Manual dos Comandos do Gretl

Gnu Regression, Econometrics and Time-series Library

Allin Cottrell
Department of Economics
Wake Forest university

Riccardo "Jack" Lucchetti
Dipartimento di Economia
Università Politecnica delle Marche

Hélio Guilherme
Tradução Portuguesa

Setembro 2008

Conteúdo

Capítulo 1

Gretl commands

1.1 Introduction

The commands defined below may be executed interactively in the command-line client program or in the console window of the GUI program. They may also be placed in a "script" or batch file for non-interactive execution.

The following notational conventions are used below:

- A `typewriter font` is used for material that you would type directly, and also for internal names of variables.
- Terms in a *slanted font* are place-holders: you should substitute some specific replacement. For example, you might type `income` in place of the generic *xvar*.
- The construction `[` *arg* `]` means that the argument *arg* is optional: you may supply it or not (but in any case don't type the brackets).
- The phrase "estimation command" means a command that generates estimates for a given model, for example `ols`, `ar` or `wls`.

In general, each line of a command script should contain one and only one complete gretl command. There are, however, two means of continuing a long command from one line of input to another. First, if the last non-space character on a line is a backslash, this is taken as an indication that the command is continued on the following line. In addition, if the comma is a valid character in a given command (for instance, as a separator between function arguments, or as punctuation in the commands `printf` and `sscanf`) then a trailing comma also indicates continuation. To emphasize the point: a backslash may be inserted "arbitrarily" to indicate continuation, but a comma works in this capacity only if it is syntactically valid as part of the command.

1.2 Commands

add

Argumento:	*lista-de-variáveis*
Opções:	`--vcv` (mostrar matriz de covariância)
	`--quiet` (não mostrar estimativas para o modelo aumentado)
	`--silent` (não mostrar nada)
	`--inst` (acrescentar como instrumento, apenas para TSLS)
	`--both` (acrescentar tanto como regressor como instrumento, apenas para TSLS)
Exemplos:	`add 5 7 9`
	`add xx yy zz --quiet`

Tem que ser invocado após um comando de estimação. As variáveis na *lista-de-variáveis* são acrescentadas ao modelo anterior e o novo modelo é estimado. É apresentada uma estatística de significância conjunta, juntamente com o seu p-value. O teste estatístico é o F no caso de estimação por mínimos quadrados (OLS), ou qui-quadrado assimptótico de Wald nos outros casos. Um p-value abaixo de 0,05 significa que os coeficientes são conjuntamente significantes num nível de 5 porcento.

Se foi fornecida a opção `--quiet` os resultados apresentados ficam confinados ao teste da significância conjunta das variáveis acrescentadas, caso contrário, também serão mostradas as estimativas para o modelo aumentado. Neste caso, a opção `--vcv` faz com que a matriz de covariâncias dos coeficientes também seja apresentada. Se for usada a opção `--silent`, não será mostrado nada; em em todo o caso, os resultados do teste podem ser obtidos usando as variáveis especiais `$test` e `$pvalue`.

Se o modelo original foi estimado usando o método dos mínimos quadrados de duas fases, ocorre uma situação ambígua: devem as novas variáveis serem acrescentadas como sendo regressores, como instrumentos, ou como ambos? Isto resolve-se do seguinte modo: por omissão as novas variáveis são acrescentadas como regressores endógenos, mas se foi dada a opção `--inst` elas são acrescentadas como instrumentos, ou se a opção `--both` está presente elas são acrescentadas como regressores exógenos.

Caminho de Menu: Janela do modelo, /Testes/Acrescentar variáveis

adf

Argumentos:	*ordem nome-de-variável*
Opções:	`--nc` (teste sem constante)
	`--c` (apenas com constante)
	`--ct` (com constante e tendência)
	`--ctt` (com constante, tendência e quadrado da tendência)
	`--seasonals` (incluir variáveis sazonais auxiliares)
	`--verbose` (mostrar resultados da regressão)
	`--quiet` (não mostrar resultados da regressão)
	`--difference` (usar a primeira diferença da variável)
	`--test-down` (ordem 'lag' automática)
Exemplos:	`adf 0 y`
	`adf 2 y --nc --c --ct`
	`adf 12 y --c --test-down`
	Ver também`jgm-1996.inp`

Determina estatísticas para um conjunto de testes de Dickey–Fuller sobre a variável especificada, com a hipótese nula de que a variável tem uma raiz unitária. (Mas se a opção de diferenciação tiver sido dada, a primeira diferença da variável é obtida, e a discussão abaixo deve ser interpretada como sendo referente à variável transformada.)

Por omissão, são apresentadas três variantes do teste: uma baseada na regressão contendo uma constante, uma usando uma constante e uma tendência linear, e uma usando uma constante e uma tendência quadrática. Você pode controlar as variantes que são apresentadas ao especificar uma ou mais opções.

Em todos os casos a variável dependente é a primeira diferença da variável especificada, y, e a variável independente chave é o primeiro 'lag' de y. O modelo é construido de modo a que o coeficiente do 'lag' de y iguale 1 menos a raiz em questão. Por exemplo, o modelo com uma constante pode ser escrito como

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

Se a ordem de 'lag', k, é maior que 0, então k 'lags' da variável dependente são incluidos no lado direito das regressões de teste, sujeitos à seguinte qualificação. Se a opção `--test-down` foi dada, k é considerada como sendo o 'lag' máximo e a ordem de 'lag' efectivamente usada é obtida testando para baixo, de acordo com o seguinte algoritmo:

1. Estimar a regressão de Dickey–Fuller com k 'lags' da variável dependente.
2. O último 'lag' é significante? Se sim, executar o teste com com a ordem de 'lag', k. Senão, fazer k = k − 1; se k for igual a 0, executar o teste com a ordem de 'lag' 0, senão saltar para o passo 1.

No contexto do passo 2 acima, "significante" quer dizer que para o último 'lag', a estatística-t, que segue uma distribuição normal, tem um p-value bilateral assimptótico menor ou igual a 0,10.

Os p-values para os testes de Dickey–Fuller baseiam-se em MacKinnon (1996). O código relevante é incluído com a generosa permissão do autor.

Caminho de Menu: /Variável/Teste de Dickey-Fuller aumentado

append

Argumento:	*ficheiro-de-dados*

Abre um ficheiro de dados e acrescenta esse conteúdo ao conjunto de dados actual, se os novos dados forem compatíveis. O programa tentará determinar o formato do ficheiro de dados (nativo, texto simples, CVS, Gnumeric, Excel, etc.).

Caminho de Menu: /Ficheiro/Acrescentar dados

ar

Argumentos:	*'lags'* ; *variável-dependente variáveis-independentes*
Opção:	`--vcv` (mostrar matriz de covariância)
Exemplo:	`ar 1 3 4 ; y 0 x1 x2 x3`

Determina estimativas para os parâmetros usando o procedimento iteractivo e generalizado de Cochrane–Orcutt (ver a Secção 9.5 de Ramanathan, 2002). A iteração termina quando os erros das somas de quadrados sucessivos não difiram em mais que 0,005 porcento ou após 20 iterações.

'lags' é uma lista de 'lags' nos resíduos, terminada por um ponto-e-vírgula. No exemplo acima o termo do erro é especificado como

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

Caminho de Menu: /Modelo/Série temporal/Estimação autoregressiva

ar1

Argumentos:	*depvar indepvars*
Opções:	`--hilu` (use Hildreth–Lu procedure)
	`--pwe` (use Prais–Winsten estimator)
	`--vcv` (print covariance matrix)
	`--no-corc` (do not fine-tune results with Cochrane-Orcutt)
Exemplos:	`ar1 1 0 2 4 6 7`
	`ar1 y 0 xlist --hilu --no-corc`
	`ar1 y 0 xlist --pwe`

Computes feasible GLS estimates for a model in which the error term is assumed to follow a first-order autoregressive process.

The default method is the Cochrane–Orcutt iterative procedure (see, for example, Section 9.4 of Ramanathan, 2002). Iteration is terminated when successive estimates of the autocorrelation coefficient do not differ by more than 0.001 or after 20 iterations.

If the `--hilu` option is given, the Hildreth–Lu search procedure is used. The results are then fine-tuned using the Cochrane–Orcutt method, unless the `--no-corc` flag is specified. (The latter option is ignored if `--hilu` is not specified.)

If the `--pwe` option is given, the Prais–Winsten estimator is used. This involves an an iteration similar to Cochrane–Orcutt; the difference is that while Cochrane–Orcutt discards the first observation, Prais–Winsten makes use of it. See, for example, Chapter 13 of Greene's *Econometric Analysis* (2000) for details.

