moltissime cose, da calcoli più o meno banali alle più sofisticate elaborazioni statistiche o econometriche. Ovviamente, per queste ultime, essendo dotati della preparazione necessaria nelle specifiche materie.

8.1 Calcoli generici

Comprendo in questa categoria tutti i calcoli aritmetici, matematici e di algebra lineare per i quali non esistono voci di menu nell'interfaccia grafica e che possono essere eseguiti ricorrendo necessariamente al terminale o allo scripting.

Come avviene per tutti i linguaggi di scripting dotati di una shell (penso soprattutto a quelli votati al calcolo come Python o Julia) GRETL può innanzi tutto essere una formidabile calcolatrice, oltre tutto facile da usare grazie alla semplicità della sintassi dei suoi comandi e delle sue funzioni.

Rammento che, per far comparire a terminale i risultati delle elaborazioni, i comandi e le funzioni utilizzate vanno preceduti dal comando `eval`. Cosa non necessaria negli script quando questi risultati vengono assegnati a variabili.

Utilizzando gli operatori aritmetici eseguiamo i più semplici calcoli:

`eval 3.5 * 4` ritorna `14`

`eval 4 ^ 2` ritorna `16`

`eval 27 ^ (1/3)` ritorna `3`

`eval 3 * 2 + 4` ritorna `10`

`eval 14 / 3` ritorna `4,6666667`

Utilizzando anche le numerose funzioni che abbiamo a disposizione possiamo divertirci.

`eval cos($pi)` utilizza le funzioni `cos` e $pi e ritorna `-1`

`eval sin($pi/2) + 2` utilizza analoghe funzioni e alcuni operatori e ritorna `3`

`eval log(5)` oppure `eval ln(5)` ritorna `1,6094379`

`eval log10(5)` ritorna `0,69897`

`eval det({3, 2; 4, 1})` ritorna `-5`

```
eval inv({2, -1, 0; 0, 3, 2; 5, 0, 3}) * {2; 16; 21}
```

risolve il sistema di equazioni lineari

$$\begin{cases} 2x_1 - x_2 = 2 \\ 3x_2 + 2x_3 = 16 \\ 5x_1 + 3x_3 = 21 \end{cases}$$

ritornando il vettore colonna delle soluzioni

```
3
4
2
```

secondo il noto procedimento

$X = M^{-1}N$

dove X è il vettore delle soluzioni, M è la matrice dei coefficienti delle incognite (M^{-1} ne è l'inversa) e N è il vettore colonna dei termini noti.

```
eval polroots({-2, 3})
```

risolve l'equazione

$3x - 2 = 0$

ritornando la soluzione *0,66667*

L'argomento della funzione è il vettore riga dei coefficienti delle potenze decrescenti del polinomio ordinati alla rovescia.

```
eval polroots({-5, 3, 4})
```

risolve l'equazione

$4x^2 + 3x - 5 = 0$

ritornando il vettore colonna delle uniche due radici reali

```
0,80425
-1,5542
```

```
eval polroots({-2, 0, 3, 0, 5})
```

risolve l'equazione

$5x^4 + 3x^2 - 2 = 0$

(notare l'inserimento del valore 0 come coefficiente delle potenze mancanti)

ritornando la matrice delle radici

```
-0,63246 0,0000
 0,63246 0,0000
 0,0000 -1,0000
 0,0000  1,0000
```

nella quale la parte reale delle radici è nella prima colonna e nella seconda c'è la parte immaginaria. Dove la parte immaginaria è 0 siamo in presenza di radici reali.

`eval easterday(2022)`

ritorna lo scalare `4.17`

che indica la data della Santa Pasqua dell'anno in cui scrivo: 17 aprile

(lo scalare ritornato è costituito da mese + giorno/100, per cui la parte intera dello scalare indica il mese e la parte decimale il giorno, facendo attenzione che il giorno 1 viene indicato come 01 e 10 viene indicato come 1, così come il giorno 2 viene indicato come 02 e 20 viene indicato come 2).

Per esempio `eval easterday(2025)` ritorna `4.2` che significa 20 aprile.

8.2 Calcoli statistici

Anche per i calcoli statistici abbiamo a disposizione una nutrita serie di funzioni utilizzabili da terminale, alcune utilizzabili in maniera più comoda da interfaccia grafica.

Gli argomenti da passare alle funzioni statistiche possono essere passati prendendoli dal dataset caricato nei modi visti nel Paragrafo 3.

Prendiamo, per esempio, la più banale funzione statistica `mean`(), che calcola la media avendo come argomento un vettore o una serie.

Con

`eval mean({3, 4, 5})`

calcoliamo la media (`4`) dei tre valori contenuti nel vettore passato come argomento.
Il vettore passato nell'esempio è un vettore riga ma sarebbe lo stesso con il vettore colonna `{3; 4; 5}`.

Se abbiamo caricato il piccolo dataset contenente redditi e consumi in un triennio, che abbiamo già utilizzato per parlare delle serie,

	reddito	consumo
2015	198	135
2016	201	136
2017	212	138

possiamo creare la serie

`series R = reddito`

e poi calcolare la media del reddito con

`eval mean(R)`

oppure, in maniera ancor più diretta, passando come argomento alla funzione `mean()` il titolo della colonna del dataset di cui vogliamo calcolare la media con

`eval mean(reddito)`

in entrambi i casi ottenendo il risultato `203,83333`.