Caminho de Menu: /Model/Time series/Cochrane-Orcutt

Caminho de Menu: /Model/Time series/Hildreth-Lu

Caminho de Menu: /Model/Time series/Prais-Winsten

arbond

Argumento:	*p* `[` *q* `]` `;` *variável-dependente variáveis-independentes* `[ ;` *instrumentos* `]`
Opções:	`--vcv` (mostrar matriz de covariância)
	`--two-step` (executa estimação pelo Método Generalizado dos Momentos (GMM) de 2-fases)
	`--time-dummies` (acrescenta variáveis auxiliares tempo)
	`--asymptotic` (erros padrão assimptóticos)
Exemplos:	`arbond 2 ; y Dx1 Dx2`
	`arbond 2 5 ; y Dx1 Dx2 ; Dx1`
	`arbond 1 ; y Dx1 Dx2 ; Dx1 GMM(x2,2,3)`
	Ver também`arbond91.inp`

Executa a estimação de modelos de painel dinâmico (ou seja, modelos de painel que contenham um ou mais 'lags' da variável dependente) recorrendo ao método Método Generalizado dos Momentos (GMM) desenvolvido por Arellano e Bond (1991).

O parâmetro *p* representa a ordem da autoregressão para a variável dependente. O parâmetro opcional *q* indica o máximo 'lag' do nível da variável dependente a ser usada como um instrumento. Se este argumento for omitido, ou de valor 0, todos os 'lags' disponíveis são usados.

A variável dependente deve ser dada na forma de níveis; ela será automaticamente diferenciada (pois este estimador usa diferenciação para anular os efeitos individuais). As variáveis independentes não são automaticamente diferenciadas; se você pretende usar diferenças (o que acontece em geral para variáveis quantitativas, mas não será, por exemplo, para variáveis auxiliares temporais), deve primeiro criar essas diferenças e depois especificar estas como sendo regressoras.

O último campo (opcional) do comando serve para especificar instrumentos. Se não for dado nenhum, então é assumido que todas as variáveis independentes são estritamente exógenas. Se você especificar alguns instrumentos, então deve incluir na lista quaisquer variáveis independentes estritamente exógenas. Para regressores predeterminados, você pode usar a função `GMM` para incluir uma gama de 'lags' especificada no modo bloco-diagonal. Isto é ilustrado no terceiro exemplo acima. O primeiro argumento de `GMM` é o nome da variável em questão, o segundo é o mínimo 'lag' a ser usado como instrumento, e o terceiro é o máximo 'lag'. Se o terceiro argumento for dado como 0, todos os 'lags' disponíveis são usados.

Por omissão são apresentados os resultados da estimação 1-fase (com erros padrão robustos). Opcionalmente, você pode escolher estimação de 2-fases. Em ambos os casos são efectuados testes de autocorrelação de ordem 1 e 2, assim como o teste de sobre-identificação de Sargan e o teste de Wald para a significância conjunta dos regressores. Note-se que este modelo de diferenciação com autocorrelação de primeira-ordem não invalida o modelo, mas que a autocorrelação de segunda-ordem não respeita as assunções estatísticas presentes.

No caso da estimação de 2-fases, por omissão, os erros padrão são determinados usando a correcção de amostra-finita sugerida por Windmeijer (2005). Os erros padrão assimptóticos associados ao estimador de 2-fases são em geral considerados como guias para inferência pouco fiáveis, mas se por alguma razão os pretender ver, você pode usar a opção `--asymptotic` para desligar a correcção de Windmeijer.

Se for dada a opção `--time-dummies`, são acrescentadas variáveis auxiliares temporais aos regressores especificados. Para evitar colinearidade perfeita com a constante, o número de auxiliares é uma unidade a menos que o número máximo de períodos usados na estimação. As auxiliares são introduzidas por níveis; se você deseja usar auxiliares de tempo na forma de primeiras-diferenças, você terá que definir e acrescentar essas variáveis manualmente.

Caminho de Menu: /Modelo/Painel/Arellano-Bond

arch

Argumentos:	*ordem variável-dependente variáveis-independentes*
Exemplo:	`arch 4 y 0 x1 x2 x3`

Testa o modelo em ARCH (Heterosquedicidade Condicional Autoregressiva) da ordem de 'lag' especificada. Se a estatística de teste LM tiver um p-value abaixo de 0,10, então a estimação ARCH também é executada. Se a variância predita de qualquer observação na regressão auxiliar não for positiva, então é usado o correspondente resíduo ao quadrado. Segue-se uma estimação por Mínimos Quadrados com Pesos sobre o modelo original.

Ver também garch.

Caminho de Menu: Janela do modelo, /Testes/ARCH

arima

Argumentos:	*p d q* [; *P D Q*] ; *variável-dependente* [*variáveis-independentes*]
Opções:	`--verbose` (mostrar detalhes das iterações)
	`--vcv` (mostrar matriz de covariância)
	`--nc` (não incluir uma constante)
	`--conditional` (usar verosimilhança máxima condicional)
	`--x-12-arima` (usar X-12-ARIMA para estimação)
Exemplos:	`arima 1 0 2 ; y`
	`arima 2 0 2 ; y 0 x1 x2 --verbose`
	`arima 0 1 1 ; 0 1 1 ; y --nc`

Se não for dada a lista de *variáveis-independentes*, é estimado um modelo ARIMA (Média Móvel, Autoregressiva, Integrada) univariado. O valores inteiros p, d e q representam respectivamente, a ordem autoregressiva (AR), a ordem de diferenciação, e ordem da média móvel (MA). Estes valores podem ser fornecidos na forma numérica, ou como nome de variáveis escalares pré-existentes. Por exemplo, um valor de 1 em d, significa que a primeira diferença da variável dependente deve ser obtida antes de estimar os parâmetros ARMA.

Os valores inteiros opcionais,P, D e Q representam respectivamente, a sazonalidade AR, a ordem para diferenciação de sazonalidade e a ordem de sazonalidade MA. Estes são apenas aplicáveis se os dados tiverem uma frequência superior a 1 (por exemplo, quadrimestral ou mensal). Mais uma vez, estas ordens podem ser dadas na forma numérica ou como variáveis.

No caso univariado é incluído no modelo por omissão, um interceptor, mas isto pode ser suprimido com a opção `--nc`. Se forem fornecidas *variáveis-independentes*, o modelo passa a ser ARMAX; neste caso a constante deve ser explicitamente incluída se você pretender um interceptor (tal como no segundo exemplo acima).

Existe outra forma alternativa para este comando: se você não pretende aplicar diferenciação (seja sazonal ou não-sazonal), você pode omitir ambos os parâmetros d e D, em vez de entrar explicitamente zeros. Além disso, `arma` é um sinónimo ou aliás para `arima`. Assim, por exemplo, o comando seguinte é válido para especificar o modelo ARMA(2, 1):

```
arma 2 1 ; y
```

O normal é usar a funcionalidade "nativa" gretl ARMA, com estimação de Máxima Verosimilhança (ML) exacta (usando o filtro de Kalman). Outras opções são: código nativo, ML condicional; X-12-ARIMA, ML exacta; e X-12-ARIMA, ML condicional. (As últimas duas opções estão disponíveis apenas se o programa X-12-ARIMA estiver instalado.) Para detalhes sobre estas opções, veja por favor *Manual de Utilização do Gretl*.

O valor AIC retornado em ligação com os modelos ARIMA é calculado conforme a definição usada no programa X-12-ARIMA, nomeadamente

$$\mathrm{AIC} = -2\ell + 2k$$

onde ℓ é o logaritmo da verosimilhança e k é o número total de parâmetros estimados. Note-se que o programa X-12-ARIMA não produz critérios de informação tal como o AIC quando a estimação é por ML condicional.

A imagem da "frequência" apresentada em ligação com as raízes AR e MA é valor λ que resolve

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

onde z é a raiz em questão e r o seu módulo.

Caminho de Menu: /Modelo/Série temporal/ARIMA

Acesso alternativo: Menu de contexto da janela principal (selecção singular)

boxplot

Argumento:	*lista-de-variáveis*
Opção:	`--notches` (mostrar intervalo de 90 porcento para a mediana delimitado por entalhes)

Estes gráficos (criados por Tukey e Chambers) apresentam a distribuição de uma variável. A caixa central contém os 50 porcento dos dados centrais, i.e. está limitada pelos primeiro e terceiro quartis. Os "bigodes" estendem-se até aos valores mínimo e máximo. É desenhada uma linha que corta a caixa na mediana.

No caso de caixas com entalhes, os entalhes representam os limites do intervalo de confiança para a mediana de cerca de 90 porcento. Isto é obtido usando o método 'bootstrap'.

A seguir a cada variável indicada no comando caixa-com-bigodes, pode-se acrescentar uma expressão Booleana para restringir a variável em questão. Tem que se inserir um espaço entre o nome da variável ou número, e a expressão. Suponha que você dispõe de valores de salários (`salary`) para homens e mulheres, e que tem a variável auxiliar `GENDER` com valor 1 para homens e 0 para mulheres. Nesse caso você podia ter gráficos caixa-com-bigodes comparativos com a seguinte *lista-de-variáveis*:

```
salary (GENDER=1) salary (GENDER=0)
```

Alguns detalhes das caixas-com-bigodes do gretl podem ser controlados por intermédio de um ficheiro (de texto simples) com o nome `.boxplotrc`. Para mais detalhes sobre isto veja *Manual de Utilização do Gretl*.