Ciò vale per le altre funzioni statistiche più ricorrenti `sd()` (per lo scarto quadratico medio), `var()` (per la varianza), `median()` (per la mediana), `min()` (per il valore minimo), `max()` (per il valore massimo), ecc.

Utilizzando l'interfaccia grafica possiamo ottenere statistiche sui dati del dataset caricato agendo sul menu VISUALIZZA ▷ STATISTICHE DESCRITTIVE e scegliendo nella successiva finestra di dialogo la serie per la quale vogliamo visualizzare le statistiche, ulteriormente potendo scegliere tra SOLO STATISTICHE PRINCIPALI (media, minimo, massimo e scarto quadratico medio) e STATISTICHE DESCRITTIVE COMPLETE (media, mediana, minimo, massimo, scarto quadratico medio, coefficiente di variazione, asimmetria, curtosi).

Altra importante funzione statistica è `corr()`, che restituisce l'indice di correlazione di Pearson. Essa funziona esattamente come quelle viste prima, avendo ovviamente bisogno di due argomenti, corrispondenti ai due vettori o alle due serie in riferimento alle quali si vuole valutare l'esistenza di una correlazione lineare

`eval corr({3,4,5}, {4,7,9})` restituisce `0,99339927`

`eval corr(reddito,consumo)` restituisce `0,98745886`

Da interfaccia grafica possiamo ottenere quest'ultimo risultato agendo sul menu VISUALIZZA ▷ MATRICE DI CORRELAZIONE e scegliendo nella successiva finestra di dialogo le serie da sottoporre a verifica.

Altro importante blocco di funzioni è quello che GRETL dedica alla statistica inferenziale.

Se lavoriamo su terminale a riga di comando abbiamo a disposizione, tra le altre, le funzioni

`cdf()` per calcolare funzioni di ripartizione,

`critical()` per calcolare i valori critici,

`pvalue()` per calcolare il p-value.

Il primo argomento da indicare tra le parentesi tonde è un carattere che identifica la distribuzione di riferimento (ad esempio `n` per la distribuzione normale, `t` per la distribuzione del test t di Student, `f` per la distribuzione del test F di Snedecor, ecc.).

Per la completa descrizione di queste funzioni e degli argomenti che esse richiedono rimando alle chiarissime indicazioni in lingua italiana accessibili da menu AIUTO ▷ GUIDA FUNZIONI ▷ PROBABILITÀ.

Se lavoriamo su interfaccia grafica abbiamo sicuramente un più agevole accesso a queste funzioni agendo sul menu STRUMENTI e poi su TAVOLE STATISTICHE, CALCOLA P-VALUE, GRAFICI DI DISTRIBUZIONE e CALCOLA TEST.

Per i valori critici scegliamo il menu STRUMENTI ▷ TAVOLE STATISTICHE, poi selezioniamo la scheda relativa alla distribuzione che ci interessa e inseriamo i richiesti valori.

Per la funzione di ripartizione e il p-value scegliamo da menu STRUMENTI ▷ CALCOLA P-VALUE, poi selezioniamo la scheda relativa alla distribuzione che ci interessa e inseriamo i richiesti valori.

Da menu STRUMENTI ▷ GRAFICI DI DISTRIBUZIONE abbiamo modo di ottenere il grafico della distribuzione che ci interessa.

Sempre lavorando su interfaccia grafica, da menu STRUMENTI ▷ CALCOLA TEST abbiamo la possibilità di calcolare i test statistici di più ricorrente uso.

Per un esempio di applicazione di GRETL alla statistica inferenziale propongo questo esercizio.

Per verificare l'efficacia di due sistemi di concimazione dei pomodori sono state piantate 9 piantine della stessa qualità, a gruppi di 3, in tre aiuole a distanza di 4 metri. La prima aiuola e le relative piante sono state trattate con un concime in polvere solubile da applicare ogni settimana, con un certo dosaggio, in soluzione di 10 litri nel terreno e in soluzione da 1 litro a spruzzo sulle foglie (identifichiamo con A questo trattamento). La seconda aiuola è stata trattata con un concime granulare distribuito in un certo dosaggio quindicinalmente sul terreno (identifichiamo con B questo trattamento). Per la terza aiuola non è stato applicato alcun trattamento. In questo modo ciascun gruppo di piante ha avuto tutti gli altri trattamenti di irrigazione, di diserbaggio, ecc. è cresciuto su un terreno omogeneo dal punto di vista della composizione ed è stato esposto agli stessi agenti atmosferici di insolazione, esposizione alla pioggia, ecc. Unica differenza quella del trattamento di concimazione. I primi 6 pomodori venuti a maturazione in ciascun gruppo di piante sono stati pesati ottenendo i seguenti risultati in grammi, inseriti in un file `pomodori.csv`

concime_A	concime_B	no_concime
58	49	41
54	55	53
58	59	39
48	63	53
67	58	41
69	43	46

La nostra verifica consiste nello stabilire se la differenza di peso riscontrata tra i pomodori provenienti dalle varie aiuole possa essere dovuta ad efficacia della concimazione o meno.