Caminho de Menu: /Ver/Gráfico das variáveis/Caixa com bigodes

break

Sai de um ciclo. Este comando pode apenas ser usado dentro de um ciclo; ele termina a execução de comandos e sai de dentro do ciclo (o mais interior). Ver também loop.

chow

Argumento:	*obs*
Exemplos:	`chow 25`
	`chow 1988:1`

Tem que se seguir a uma regressão de Mínimos Quadrados (OLS). Cria uma variável auxiliar que é igual a 1 a partir do ponto especificado por *obs* até ao final da amostra, caso contrário é 0, e cria também termos de interacção entre esta variável auxiliar e as variáveis independentes originais. É executada uma regressão aumentada que inclui estes termos e é calculada a estatística F, considerando a regressão aumentada como não restringida e a original como restringida. Esta estatística é apropriada para testar a hipótese nula de não existência de quebra estrutural no ponto de separação dado.

Caminho de Menu: Janela do modelo, /Testes/Teste de Chow

coeffsum

Argumento:	*lista-de-variáveis*
Exemplo:	`coeffsum xt xt_1 xr_2`
	`restrict.inp`

Tem que se seguir a uma regressão. Calcula a soma dos coeficientes nas variáveis indicadas na *lista-de-variáveis*. Apresenta esta soma juntamente com o seu erro padrão e o p-value para a hipótese nula de que a soma é zero.

Note-se a diferença entre este teste e omit, que testa a hipótese nula de que os coeficientes num conjunto especificado de variáveis independentes são *todos* iguais a zero.

Caminho de Menu: Janela do modelo, /Testes/Soma de coeficientes

coint

Argumentos:	*ordem variável-dependente variáveis-independentes*
Opções:	`--nc` (não incluir uma constante)
	`--ct` (incluir constante e tendência)
	`--ctt` (incluir constante e tendência quadrática)
	`--skip-df` (não efectuar testes DF nas variáveis individuais)
Exemplos:	`coint 4 y x1 x2`
	`coint 0 y x1 x2 --ct --skip-df`

O teste de cointegração Engle–Granger. O procedimento por omissão é: (1) efectuar testes de Dickey–Fuller (DF) segundo a hipótese nula de que cada variável listada tem uma raiz unitária; (2) estima a regressão de cointegração; e (3)executar um teste DF sobre os resíduos da regressão de cointegração. Se for dada a opção `--skip-df`, o passo (1) é omitido.

Se a ordem de 'lag' especificada é positiva, todos os testes Dickey–Fuller usam essa ordem. Se a ordem for antecedida de um sinal menos, ela é encarada como sendo o máximo 'lag' e a ordem efectivamente usada em cada caso é obtida testando para baixo: ver o comando adf para detalhes.

Por omissão, a regressão de cointegração contém uma constante. Se você deseja suprimir a constante, acrescente a opção `--nc`. Se você deseja aumentar a lista de termos determinísticos na regressão de cointegração com uma tendência linear ou quadrática, use a opção `--ct` ou `--ctt`. Estas opções são mutualmente exclusivas.

Os *P*-values para este teste são baseados em MacKinnon (1996). O código relevante é incluído com a generosa permissão do autor.

Caminho de Menu: /Modelo/Série temporal/Testes de Cointegração/Engle-Granger

coint2

Argumentos:	*ordem variável-dependente variáveis-independentes*
Opções:	`--nc` (sem constante)
	`--rc` (constante restringida)
	`--crt` (constante e tendência restringida)
	`--ct` (constante e tendência não restringida)
	`--seasonals` (incluir auxiliares sazonais centradas)
	`--quiet` (apenas mostrar os testes)
	`--verbose` (mostrar detalhes das regressões auxiliares)
Exemplos:	`coint2 2 y x`
	`coint2 4 y x1 x2 --verbose`
	`coint2 3 y x1 x2 --rc`

Executa o teste de Johansen para a cointegração entre as variáveis listadas para a dada ordem de 'lag'. Os valores críticos são determinados usando a aproximação gamma de J. Doornik (Doornik, 1998). Para detalhes sobre este teste ver o Capítulo 20 do livro de Hamilton, *Time Series Analysis* (1994).

A inclusão de termos determinísticos no modelo é controlada por intermédio das opções. Por omissão, se não tiver sido indicada nenhuma opção, será incluída uma "constante não restringida", o que permite a presença de um interceptor não-nulo nas relações cointegrantes assim como uma tendência nos níveis das variáveis endógenas. Na literatura derivada do trabalho de Johansen (ver por exemplo o livro dele de 1995) isto é frequentemente referido como sendo o "caso 3". As primeiras quatro opções apresentadas acima, que são mutualmente exclusivas, produzem respectivamente os casos 1, 2, 4, e 5. O significado destes casos e os critérios para seleccionar um caso estão explicados no *Manual de Utilização do Gretl*.

A opção `--seasonals`, que pode ser combinada com qualquer outra opção, especifica a inclusão de um conjunto de variáveis auxiliares sazonais. Esta opção apenas está disponível para dados trimestrais ou mensais.

A seguinte tabela serve como uma guia à interpretação dos resultados apresentados pelo teste, num caso de 3-variáveis. `H0` significa a hipótese nula, `H1` a hipótese alternativa, e `c` o número de relações cointegrantes.

```
Ordem    Teste Traço        Teste Lmax
         H0     H1          H0     H1
---------------------------------------
 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
---------------------------------------
```

Ver também o comando vecm.

Caminho de Menu: /Modelo/Série temporal/Testes de Cointegração/Johansen

corr

Argumento:	[*lista-de-variáveis*]
Exemplo:	`corr y x1 x2 x3`

Apresenta os coeficientes de correlação emparelhados das variáveis em *lista-de-variáveis*, ou de todas as variáveis no conjunto de dados se não for dada a *lista-de-variáveis*.

Caminho de Menu: /Ver/Matriz de correlação

Acesso alternativo: Menu de contexto da janela principal (selecção múltipla)

corrgm

Argumentos:	*variável* [*maxlag*]
Exemplo:	`corrgm x 12`

Apresenta os valores da função de autocorrelação para a *variável*, que pode ser especificada por nome ou por número. Os valores são definidos pela equação $\hat{\rho}(u_t, u_{t-s})$ onde u$_t$ é a t–ésima observação da variável u e s é o número de "lags".

Também são apresentadas as autocorrelações parciais (obtidas segundo o algoritmo de Durbin–Levinson): estas constituem a rede dos efeitos dos "lags"intervenientes. O comando também produz o gráfico correlograma e apresenta a estatística de teste Q de Box–Pierce, para a hipótese nula de que a série temporal é "ruído branco": terá uma distribuição qui-quadrado assimptótico com os graus de liberdade iguais ao número de "lags"usados.

Se o valor *maxlag* for especificado o comprimento do correlograma fica limitado a esse máximo número de "lags", senão o comprimento é determinado automáticamente, como uma função da frequência dos dados e do número de observações.

Caminho de Menu: /Variável/Correlograma

Acesso alternativo: Menu de contexto da janela principal (selecção singular)

criteria

Argumentos:	*ess T k*
Exemplo:	`criteria 23.45 45 8`

Determina o Critério de Informação de Akaike (AIC) e o Critério de Informação Bayesiano de Schwarz (BIC), dados *ess* (erro da soma de quadrados), o número de observações (T), e o número de coeficientes (k). T, k, e *ess* podem ser valores numéricos ou nomes de variáveis préviamente definidas.

O AIC é obtido segundo a formulação original de Akaike (1974), nomeadamente

$$\mathrm{AIC} = -2\ell + 2k$$

onde ℓ designa a verosimilhança logaritmica maximizada. O BIC é calculado por

$$\mathrm{BIC} = -2\ell + k \log T$$

Por favor consulte *Manual de Utilização do Gretl* para mais pormenores.

cusum

Opção: `--squares` (executa o teste CUSUMSQ)

Tem que se seguir à estimação de um modelo por via de OLS. Executa o teste CUSUM —ou se for dada a opção `--squares`, o teste CUSUMSQ —para a estabilidade dos parâmetros. É obtida uma série temporal de erros de predição um passo-à-frente, pela execução de séries de regressões: a primeira regressão usa as primeiras k observações e é usada para gerar a predição da variável dependente na observação k + 1; a segunda usa a primeira predição para a observação k + 2, e por aí a diante (onde k é o número de parâmetros no modelo original).

A soma acumulada dos erros de predição escalados, ou os quadrados desses erros, é mostrada e apresentada em gráfico. A hipótese nula para a estabilidade dos parâmetros é rejeitada ao nível de cinco porcento, se a soma acumulada se desviar do intervalo de confiança de 95 porcento.

No caso do teste CUSUM, é também apresentada a estatística de teste t de Harvey–Collier, para a hipótese nula da estabilidade dos parâmetros. Ver o livro *Econometric Analysis* de Greene para mais detalhes. Para o teste CUSUMSQ, o intervalo de confiança a 95 porcento é calculado de acordo com o algoritmo apresentado por Edgerton e Wells (1994).