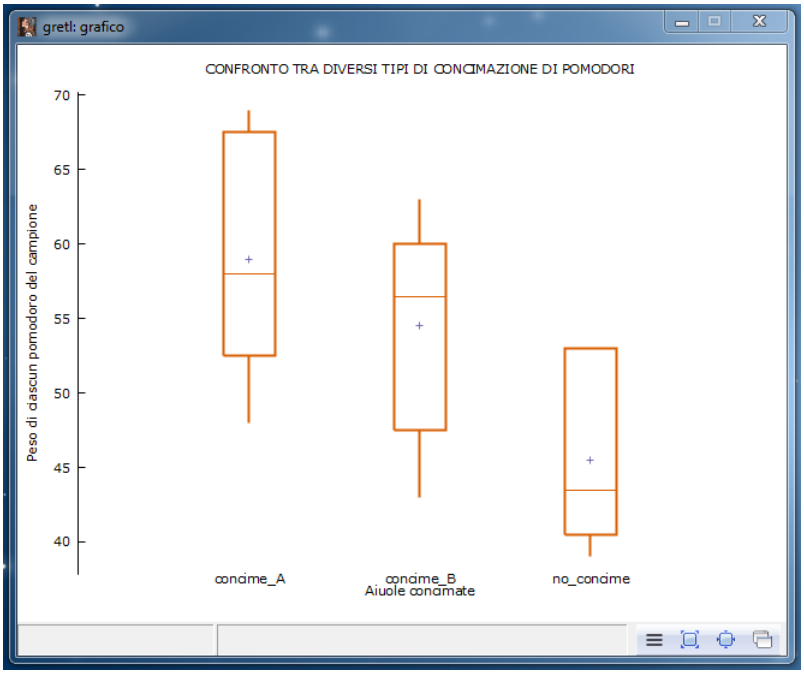
Prima cosa da fare è caricare il dataset e, lanciato GRETL, lo facciamo da menu FILE ▷ APRI DATI ▷ FILE UTENTE...

Ora sintetizziamo i dati calcolandone medie e varianze e lo facciamo da terminale con questi comandi

`eval mean(concime_A)`, che ritorna `59`, media del campione concimato A,
`eval var(concime_A)`, che ritorna `62,4`, varianza del campione concimato A,
`eval mean(concime_B)`, che ritorna `54,5`, media del campione concimato B,
`eval var(concime_B)`, che ritorna `53,5`, varianza del campione concimato B,
`eval mean(no_concime)`, che ritorna `45,5`, media del campione non concimato,
`eval var(no_concime)`, che ritorna `45,5`, varianza del campione non concimato.[2].

Esaminando a occhio questi dati siamo portati a concludere che entrambe le concimazioni, ma specialmente quella denominata A, hanno avuto effetto.

Ciò sembrerebbe confermato anche dall'esame di questo grafico, che creiamo da menu VISUALIZZA ▷ GRAFICO ▷ BOXPLOT..., un tipo di grafico particolarmente adatto al nostro dataset



I box creati, uno per ciascun campione, si estendono dal primo quintile al terzo quintile, il simbolo + all'interno di ciascun box indica la media dei dati e la linea orizzontale all'interno di ciascun box indica la mediana.
Le linee che vediamo sopra ciascun box si estendono fino al valore massimo dei dati del campione rappresentato dal box e le linee che vediamo sotto ciascun box si estendono fino al valore minimo.

Ma per evitare i giudizi a occhio esistono i test statistici.

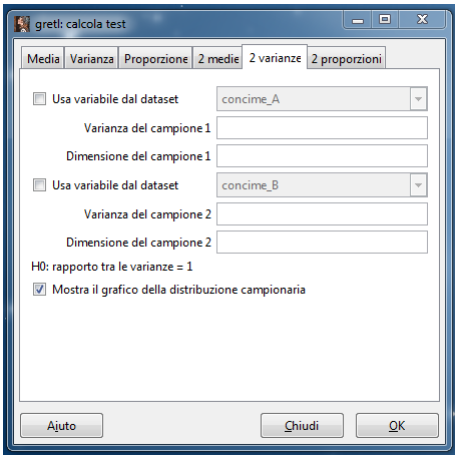
[2]Notare come GRETL, sapendo che siamo su un computer italiano, ci ritorni i dati con la virgola come separatore decimale. Non illudiamoci: quando inseriamo noi dati per lui, dobbiamo usare come separatore decimale il punto.

Il test determinante è quello per verificare se la differenza tra le medie rilevate nei campioni è significativa o può semplicemente essere dovuta al caso. Per questa valutazione sappiamo che esiste il test t di Student.

Ma prioritariamente è bene sottoporre a test per la significatività della differenza la varianza. E questa valutazione la possiamo fare con il test F di Snedecor.

Cominciamo da quest'ultimo.