Caminho de Menu: Janela do modelo, /Testes/Teste CUSUM(SQ)

data

Argumento: *lista-de-variáveis*

Reads the variables in *lista-de-variáveis* from a database (gretl, RATS 4.0 or PcGive), which must have been opened previously using the open command. The data frequency and sample range may be established via the setobs and smpl commands prior to using this command. Here is a full example:

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

The commands above open a database named `macrodat.rat`, establish a quarterly data set starting in the first quarter of 1959 and ending in the fourth quarter of 1999, and then import the series named `GDP_JP` and `GDP_UK`.

If `setobs` and `smpl` are not specified in this way, the data frequency and sample range are set using the first variable read from the database.

If the series to be read are of higher frequency than the working data set, you may specify a compaction method as below:

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

The four available compaction methods are “average” (takes the mean of the high frequency observations), “last” (uses the last observation), “first” and “sum”. If no method is specified, the default is to use the average.

Caminho de Menu: /Ficheiro/Databases

dataset

Argumentos:	*keyword parameters*
Exemplos:	`dataset addobs 24`
	`dataset compact 1`
	`dataset compact 4 last`
	`dataset expand 12`
	`dataset transpose`
	`dataset sortby x1`

Performs various operations on the data set as a whole, depending on the given *keyword*, which must be `addobs`, `compact`, `expand`, `transpose`, `sortby` or `dsortby`. Note: these actions are not available when the dataset is currently subsampled by selection of cases on some Boolean criterion.

`addobs`: Must be followed by a positive integer. Adds the specified number of extra observations to the end of the working dataset. This is primarily intended for forecasting purposes. The values of most variables over the additional range will be set to missing, but certain deterministic variables are recognized and extended, namely, a simple linear trend and periodic dummy variables.

`compact`: Must be followed by a positive integer representing a new data frequency, which should be lower than the current frequency (for example, a value of 4 when the current frequency is 12 indicates compaction from monthly to quarterly). This command is available for time series data only; it compacts all the series in the data set to the new frequency. A second parameter may be given, namely one of `sum`, `first` or `last`, to specify, respectively, compaction using the sum of the higher-frequency values, start-of-period values or end-of-period values. The default is to compact by averaging.

`expand`: Must be followed by a positive integer representing a new data frequency, which should be higher than the current frequency. This command is only available for annual or quarterly time series data. Annual data can be expanded to quarterly or monthly; quarterly data can be expanded to monthly. All the series in the data set are padded out to the new frequency by repeating the existing values.

`transpose`: No additional parameter required. Transposes the current data set. That is, each observation (row) in the current data set will be treated as a variable (column), and each variable as an observation. This command may be useful if data have been read from some external source in which the rows of the data table represent variables.

`sortby`: One variable name is required; this variable is used as a sort key. The observations on all variables in the dataset are re-ordered by increasing value of the key variable. This command is available only for undated data.

`dsortby`: Works as `sortby` except that the re-ordering is by decreasing value of the key variable.

Caminho de Menu: /Dados/Acrescentar observações

Caminho de Menu: /Dados/Compactar datos

Caminho de Menu: /Dados/Expandir datos

Caminho de Menu: /Dados/Transpôr

Caminho de Menu: /Dados/Ordenar

delete

Argumento: [*lista-de-variáveis*]

Removes the listed variables (given by name or number) from the dataset. *Use with caution*: no confirmation is asked, and any variables with higher ID numbers will be re-numbered.

If no *lista-de-variáveis* is given with this command, it deletes the last (highest numbered) variable from the dataset.

Caminho de Menu: Main window pop-up (single selection)

diff

Argumento: *lista-de-variáveis*

The first difference of each variable in *lista-de-variáveis* is obtained and the result stored in a new variable with the prefix `d_`. Thus diff x y creates the new variables

```
d_x = x(t) - x(t-1)
d_y = y(t) - y(t-1)
```

Caminho de Menu: /Add/First differences of selected variables

difftest

Argumentos:	*var1 var2*
Opções:	`--sign` (Sign test, the default)
	`--rank-sum` (Wilcoxon rank-sum test)
	`--signed-rank` (Wilcoxon signed-rank test)
	`--verbose` (print extra output)

Carries out a nonparametric test for a difference between two populations or groups, the specific test depending on the option selected.

With the `--sign` option, the Sign test is performed. This test is based on the fact that if two samples, x and y, are drawn randomly from the same distribution, the probability that $x_i > y_i$, for each observation i, should equal 0.5. The test statistic is w, the number of observations for which $x_i > y_i$. Under the null hypothesis this follows the Binomial distribution with parameters (n, 0.5), where n is the number of observations.

With the `--rank-sum` option, the Wilcoxon rank-sum test is performed. This test proceeds by ranking the observations from both samples jointly, from smallest to largest, then finding the sum of the ranks of the observations from one of the samples. The two samples do not have to be of the same size, and if they differ the smaller sample is used in calculating the rank-sum. Under the null hypothesis that the samples are drawn from populations with the same median, the probability distribution of the rank-sum can be computed for any given sample sizes; and for reasonably large samples a close Normal approximation exists.

With the `--signed-rank` option, the Wilcoxon signed-rank test is performed. This is designed for matched data pairs such as, for example, the values of a variable for a sample of individuals before and after some treatment. The test proceeds by finding the differences between the paired observations, $x_i - y_i$, ranking these differences by absolute value, then assigning to each pair a signed rank, the sign agreeing with the sign of the difference. One then calculates W_+, the sum of the positive signed ranks. As with the rank-sum test, this statistic has a well-defined distribution under the null that the median difference is zero, which converges to the Normal for samples of reasonable size.

For the Wilcoxon tests, if the `--verbose` option is given then the ranking is printed. (This option has no effect if the Sign test is selected.)

discrete

Argumento:	*lista-de-variáveis*
Opção:	`--reverse` (mark variables as continuous)

Marks each variable in *lista-de-variáveis* as being discrete. By default all variables are treated as continuous; marking a variable as discrete affects the way the variable is handled in frequency plots, and also allows you to select the variable for the command dummify.

If the `--reverse` flag is given, the operation is reversed; that is, the variables in *lista-de-variáveis* are marked as being continuous.

Caminho de Menu: /Variável/Edit attributes

dummify

Argumento:	*lista-de-variáveis*
Opções:	`--drop-first` (omit lowest value from encoding)
	`--drop-last` (omit highest value from encoding)

For any suitable variables in *lista-de-variáveis*, creates a set of dummy variables coding for the distinct values of that variable. Suitable variables are those that have been explicitly marked as discrete, or those that take on a fairly small number of values all of which are "fairly round" (multiples of 0.25).

By default a dummy variable is added for each distinct value of the variable in question. For example if a discrete variable `x` has 5 distinct values, 5 dummy variables will be added to the data set, with names `Dx_1`, `Dx_2` and so on. The first dummy variable will have value 1 for observations where `x` takes on its smallest value, 0 otherwise; the next dummy will have value 1 when `x` takes on its second-smallest value, and so on. If one of the option flags `--drop-first` or `--drop-last` is added, then either the lowest or the highest value of each variable is omitted from the encoding (which may be useful for avoiding the "dummy variable trap").

This command can also be embedded in the context of a regression specification. For example, the following line specifies a model where `y` is regressed on the set of dummy variables coding for `x`. (Option flags cannot be passed to dummify in this context.)

```
ols y dummify(x)
```

elif

See if.

else

See if.

end

Ends a block of commands of some sort. For example, end system terminates an equation system.

endif

See if.

endloop

Marks the end of a command loop. See loop.

eqnprint

Argumento:	[*-f filename*]
Opção:	`--complete` (Create a complete document)

Must follow the estimation of a model. Prints the estimated model in the form of a LaTeX equation. If a filename is specified using the `-f` flag output goes to that file, otherwise it goes to a file with a name of the form `equation_N.tex`, where `N` is the number of models estimated to date in the current session. See also tabprint.

If the `--complete` flag is given, the LaTeX file is a complete document, ready for processing; otherwise it must be included in a document.

Caminho de Menu: Janela do modelo, /LaTeX

equation

Argumentos:	*variável-dependente variáveis-independentes*
Exemplo:	`equation y x1 x2 x3 const`

Specifies an equation within a system of equations (see system). The syntax for specifying an equation within an SUR system is the same as that for, e.g., ols. For an equation within a Three-Stage Least Squares system you may either (a) give an OLS-type equation specification and provide a common list of instruments using the instr keyword (again, see system), or (b) use the same equation syntax as for tsls.

estimate

Argumentos:	*systemname estimator*
Opções:	`--iterate` (iterate to convergence)
	`--no-df-corr` (no degrees of freedom correction)
	`--geomean` (see below)
Exemplos:	`estimate "Klein Model 1"method=fiml`
	`estimate Sys1 method=sur`
	`estimate Sys1 method=sur --iterate`

Calls for estimation of a system of equations, which must have been previously defined using the system command. The name of the system should be given first, surrounded by double quotes if the name contains spaces. The estimator, which must be one of ols, tsls, sur, 3sls, fiml or liml, is preceded by the string `method=`.

If the system in question has had a set of restrictions applied (see the restrict command), estimation will be subject to the specified restrictions.

If the estimation method is sur or 3sls and the `--iterate` flag is given, the estimator will be iterated. In the case of SUR, if the procedure converges the results are maximum likelihood estimates. Iteration of three-stage least squares, however, does not in general converge on the full-information maximum likelihood results. The `--iterate` flag is ignored for other methods of estimation.

If the equation-by-equation estimators ols or tsls are chosen, the default is to apply a degrees of freedom correction when calculating standard errors. This can be suppressed using the `--no-df-corr` flag. This flag has no effect with the other estimators; no degrees of freedom correction is applied in any case.

By default, the formula used in calculating the elements of the cross-equation covariance matrix is

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

If the `--geomean` flag is given, a degrees of freedom correction is applied: the formula is

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

where the ks denote the number of independent parameters in each equation.

fcast

Argumentos:	[*startobs endobs*] *fitvar*
Opções:	`--dynamic` (create dynamic forecast)
	`--static` (create static forecast)
Exemplos:	`fcast 1997:1 2001:4 f1`
	`fcast fit2`

Must follow an estimation command. Forecasts are generated for the specified range (or the largest possible range if no *startobs* and *endobs* are given) and the values saved as *fitvar*, which can be printed, graphed, or plotted. The right-hand side variables are those in the original model. There is no provision to substitute other variables. If an autoregressive error process is specified the forecast incorporates the predictable fraction of the error process.

The choice between a static and a dynamic forecast applies only in the case of dynamic models, with an autoregressive error process or including one or more lagged values of the dependent variable as regressors. See fcasterr for more details.

Caminho de Menu: Janela do modelo, /Analysis/Forecasts

foreign

Argumento:	[*language-spec*]
Opções:	`--send-data` (pre-load the current dataset)
	`--quiet` (suppress output from foreign program)

This command opens a special mode in which commands to be executed by another program are accepted. You exit this mode with `end foreign`; at this point the stacked commands are executed.

At present the only "foreign" program supported in this way is GNU R, and the only acceptable *language-spec* is `language=R` (which is implicit, and may be omitted). In future other programs may be supported.

See *Manual de Utilização do Gretl* for details and examples.

freq

Argumento:	*var*
Opções:	`--quiet` (suppress printing of histogram)
	`--gamma` (test for gamma distribution)

With no options given, displays the frequency distribution for *var* (given by name or number) and shows the results of the Doornik–Hansen chi-square test for normality.

If the `--quiet` option is given, the histogram is not shown. If the `--gamma` option is given, the test for normality is replaced by Locke's nonparametric test for the null hypothesis that the variable follows the gamma distribution; see Locke (1976), Shapiro and Chen (2001).

In interactive mode a graph of the distribution is displayed.

Caminho de Menu: /Variável/Frequency distribution

function

Argumento:	*fnname*

Opens a block of statements in which a function is defined. This block must be closed with `end function`. Please see *Manual de Utilização do Gretl* for details.

garch

Argumentos:	*p q* `;` *variável-dependente* `[` *variáveis-independentes* `]`
Opções:	`--robust` (robust standard errors)
	`--verbose` (mostrar detalhes das iterações)
	`--vcv` (mostrar matriz de covariância)
	`--arma-init` (initial variance parameters from ARMA)
Exemplos:	`garch 1 1 ; y`
	`garch 1 1 ; y 0 x1 x2 --robust`

Estimates a GARCH model (GARCH = Generalized Autoregressive Conditional Heteroskedasticity), either a univariate model or, if *variáveis-independentes* are specified, including the given exogenous variables. The integer values *p* and *q* (which may be given in numerical form or as the names of pre-existing scalar variables) represent the lag orders in the conditional variance equation:

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

The gretl GARCH algorithm is basically that of Fiorentini, Calzolari and Panattoni (1996), used by kind permission of Professor Fiorentini.

Several variant estimates of the coefficient covariance matrix are available with this command. By default, the Hessian is used unless the `--robust` option is given, in which case the QML (White) covariance matrix is used. Other possibilities (e.g. the information matrix, or the Bollerslev–Wooldridge estimator) can be specified using the set command.

By default, the estimates of the variance parameters are initialized using the unconditional error variance from initial OLS estimation for the constant, and small positive values for the coefficients on the past values of the squared error and the error variance. The flag `--arma-init` calls for the starting values of these parameters to be set using an initial ARMA model, exploiting the relationship between GARCH and ARMA set out in Chapter 21 of Hamilton's *Time Series Analysis*. In some cases this may improve the chances of convergence.

The GARCH residuals and estimated conditional variance can be retrieved as `$uhat` and `$h` respectively. For example, to get the conditional variance:

```
genr ht = $h
```

Caminho de Menu: /Modelo/Série temporal/GARCH

genr

Argumentos: *newvar* = *formula*

Creates new variables, usually through transformations of existing variables. See also diff, logs, lags, ldiff, multiply, sdiff and square for shortcuts. In the context of a `genr` formula, existing variables must be referenced by name, not ID number. The formula should be a well-formed combination of variable names, constants, operators and functions (described below). Note that further details on some aspects of this command can be found in *Manual de Utilização do Gretl*.

This command may yield either a series or a scalar result. For example, the formula `x2 = x * 2` naturally yields a series if the variable `x` is a series and a scalar if `x` is a scalar. The formulae `x = 0` and `mx = mean(x)` naturally return scalars. Under some circumstances you may want to have a scalar result expanded into a series or vector. You can do this by using `series` as an "alias" for the `genr` command. For example, `series x = 0` produces a series all of whose values are set to 0. You can also use `scalar` as an alias for `genr`. It is not possible to coerce a vector result into a scalar, but use of this keyword indicates that the result *should be* a scalar: if it is not, an error occurs.

When a formula yields a series or vector result, the range over which the result is written to the target variable depends on the current sample setting. It is possible, therefore, to define a series piecewise using the `smpl` command in conjunction with `genr`.

Supported *arithmetical operators* are, in order of precedence: `^` (exponentiation); `*`, `/` and `%` (modulus or remainder); `+` and `-`.

The available *Boolean operators* are (again, in order of precedence): `!` (negation), `&&` (logical AND), `||` (logical OR), `>`, `<`, `=`, `>=` (greater than or equal), `<=` (less than or equal) and `!=` (not equal). The Boolean operators can be used in constructing dummy variables: for instance `(x > 10)` returns 1 if `x` > 10, 0 otherwise.

Built-in constants are `pi` and `NA`. The latter is the missing value code: you can initialize a variable to the missing value with `scalar x = NA`.

Supported *functions* fall into these groups:

- Standard mathematical functions: abs, exp, int (integer part), ln (natural logarithm: log is a synonym), log10 (log to the base 10), log2 (log to the base 2), sqrt. All of these take a single argument, which may be either a series or a scalar.
- Trigonometric functions: cos, sin, tan and atan (arc tangent). These functions also take a single series or scalar argument.
- Standard statistical functions taking a single argument and yielding a scalar result: max (maximum value in a series), min (minimum value in series), mean (arithmetic mean), median, var (variance), sd (standard deviation), sst (sum of squared deviations from the mean), sum, gini (Gini coefficient).
- Statistical functions taking one series as argument and yielding a series or vector result: sort (sort a series in ascending order of magnitude), dsort (sort in descending order), cum (cumulate, or running sum). For panel data only, the functions pmean and psd take a single series as argument and return series containing, respectively, the means and standard deviations of the variable for each of the cross-sectional units. See *Manual de Utilização do Gretl* for details.
- Statistical functions taking two series as arguments and yielding a scalar result: cov (covariance), corr (correlation coefficient).

- Special statistical functions: pvalue, cdf and critical (see below), cnorm (standard normal CDF), dnorm (standard normal PDF), qnorm (standard normal quantiles, or inverse of standard normal PDF), resample (resample a series with replacement, for bootstrap purposes), hpfilt (Hodrick–Prescott filter: this function returns the "cycle" component of the series), bkfilt (Baxter–King bandpass filter).

- Time-series functions: diff (first difference), ldiff (log-difference, or first difference of natural logs), sdiff (seasonal difference), fracdiff (fractional difference). To generate lags of a variable `x`, use the syntax x(-N), where `N` represents the desired lag length; to generate leads, use x(+N).

- Dataset functions yielding a series: misszero (replaces the missing observation code in a given series with zeros); zeromiss (the inverse operation to misszero); missing (at each observation, 1 if the argument has a missing value, 0 otherwise); ok (the opposite of missing).

- Dataset functions yielding a scalar: nobs (gives the number of valid observations in a data series), firstobs (gives the 1-based observation number of the first non-missing value in a series), lastobs (observation number of the last non-missing observation in a series).