Da menu STRUMENTI ▷ CALCOLA TEST apriamo una finestra di dialogo attraverso la quale possiamo accedere a più schede che ci guidano al calcolo di test statistici. Per la nostra attuale esigenza apriamo la scheda intitolata 2 VARIANZE, che si presenta così

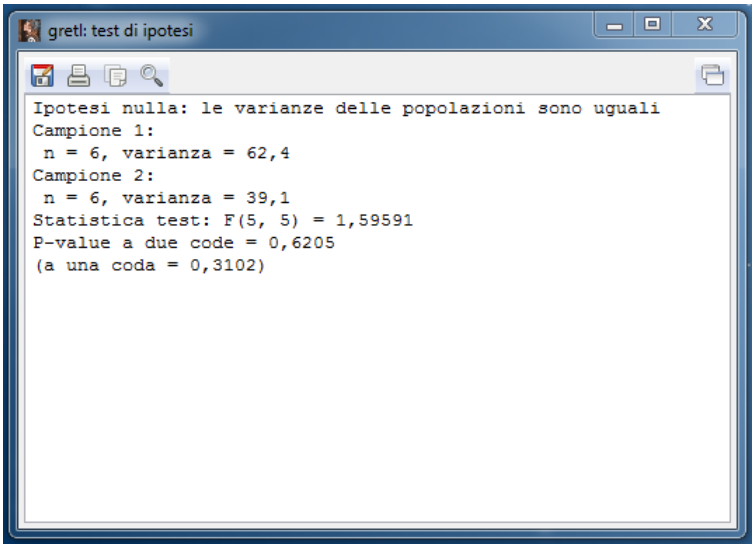


Eseguiamo la prima verifica confrontando la varianza del campione concimato A con quella del campione non concimato.

Pertanto selezioniamo il primo USA VARIABILE DAL DATASET e lasciamo la scelta CONCIME_A e poi selezioniamo il secondo USA VARIABILE DAL DATASET e poi scorriamo sulla scelta NO_CONCIME.

Lasciamo selezionata, o selezioniamo, l'opzione MOSTRA IL GRAFICO DELLA DISTRIBUZIONE CAMPIONARIA.

Cliccando su OK otteniamo il risultato del confronto in questa finestrella



nella quale viene visualizzato il valore del test F di Snedecor (il cui calcolo, per la verità, non aveva bisogno di tutto questo apparato, in quanto il valore del test è semplicemente il rapporto tra le due varianze messe a confronto) arricchito del relativo p-value, il cui valore è alquanto elevato e ci porta ad accettare l'ipotesi nulla, cioè a ritenere che la differenza tra le due varianze sia accidentale e statisticamente non rilevante.

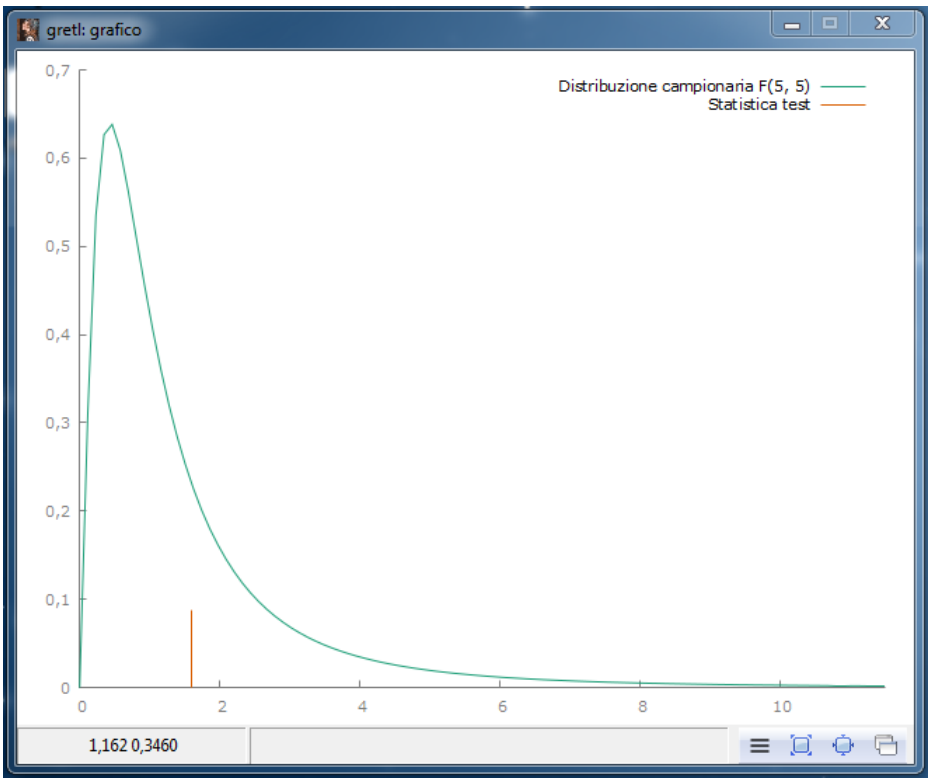
Possiamo averne conferma se calcoliamo il valore critico del test F in presenza dei gradi di libertà riferiti ai nostri piccoli campioni.

Lo potremmo fare da menu STRUMENTI ▷ TAVOLE STATISTICHE selezionando la scheda F in cui inserire i dati richiesti, ma lo possiamo anche fare più semplicemente a terminale con il comando

```
eval critical(f, 5, 5, 0.025)
```

dove il primo parametro indica la scelta del test F, i successivi due parametri sono i gradi di libertà dei due campioni e l'ultimo parametro è la probabilità a una coda per ottenere una probabilità a due code di 0.05, cioè un risultato attendibile al 95%.
Il valore critico che otteniamo è `7,1463818`, di molto superiore al valore del test, a conferma del fatto che va accettata l'ipotesi nulla.