- Pseudo-random numbers: uniform, normal, chisq, student, binomial and poisson. The first two functions do not take an argument and should be written with empty parentheses: `uniform()`, `normal()`. They create pseudo-random series drawn from the uniform (0–1) and standard normal distributions respectively. The next two take a single argument, namely the (integer) degrees of freedom; they generate drawings from the Chi-square and Student's t distributions respectively. The binomial function takes two parameters: an integer number of trials and a "success" probability. The poisson function takes a single argument, the mean, which may be either a scalar or a series. Uniform series, which form the basis for the more complex distributions, are generated using the Mersenne Twister;See Matsumoto and Nishimura (1998). The implementation is provided by glib, if available, or by the C code written by Nishimura and Matsumoto.

  for normal series the method of Box and Muller (1958) is used, taking input from the Mersenne Twister. Ver também o comando set command, `seed` option.

With a few exceptions as noted, all the above functions take as their single argument either the name of a variable or an expression that evaluates to a variable (e.g. ln((x1+x2)/2)).

The pvalue, critical and cdf functions take, as their first parameter, a one-character code for the distribution of interest:

- z or N: Normal;
- t: Student's t;
- X: chi-square;
- F: Snedecor;
- G: Gamma (only pvalue and cdf);
- B: Binomial (only pvalue and cdf);
- D: Bivariate normal (only cdf).

The pvalue function takes the same arguments as the pvalue command, but in this context commas should be placed between the arguments. It returns a one-tailed, right-hand p-value, that is, the integral of the given density function from the specified value to plus infinity.

The cdf function also takes the same arguments as the pvalue command, but it returns the complementary value, that is, the integral of the relevant density function from its lower limit (either minus infinity or zero) to the specified value. For the bivariate normal distribution, three arguments must be given: x, y and the correlation coefficient rho, in this order. For additional discussion of these functions see *Manual de Utilização do Gretl*.

The critical function returns the critical value for a specified probability distribution and a specified proportion in the right-hand tail (with specified degrees of freedom where applicable). The last parameter is the right-hand tail proportion. If the first parameter is t or X, a second parameter must give the degrees of freedom. For the F distribution, the second and third parameters must give the numerator and denominator degrees of freedom.

Here are some examples of use of the pvalue and critical functions (spaces between the arguments are optional):

```
genr p1 = pvalue(z, 2.2)
genr p2 = pvalue(X, 3, 5.67)
genr p3 = cdf(D, -1, 1, 0.25)
genr c1 = critical(t, 20, 0.025)
genr c2 = critical(F, 4, 48, 0.05)
```

The functions relating to the standard normal density work thus, for a given argument x: cnorm returns the area under the standard normal density function integrated from minus infinity to x; dnorm returns the standard normal density evaluated at x; and qnorm returns the z such that the area under the standard normal density, integrated from minus infinity to z, equals x.

The fracdiff function takes two arguments: the name of the series and a fraction, in the range -1 to 1.

Besides the operators and functions noted above there are some special uses of genr:

- genr time creates a time trend variable (1,2,3,. . .) called time. genr index does the same thing except that the variable is called `index`.
- genr dummy creates dummy variables up to the periodicity of the data. For example, in the case of quarterly data (periodicity 4), the program creates `dummy_1` $= 1$ for first quarter and 0 in other quarters, `dummy_2` $= 1$ for the second quarter and 0 in other quarters, and so on.
- genr unitdum and genr timedum create sets of special dummy variables for use with panel data. The first codes for the cross-sectional units and the second for the time period of the observations.

Various internal variables defined in the course of running a regression can be retrieved using genr, as follows:

- `$ess`: error sum of squares
- `$rsq`: unadjusted R-squared
- `$T`: number of observations used
- `$df`: degrees of freedom
- `$ncoeff`: total number of estimated coefficients
- `$trsq`: TR-squared (sample size times R-squared)
- `$sigma`: standard error of residuals
- `$aic`: Akaike Information Criterion
- `$bic`: Schwarz's Bayesian Information Criterion
- `$hqc`: Hannan–Quinn Criterion
- `$lnl`: log-likelihood (where applicable)
- `$coeff`(*var*): estimated coefficient for variable *var*
- `$stderr`(*var*): estimated standard error for variable *var*
- `$rho`(i): ith order autoregressive coefficient for residuals
- `$vcv`(*x1*,*x2*): estimated covariance between coefficients for the variables *x1* and *x2*

Note: In the command-line program, genr commands that retrieve model-related data always reference the model that was estimated most recently. This is also true in the GUI program, if one uses genr in the "gretl console" or enters a formula using the "Define new variable" option under the Variable menu in the main window. With the GUI, however, you have the option of retrieving data from any model currently displayed in a window (whether or not it's the most recent model). You do this under the "Model data" menu in the model's window.

The internal series `$uhat` and `$yhat` hold, respectively, the residuals and fitted values from the last regression. For GARCH models, the conditional variance is held in the internal variable `$h`.

Three "internal" dataset variables are available: `$nobs` holds the number of observations in the current sample range (which may or may not equal the value of `$T`, the number of observations used in estimating the last model); `$nvars` holds the number of variables in the dataset (including the constant); and `$pd` holds the frequency or periodicity of the data (e.g. 4 for quarterly data).

Two special internal scalars, `$test` and `$pvalue`, hold respectively the value and the p-value of the test statistic that was generated by the last explicit hypothesis-testing command, if any (e.g. `chow`). Please see *Manual de Utilização do Gretl* for details on this.

The variable `t` serves as an index of the observations. For instance `genr dum = (t=15)` will generate a dummy variable that has value 1 for observation 15, 0 otherwise. The variable `obs` is similar but more flexible: you can use this to pick out particular observations by date or name. For example, `genr d = (obs>1986:4)` or `genr d = (obs="CA")`. The last form presumes that the observations are labeled; the label must be put in double quotes.

Scalar values can be pulled from a series in the context of a `genr` formula, using the syntax *varname*`[`*obs*`]`. The *obs* value can be given by number or date. Examples: `x[5]`, `CPI[1996:01]`. For daily data, the form *YYYY/MM/DD* should be used, e.g. `ibm[1970/01/23]`.

An individual observation in a series can be modified via `genr`. To do this, a valid observation number or date, in square brackets, must be appended to the name of the variable on the left-hand side of the formula. For example, `genr x[3] = 30` or `genr x[1950:04] = 303.7`.

Here are a couple of tips on dummy variables:

- Suppose `x` is coded with values 1, 2, or 3 and you want three dummy variables, `d1` = 1 if `x` = 1, 0 otherwise, `d2` = 1 if `x` = 2, and so on. To create these, use the commands:

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

- To create `z` = `max(x,y)` do

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

Caminho de Menu: /Variável/Define new variable

Acesso alternativo: Menu de contexto da janela principal

gmm

Opções:	`--two-step` (two step estimation)
	`--iterate` (iterated GMM)
	`--vcv` (print covariance matrix)
	`--verbose` (mostrar detalhes das iterações)

Performs Generalized Method of Moments (GMM) estimation using the BFGS (Broyden, Fletcher, Goldfarb, Shanno) algorithm. You must specify one or more commands for updating the relevant quantities (tyically, GMM residuals), one or more sets of orthogonality conditions, an initial matrix of weights, and a listing of the parameters to be estimated, all enclosed between the tags `gmm` and `end gmm`.

Please see *Manual de Utilização do Gretl* for details on this command. Here we just illustrate with a simple example.

Tabela 1.1: Examples of use of genr command

Formula	*Comment*
`y = x1^3`	`x1` cubed
`y = ln((x1+x2)/x3)`	
`z = x>y`	`z(t)` = 1 if `x(t) > y(t)`, otherwise 0
`y = x(-2)`	`x` lagged 2 periods
`y = x(+2)`	`x` led 2 periods
`y = diff(x)`	`y(t) = x(t) - x(t-1)`
`y = ldiff(x)`	`y(t) = log x(t) - log x(t-1)`, the instantaneous rate of growth of `x`
`y = sort(x)`	sorts `x` in increasing order and stores in `y`
`y = dsort(x)`	sort `x` in decreasing order
`y = int(x)`	truncate `x` and store its integer value as `y`
`y = abs(x)`	store the absolute values of `x`
`y = sum(x)`	sum `x` values excluding missing −999 entries
`y = cum(x)`	cumulation: $y_t = \sum_{\tau=1}^{t} x_\tau$
`aa = $ess`	set `aa` equal to the Error Sum of Squares from last regression
`x = $coeff(sqft)`	grab the estimated coefficient on the variable `sqft` from the last regression
`rho4 = $rho(4)`	grab the 4th-order autoregressive coefficient from the last model (presumes an `ar` model)
`cvx1x2 = $vcv(x1, x2)`	grab the estimated coefficient covariance of vars `x1` and `x2` from the last model
`foo = uniform()`	uniform pseudo-random variable in range 0–1
`bar = 3 * normal()`	normal pseudo-random variable, $\mu = 0$, $\sigma = 3$
`samp = ok(x)`	= 1 for observations where `x` is not missing.

```
gmm e = y - X*b
  orthog e ; W
  weights V
  params b
end gmm
```

In the example above we assume that `y` and `X` are data matrices, `b` is an appropriately sized vector of parameter values, `W` is a matrix of instruments, and `V` is a suitable matrix of weights. The statement

```
orthog e ; W
```

indicates that the residual vector `e` is in principle orthogonal to each of the instruments composing the columns of `W`.

Caminho de Menu: /Model/GMM

gnuplot

Argumentos:	*yvars xvar* [*dumvar*]
Opções:	`--with-lines` (use lines, not points)
	`--with-impulses` (use vertical lines)
	`--suppress-fit` (don't show fitted line)
	`--linear-fit` (show least squares fit)
	`--inverse-fit` (show inverse fit)
	`--quadratic-fit` (show quadratic fit)
	`--loess-fit` (show loess fit)
	`--dummy` (see below)
Exemplos:	`gnuplot y1 y2 x`
	`gnuplot x time --with-lines`
	`gnuplot wages educ gender --dummy`

Without the `--dummy` option, the *yvars* are graphed against *xvar*. With `--dummy`, *yvar* is graphed against *xvar* with the points shown in different colors depending on whether the value of *dumvar* is 1 or 0.

The `time` variable behaves specially: if it does not already exist then it will be generated automatically.

In interactive mode the result is displayed immediately. In batch mode a gnuplot command file is written, with a name on the pattern `gpttmpN.plt`, starting with N = `01`. The actual plots may be generated later using gnuplot (under MS Windows, wgnuplot).

The various "fit" options are applicable only in the case of a bivariate scatterplot. The default behavior is to show the OLS fitted line if and only if the slope coefficient is significant at the 10 percent level. If the `suppress` option is given, no fitted line is shown. If the `linear` option is given, the OLS line is shown regardless of whether or not it is significant. The other options—`inverse`, `quadratic` and loess—produce respectively an inverse fit (regression of y on 1/x), a quadratic fit, or a loess fit. Loess (also sometimes called "lowess") is a robust locally weighted regression.

A further option to this command is available: following the specification of the variables to be plotted and the option flag (if any), you may add literal gnuplot commands to control the appearance of the plot (for example, setting the plot title and/or the axis ranges). These commands should be enclosed in braces, and each gnuplot command must be terminated with a semi-colon. A backslash may be used to continue a set of gnuplot commands over more than one line. Here is an example of the syntax:

```
{ set title 'My Title'; set yrange [0:1000]; }
```

Caminho de Menu: /View/Graph specified vars

Acesso alternativo: Menu de contexto da janela principal, graph button on toolbar

graph

Argumentos:	*yvars xvar*
Opção:	`--tall` (use 40 rows)

ASCII graphics. The *yvars* (which may be given by name or number) are graphed against *xvar* using ASCII symbols. The `--tall` flag will produce a graph with 40 rows and 60 columns. Without it, the graph will be 20 by 60 (for screen output). See also gnuplot.

hausman

This test is available only after estimating an OLS model using panel data (see also setobs). It tests the simple pooled model against the principal alternatives, the fixed effects and random effects models.

The fixed effects model allows the intercept of the regression to vary across the cross-sectional units. An F-test is reported for the null hypotheses that the intercepts do not differ. The random effects model decomposes the residual variance into two parts, one part specific to the cross-sectional unit and the other specific to the particular observation. (This estimator can be computed only if the number of cross-sectional units in the data set exceeds the number of parameters to be estimated.) The Breusch–Pagan LM statistic tests the null hypothesis that the pooled OLS estimator is adequate against the random effects alternative.

The pooled OLS model may be rejected against both of the alternatives, fixed effects and random effects. Provided the unit- or group-specific error is uncorrelated with the independent variables, the random effects estimator is more efficient than the fixed effects estimator; otherwise the random effects estimator is inconsistent and the fixed effects estimator is to be preferred. The null hypothesis for the Hausman test is that the group-specific error is not so correlated (and therefore the random effects model is preferable). A low p-value for this test counts against the random effects model and in favor of fixed effects.

Caminho de Menu: Janela do modelo, /Testes/Panel diagnostics

hccm

Argumentos:	*variável-dependente variáveis-independentes*
Opção:	`--vcv` (mostrar matriz de covariância)

Heteroskedasticity-Consistent Covariance Matrix: this command runs a regression where the coefficients are estimated via the standard OLS procedure, but the standard errors of the coefficient estimates are computed in a manner that is robust in the face of heteroskedasticity, namely using the MacKinnon–White "jackknife" procedure.

heckit

Argumentos:	*variável-dependente variáveis-independentes* ; *selection equation*
Opções:	`--two-step` (perform two-step estimation)
	`--vcv` (print covariance matrix)
	`--verbose` (print extra output)
Exemplo:	`heckit y 0 x1 x2 ; ys 0 x3 x4`

Heckman-type selection model. In the specification, the list before the semicolon represents the outcome equation, and the second list represents the selection equation. The dependent variable in the selection equation (`ys` in the example above) must be a binary variable.

By default, the parameters are estimated by maximum likelihood. The covariance matrix of the parameters is computed using the negative inverse of the Hessian. If two-step estimation is desired, use the `--two-step` option. In this case, the covariance matrix of the parameters of the outcome equation is appropriately adjusted as per Heckman (1979).

FIXME this entry needs to be completed.

Caminho de Menu: /Model/Nonlinear models/Heckit

help

Gives a list of available commands. help *command* describes *command* (e.g. help smpl). You can type man instead of help if you like.

Caminho de Menu: /Help

hsk

Argumentos:	*variável-dependente variáveis-independentes*
Opção:	`--vcv` (mostrar matriz de covariância)

An OLS regression is run and the residuals are saved. The logs of the squares of these residuals then become the dependent variable in an auxiliary regression, on the right-hand side of which are the original independent variables plus their squares. The fitted values from the auxiliary regression are then used to construct a weight series, and the original model is re-estimated using weighted least squares. This final result is reported.

The weight series is formed as $1/\sqrt{e^{y^*}}$, where y* denotes the fitted values from the auxiliary regression.

Caminho de Menu: /Model/Other linear models/Heteroskedasticity corrected

hurst

Argumento:	*nome-de-variável*

Calculates the Hurst exponent (a measure of persistence or long memory) for a time-series variable having at least 128 observations.

The Hurst exponent is discussed by Mandelbrot. In theoretical terms it is the exponent, H, in the relationship

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

where RS is the "rescaled range" of the variable x in samples of size n and a is a constant. The rescaled range is the range (maximum minus minimum) of the cumulated value or partial sum of x over the sample period (after subtraction of the sample mean), divided by the sample standard deviation.

As a reference point, if x is white noise (zero mean, zero persistence) then the range of its cumulated "wandering" (which forms a random walk), scaled by the standard deviation, grows as the square root of the sample size, giving an expected Hurst exponent of 0.5. Values of the exponent significantly in excess of 0.5 indicate persistence, and values less than 0.5 indicate anti-persistence (negative autocorrelation). In principle the exponent is bounded by 0 and 1, although in finite samples it is possible to get an estimated exponent greater than 1.

In gretl, the exponent is estimated using binary sub-sampling: we start with the entire data range, then the two halves of the range, then the four quarters, and so on. For sample sizes smaller than the data range, the RS value is the mean across the available samples. The exponent is then estimated as the slope coefficient in a regression of the log of RS on the log of sample size.

Caminho de Menu: /Variável/Hurst exponent

if

Flow control for command execution. Three sorts of construction are supported, as follows.

```
# simple form
if condition
   commands
endif
```

```
# two branches
if condition
   commands1
else
   commands2
endif
```

```
# three or more branches
if condition1
   commands1
elif condition2
   commands2
else
   commands3
endif
```

condition must be a Boolean expression, for the syntax of which see genr. More than one elif block may be included. In addition, `if` ... `endif` blocks may be nested.

include

Argumento:	*inputfile*
Exemplos:	`include myfile.inp`
	`include sols.gfn`

Intended for use in a command script, primarily for including definitions of functions. Executes the commands in *inputfile* then returns control to the main script. To include a packaged function, be sure to include the filename extension.

See also run.

info

Prints out any supplementary information stored with the current datafile.

Caminho de Menu: /Data/Read info

Acesso alternativo: Data browser windows

kpss

Argumentos:	*order varlist*
Opções:	`--trend` (include a trend)
	`--verbose` (print regression results)
	`--quiet` (suppress printing of results)
	`--difference` (use first difference of variable)
Exemplos:	`kpss 8 y`
	`kpss 4 x1 --trend`

Computes the KPSS test (Kwiatkowski, Phillips, Schmidt and Shin, 1992) for stationarity, for each of the specified variables (or their first difference, if the `--difference` option is selected). The null hypothesis is that the variable in question is stationary, either around a level or, if the `--trend` option is given, around a deterministic linear trend.

The order argument determines the size of the window used for Bartlett smoothing. If the `--verbose` option is chosen the results of the auxiliary regression are printed, along with the estimated variance of the random walk component of the variable.