Tutto ciò è visivamente desumibile dal grafico della nostra distribuzione campionaria

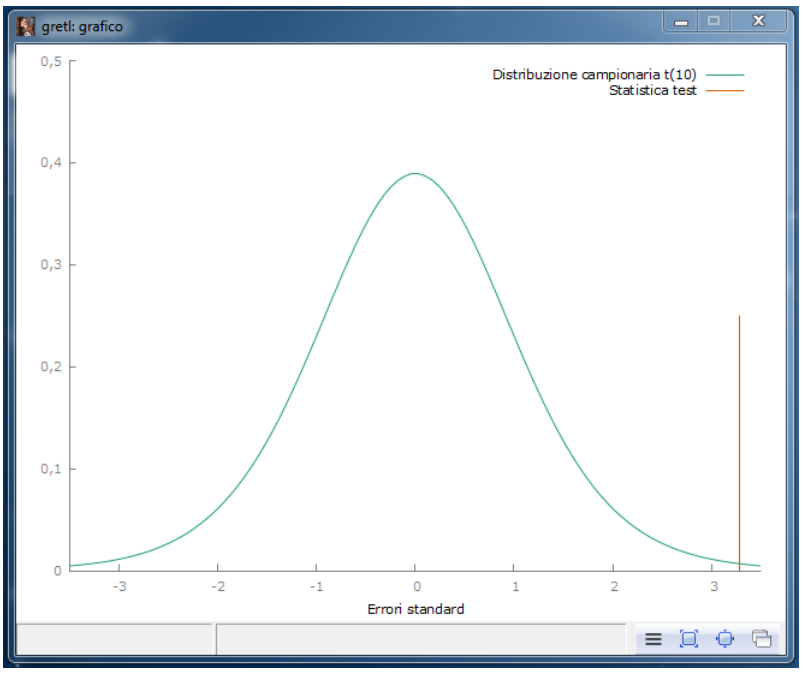


nel quale vediamo come il nostro test F del valore di 1,59591, indicato dalla barra rossa verticale, sia sicuramente localizzato nell'area di accettazione dell'ipotesi nulla.

Alle stesse conclusioni arriviamo se ripetiamo il confronto tra la varianza del campione concimato B e quella del campione non concimato.

Ora possiamo affrontare il decisivo confronto tra le medie.

Da menu STRUMENTI ▷ CALCOLA TEST apriamo la finestra di dialogo da cui, questa volta, apriamo la scheda intitolata 2 MEDIE e, con lo stesso procedimento con cui abbiamo prima instaurato il confronto tra varianze, instauriamo il confronto tra la media del campione concimato A con la media del campione non concimato.
Nella finestra, visti i risultati dei confronti tra le varianze, selezioniamo l'opzione ASSUMI CHE LO SCARTO QUADRATICO MEDIO SIA COMUNE TRA LE POPOLAZIONI.
Il valore del test t risulta essere 3,28229, contro un valore critico 2,22814 con probabilità a due code 0.05 (attendibilità al 95%) e un valore critico 3,16927 con probabilità a due code 0.01 (attendibilità al 99%).
Dal momento che il valore del test supera in entrambi i casi il valore critico dobbiamo concludere con il respingere l'ipotesi nulla e ammettere che la differenza tra le medie è statisticamente significativa e non può essere attribuita al caso.
Il che significa che il trattamento di concimazione A ha sicuramente avuto effetto.
Lo possiamo vedere chiaramente anche dal grafico della distribuzione del test t campionario



nel quale vediamo come il nostro test t del valore di 3,28229, indicato dalla barra rossa verticale, sia sicuramente localizzato fuori dall'area di accettazione dell'ipotesi nulla.

Con lo stesso procedimento possiamo instaurare il confronto tra la media del campione concimato B con la media del campione non concimato.
Questa volta il test t risulta essere del valore di 2,29093, sempre contro il valore critico 2,22814 con probabilità a due code 0.05 (attendibilità al 95%) e il valore critico 3,16927 con probabilità a due code 0.01 (attendibilità al 99%).
Il valore del test supera di pochissimo il valore critico con probabilità 0.05 e non supera quello con probabilità 0.01.
Possiamo pertanto scartare con molta cautela l'ipotesi nulla e non è possibile ammettere con la stessa sicurezza di prima che anche il tipo di concimazione B abbia avuto effetto.

Certo non ci ha aiutati in questa indagine il ridottissimo numero degli elementi dei campioni: se avessimo pesato qualche pomodoro in più avremmo probabilmente ottenuto, anche nel secondo raffronto, un risultato meno incerto.

8.3 Calcoli econometrici

Per le elaborazioni di econometria abbiamo a disposizione molti comandi eseguibili da terminale ma è molto più comodo lavorare da interfaccia grafica, dove siamo meglio guidati nella scelta delle varie opzioni con cui impostare le elaborazioni e possiamo con più facilità passare alle rappresentazioni grafiche di ciò che stiamo facendo.

I comandi a disposizione per queste elaborazioni sono ben descritti nella Guida comandi, nella sezione Stima.

Nell'interfaccia grafica la voce di menu dove troviamo gli strumenti econometrici è MODELLO. Una volta scelto un modello si apre una finestra di dialogo da cui avviare l'elaborazione, finestra che ha sempre un pulsante AIUTO, cliccando sul quale abbiamo accesso ad una guida sempre molto ben fatta e in lingua italiana.

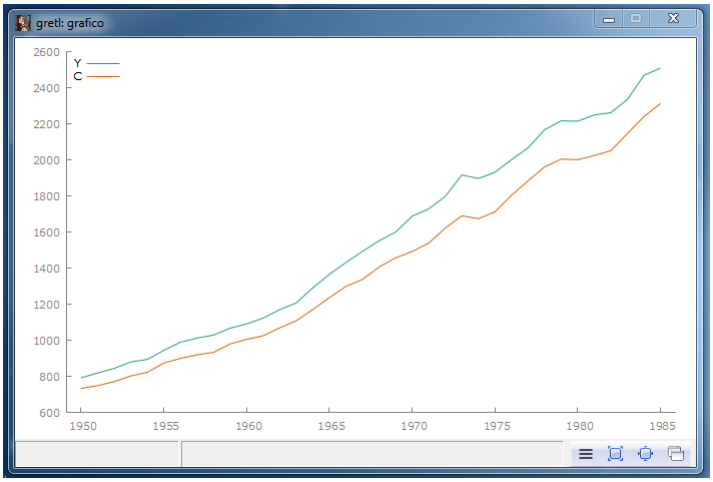
Per una esemplificazione di ciò che possiamo fare ho scelto di lavorare con un dataset che troviamo tra i file che fanno parte della dotazione di GRETL. Si tratta della serie storica di reddito e consumo USA tra il 1950 e il 1985.

Carichiamo il dataset da menu FILE ▷ APRI DATI ▷ FILE DI ESEMPIO... andando a scegliere, nella scheda Greene, il file GREENE11_3 AGGREGATE CONSUMPTION AND INCOME DATA.

Una volta caricato il file vediamo che si compone di due serie, una denominata Y, che rappresenta il reddito (aggregate income), l'altra denominata C, che rappresenta il consumo (aggregate consumption).

Possiamo vedere i dati selezionando entrambe le serie nella finestra di lavoro di GRETL e poi da menu DATI ▷ MOSTRA VALORI.

Possiamo vederne il grafico da menu VISUALIZZA ▷ GRAFICO ▷ SERIE STORICA, selezionando entrambe le variabili nella finestra di dialogo che si apre e dando OK. Il risultato è questo



Abbiamo l'andamento della variabile C in rosso (consumo) che segue come una fedele ombra la variabile R in azzurro (reddito).

Poniamo ora di essere all'inizio dell'anno 1986 e di voler fare una previsione sullo sviluppo dei consumi nel prossimo triennio.

Dal momento che, in economia, ciò che succede ad una variabile al tempo t, in assenza di elementi perturbatori o modificatori, è sempre fortemente influenzato da ciò che è successo al tempo $t-1$, la via più rapida potrebbe essere quella di affidare questa previsione ad un modello autoregressivo applicato alla serie storica del consumo.

GRETL, tra i suoi strumenti della specie, ha il modello ARIMA (AutoRegressive Integrated MovingAverage).

Possiamo utilizzarlo da interfaccia grafica andando su menu MODELLO ▷ SERIE STORICHE UNIVARIATE ▷ ARIMA oppure da terminale con il comando `arima`.

Dobbiamo innanzi tutto adattarlo alla nostra serie storica e lo possiamo fare comodamente ricorrendo al terminale, che apriamo da menu STRUMENTI ▷ TERMINALE DI GRETL, utilizzando il comando `arima` che ha la seguente sintassi

`arima p d q ; <variabile>`

L'adattamento consiste nel trovare il valore dei parametri p, d e q che meglio interpretano la serie storica:

. p (altrimenti detto AR) rappresenta l'ordine dei termini autoregressivi,

. d (altrimenti detto I) rappresenta l'ordine di differenziazione,

. q (altrimenti detto MA) rappresenta l'ordine dei termini a media mobile.

Facciamo un primo tentativo attribuendo valore 1 a p, valore 0 a d e valore 1 a q con il comando

```
arima 1 0 1; C
```

La risposta a terminale comprende un sacco di dati che dicono tantissime cose utili agli esperti. Noi principianti ci limitiamo alle cose più vistose e facili da capire.

Nella zona centrale della risposta abbiamo queste indicazioni

```
            coefficiente   errore std.       z      p-value
  ---------------------------------------------------------
  const      1528,23        767,550          1,991   0,0465   **
  phi_1         0,996730      0,00460524   216,4     0,0000   ***
  theta_1       0,737274      0,121206       6,083   1,18e-09 ***
```

e notiamo un errore standard molto elevato in corrispondenza di const, evidenziato anche dal valore del test z e del suo p-value, alquanto elevato. Per i più distratti, in corrispondenza a questa riga abbiamo solo due asterischi al posto dei tre che abbiamo sulle altre righe.

Dobbiamo almeno fare in modo che anche questa riga porti tre asterischi e tentiamo di farlo portando a 2 l'ordine dei termini autoregressivi con il comando

```
arima 2 0 1; C
```

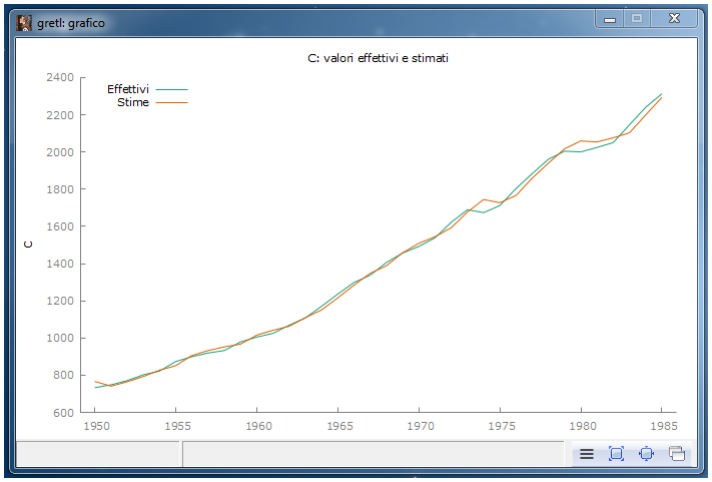
Nella risposta a terminale la zona centrale diventa questa

```
              coefficiente    errore std.        z       p-value
  ------------------------------------------------------------------
  const       1898,79        393,547             4,825   1,40e-06 ***
  phi_1          1,99751       0,00180233     1108       0,0000   ***
  phi_2         -0,999585      0,000520514   -1920       0,0000   ***
  theta_1       -0,999964      0,0792117       -12,62    1,56e-36 ***
```

e notiamo un errore standard praticamente dimezzato, con un p-value del test z praticamente azzerato, successo dimostrato dall'ottenimento dei tre asterischi anche su questa riga.

Per il nostro esercizio direi che ci possiamo accontentare e basarci sul modello ARIMA a 2, 0, 1.

Ora possiamo proseguire per arrivare alla previsione lanciando il modello da interfaccia grafica da menu MODELLO ▷ SERIE STORICHE UNIVARIATE ▷ ARIMA.
Nella finestra di dialogo che si apre inseriamo C come variabile dipendente, e i valori 2 nella finestrella AR, 0 nella finestrella I e 1 nella finestrella MA ed otteniamo così una schermata che è simile all'ultima risposta a terminale di prima, con il vantaggio che questa finestra ha una barra di menu da cui, per esempio, possiamo vedere graficamente quale sia la rispondenza del nostro modello nell'interpretare la serie storica del consumo da menu GRAFICI ▷ VALORI EFFETTIVI E STIMATI ▷ RISPETTO AL TEMPO



Confortati anche da questo grafico, affidiamo allora la previsione dei consumi nel prossimo triennio 1986, 1987, 1988 a questo modello.

Lo facciamo dalla schermata del modello da menu ANALISI ▷ PREVISIONE..., inserendo, nella successiva finestrella di dialogo, 3 nella finestrella NUMERO DI OSSERVAZIONI DA AGGIUNGERE.

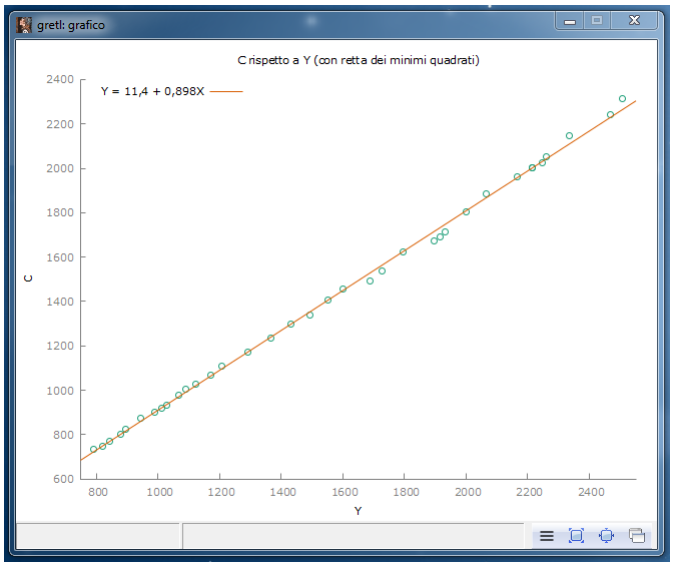
Possiamo accettare inalterata la successiva finestra di dialogo e, dando OK, otteniamo una finestra nella quale viene indicata la cercata previsione come punto centrale di un intervallo entro il quale si può collocare il valore previsto con probabilità pari a 0,95

```
Per intervalli di confidenza al 95%, z(0,025) = 1,96

                C    Previsione    Errore std   Intervallo al 95%

1986                  2364,53       25,0189      2315,49 -   2413,56
1987                  2415,47       35,3386      2346,20 -   2484,73
1988                  2465,31       43,1977      2380,64 -   2549,98
```

L'esercizio che abbiamo svolto ha previsto lo sviluppo dei consumi basandosi su un modello autoregressivo dei consumi stessi.

Sappiamo, tuttavia, che i consumi sono strettamente correlati al reddito, anzi, i consumi esistono solo se c'è reddito, cioè sono funzione del reddito e lo si vede bene dal grafico che, a pagina 23, illustra le serie storiche del reddito e del consumo che abbiamo nel nostro dataset.

Lo vediamo ancor meglio dal grafico che possiamo creare da menu VISUALIZZA ▷ GRAFICO ▷ X-Y A DISPERSIONE, mettendo sull'asse delle X la variabile Y (reddito) e sull'asse delle Y la variabile C (consumo)



Il grafico riporta anche l'equazione della retta di regressione che evidenzia il legame funzionale esistente tra consumo e reddito secondo un altro modello che GRETL ci consente di utilizzare, il così detto modello OLS (Ordinary Least Square) che troviamo, nell'interfaccia grafica, da menu MODELLO ▷ MINIMI QUADRATI ORDINARI e che possiamo lanciare da terminale con il comando ols con la seguente sintassi

`ols` <variabile_dipendente> <variabili_indipendenti>

inserendo, tra le due variabili, il valore 0 se si vuole calcolare il termine costante, altrimenti detto intercetta.

Notare come <variabili_indipendenti> sia al plurale, ad indicare che il modello si presta anche alla regressione multipla.

Avendo caricato il nostro dataset su reddito e consumi, con il comando a terminale

```
ols C 0 Y
```

otteniamo una risposta che comprende questa zona

```
             coefficiente   errore std.   rapporto t   p-value
-------------------------------------------------------------
const       11,3737       9,62946          1,181     0,2457
Y            0,898329     0,00584839     153,6       6,61e-50 ***
```

nella quale notiamo come la retta interpolante con intercetta, che viene per default calcolata nel grafico, non sia forse il meglio per interpretare la relazione esistente tra reddito e consumo.

Forse è migliore la retta senza intercetta con il coefficiente angolare che calcoliamo con il comando

```
ols C Y
```

ottenendo in risposta

```
             coefficiente   errore std.   rapporto t   p-value
-------------------------------------------------------------
Y            0,904860     0,00191589      472,3      3,54e-68 ***
```

da cui la retta dei minimi quadrati diventa $C = 0,904860Y$.

Per prevedere il consumo dei prossimi tre anni con questo modello che fa dipendere il consumo dal reddito dobbiamo predisporre una previsione del reddito per i prossimi tre anni e, se non abbiamo altre fonti più attendibili per farlo, dobbiamo ricorrere ad un modello autoregressivo come ARIMA.

Ripetendo il procedimento applicato prima per l'autoregressione sul consumo alla serie storica del reddito, ancora una volta troviamo che la migliore interpretazione dell'andamento

del reddito si ottiene con un modello ARIMA con AR pari a 2, I pari a 0 e MA = 1, con la seguente previsione

```
Per intervalli di confidenza al 95%, z(0,025) = 1,96

               Y    Previsione    Errore std   Intervallo al 95%

1986                 2555,84      29,9527      2497,14 -   2614,55
1987                 2600,63      42,2844      2517,76 -   2683,51
1988                 2643,25      51,6469      2542,02 -   2744,47
```

La previsione del consumo basata su questa previsione del reddito si può ottenere semplicemente moltiplicando per 0,904860 i valori del reddito previsto, secondo la relazione determinata prima.

Se vogliamo una indicazione comprensiva dell'intervallo di confidenza dobbiamo inserire la previsione dei redditi nel dataset agendo da menu DATI ▷ MODIFICA VALORI con selezionata la variabile Y nel foglio di lavoro di GRETL e inserendo i dati nella specie di foglio elettronico che ci si presenta, confermando alla fine con click sul segno di spunta nella barra degli strumenti.

Lanciamo ora il modello da menu MODELLO ▷ MINIMI QUADRATI ORDINARI, indicando la C come variabile dipendente e la Y come regressore (senza la const) e, agendo da menu ANALISI ▷ PREVISIONE nella finestra dei risultati, arriviamo a questa previsione del consumo

```
Per intervalli di confidenza al 95%, t(35, 0,025) = 2,030

               C    Previsione    Errore std   Intervallo al 95%

1986                 2312,64      19,5504      2272,95 -   2352,33
1987                 2353,18      19,5720      2313,45 -   2392,91
1988                 2391,73      19,5930      2351,95 -   2431,50
```

Rispetto alla previsione ottenuta con l'autoregressione sulla serie storica dei consumi che qui riporto per facilità di raffronto

```
Per intervalli di confidenza al 95%, z(0,025) = 1,96

               C    Previsione    Errore std   Intervallo al 95%

1986                 2364,53      25,0189      2315,49 -   2413,56
1987                 2415,47      35,3386      2346,20 -   2484,73
1988                 2465,31      43,1977      2380,64 -   2549,98
```

essa è più conservativa ma, a giudicare dall'errore standard, pare più attendibile, come, peraltro, dovrebbe essere se teniamo conto della subordinazione della variabile consumo rispetto alla variabile reddito.

* * *

Gli esempi che ho proposto sono tutti impostati su un rapporto interattivo con GRETL.

Numerosi esempi di script, per chi fosse interessato a questo tipo di approfondimento, vengono installati insieme a GRETL e, nel sistema Windows, si trovano nella directory di installazione, sottodirectory `scripts`, mentre in Linux e Mac si trovano in `usr/share/gretl/scripts`.

Lo script è un file con estensione `.inp` cliccando sul quale si apre l'editor degli script che ne mostra il contenuto.

Cliccando sulla sesta icona da sinistra della barra degli strumenti dell'editor, quella con il simbolo degli ingranaggi, lo script viene eseguito.