Caminho de Menu: /Variable/KPSS test

labels

Prints out the informative labels for any variables that have been generated using genr, and any labels added to the data set via the GUI.

lad

Argumentos:	*depvar indepvars*
Opção:	`--vcv` (print covariance matrix)

Calculates a regression that minimizes the sum of the absolute deviations of the observed from the fitted values of the dependent variable. Coefficient estimates are derived using the Barrodale–Roberts simplex algorithm; a warning is printed if the solution is not unique.

Standard errors are derived using the bootstrap procedure with 500 drawings. The covariance matrix for the parameter estimates, printed when the `--vcv` flag is given, is based on the same bootstrap.

Caminho de Menu: /Model/Robust estimation/Least Absolute Deviation

lags

Variantes:	`lags` *varlist*
	`lags` *order* `;` *varlist*
Exemplos:	`lags x y`
	`lags 12 ; x y`

Creates new variables which are lagged values of each of the variables in *varlist*. By default the number of lagged variables equals the periodicity of the data. For example, if the periodicity is 4 (quarterly), the command lags x creates

```
x_1 = x(t-1)
x_2 = x(t-2)
x_3 = x(t-3)
x_4 = x(t-4)
```

The number of lags created can be controlled by the optional first parameter.

Caminho de Menu: /Add/Lags of selected variables

ldiff

Argumento:	*varlist*

The first difference of the natural log of each variable in *varlist* is obtained and the result stored in a new variable with the prefix `ld_`. Thus ldiff x y creates the new variables

```
ld_x = log(x) - log(x(-1))
ld_y = log(y) - log(y(-1))
```

Caminho de Menu: /Add/Log differences of selected variables

leverage

Opção: `--save` (save variables)

Must immediately follow an ols command. Calculates the leverage (h, which must lie in the range 0 to 1) for each data point in the sample on which the previous model was estimated. Displays the residual (u) for each observation along with its leverage and a measure of its influence on the estimates, $uh/(1-h)$. "Leverage points" for which the value of h exceeds $2k/n$ (where k is the number of parameters being estimated and n is the sample size) are flagged with an asterisk. For details on the concepts of leverage and influence see Davidson and MacKinnon (1993, Chapter 2).

DFFITS values are also shown: these are "studentized residuals" (predicted residuals divided by their standard errors) multiplied by $\sqrt{h/(1-h)}$. For a discussion of studentized residuals and DFFITS see G. S. Maddala, *Introduction to Econometrics*, chapter 12; also Belsley, Kuh and Welsch (1980).

Briefly, a "predicted residual" is the difference between the observed value of the dependent variable at observation t, and the fitted value for observation t obtained from a regression in which that observation is omitted (or a dummy variable with value 1 for observation t alone has been added); the studentized residual is obtained by dividing the predicted residual by its standard error.

If the `--save` flag is given with this command, then the leverage, influence and DFFITS values are added to the current data set.

Caminho de Menu: Model window, /Tests/Influential observations

lmtest

Argumento:	`[` *order* `]`
Opções:	`--logs` (non-linearity, logs)
	`--autocorr` (serial correlation)
	`--arch` (ARCH)
	`--squares` (non-linearity, squares)
	`--white` (heteroskedasticity, White's test)
	`--panel` (heteroskedasticity, groupwise)
	`--quiet` (don't print auxiliary regression)

Must immediately follow an ols command. Depending on the options given, this command carries out some combination of the following: Lagrange Multiplier tests for nonlinearity (logs and squares); White's test for heteroskedasticity; the LMF test for serial correlation (see Kiviet, 1986); and a test for ARCH (Autoregressive Conditional Heteroskedasticity, see also the arch command).

The optional `order` argument is relevant only in case the `--autocorr` or `--arch` options are selected. The default is to run these tests using a lag order equal to the periodicity of the data, but this can be adjusted by supplying a specific lag order.

The `--panel` option is available only when the model is estimated on panel data: in this case a test for groupwise heteroskedasticity is performed (that is, for a differing error variance across the cross-sectional units).

By default, the program prints the auxiliary regression on which the test statistic is based. This may be suppressed by using the `--quiet` flag.

Caminho de Menu: Janela do modelo, /Tests

logistic

Argumentos:	*variável-dependente variáveis-independentes* `[ ymax=`*value* `]`
Opção:	`--vcv` (mostrar matriz de covariância)
Exemplos:	`logistic y const x`
	`logistic y const x ymax=50`

Logistic regression: carries out an OLS regression using the logistic transformation of the dependent variable,

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

The dependent variable must be strictly positive. If it is a decimal fraction, between 0 and 1, the default is to use a y* value (the asymptotic maximum of the dependent variable) of 1. If the dependent variable is a percentage, between 0 and 100, the default y* is 100.

If you wish to set a different maximum, use the optional `ymax=`*value* syntax following the list of regressors. The supplied value must be greater than all of the observed values of the dependent variable.

The fitted values and residuals from the regression are automatically transformed using

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

where x represents either a fitted value or a residual from the OLS regression using the transformed dependent variable. The reported values are therefore comparable with the original dependent variable.

Note that if the dependent variable is binary, you should use the logit command instead.

Caminho de Menu: /Model/Nonlinear models/Logistic

logit

Argumentos:	*variável-dependente variáveis-independentes*
Opções:	`--robust` (robust standard errors)
	`--vcv` (mostrar matriz de covariância)
	`--verbose` (mostrar detalhes das iterações)

If the dependent variable is a binary variable (all values are 0 or 1) maximum likelihood estimates of the coefficients on *variáveis-independentes* are obtained via the "binary response model regression" (BRMR) method outlined by Davidson and MacKinnon (2004). As the model is nonlinear the slopes depend on the values of the independent variables: the reported slopes are evaluated at the means of those variables. The chi-square statistic tests the null hypothesis that all coefficients are zero apart from the constant.

By default, standard errors are computed using the negative inverse of the Hessian. If the `--robust` flag is given, then QML or Huber–White standard errors are calculated instead. In this case the estimated covariance matrix is a "sandwich" of the inverse of the estimated Hessian and the outer product of the gradient. See Davidson and MacKinnon (2004, Chapter 10) for details.

If the dependent variable is not binary, but is discrete and has a minimum value of 0, then Ordered Logit estimates are obtained. In this case robust standard errors are not yet available. (If the variable selected as dependent is not discrete, or does not have a minimum of zero, an error is flagged.)

If you want to use logit for analysis of proportions (where the dependent variable is the proportion of cases having a certain characteristic, at each observation, rather than a 1 or 0 variable indicating whether the characteristic is present or not) you should not use the logit command, but rather construct the logit variable, as in

```
        genr lgt_p = log(p/(1 - p))
```

and use this as the dependent variable in an OLS regression. See Ramanathan (2002, Chapter 12).

Caminho de Menu: /Model/Nonlinear models/Logit

logs

Argumento:	*lista-de-variáveis*

The natural log of each of the variables in *lista-de-variáveis* is obtained and the result stored in a new variable with the prefix `l_` ("el" underscore). For example, logs x y creates the new variables `l_x` = ln(x) and `l_y` = ln(y).

Caminho de Menu: /Add/Logs of selected variables

loop

Argumento:	*control*
Opções:	`--progressive` (enable special forms of certain commands)
	`--verbose` (report details of genr commands)
	`--quiet` (do not report number of iterations performed)
Exemplos:	`loop 1000`
	`loop 1000 --progressive`
	`loop while essdiff > .00001`
	`loop for i=1991..2000`
	`loop for (r=-.99; r<=.99; r+=.01)`

The parameter *control* must take one of four forms, as shown in the examples: an integer number of times to repeat the commands within the loop; "`while`" plus a numerical condition; "`for`" plus a range of values for the internal index variable `i`; or "`for`" plus three expressions in parentheses, separated by semicolons. In the last form the left-hand expression initializes a variable, the middle expression sets a condition for iteration to continue, and the right-hand expression sets an increment or decrement to be applied at the start of the second and subsequent iterations. (This is a restricted form of the `for` statement in the C programming language.)

This command opens a special mode in which the program accepts commands to be executed repeatedly. You exit the mode of entering loop commands with endloop: at this point the stacked commands are executed.

See *Manual de Utilização do Gretl* for further details and examples. The effect of the `--progressive` option (which is designed for use in Monte Carlo simulations) is explained there. Not all gretl commands may be used within a loop; the commands available in this context are also set out there.

mahal

Argumento:	*lista-de-variáveis*
Opções:	`--save` (add distances to the dataset)
	`--vcv` (mostrar matriz de covariância)

The Mahalanobis distance is the distance between two points in an k-dimensional space, scaled by the statistical variation in each dimension of the space. For example, if p and q are two observations on a set of k variables with covariance matrix C, then the Mahalanobis distance between the observations is given by

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

where $(p-q)$ is a k-vector. This reduces to Euclidean distance if the covariance matrix is the identity matrix.

The space for which distances are computed is defined by the selected variables. For each observation in the current sample range, the distance is computed between the observation and the centroid of the selected variables. This distance is the multidimensional counterpart of a standard z-score, and can be used to judge whether a given observation "belongs" with a group of other observations.

If the `--vcv` option is given, the covariance matrix and its inverse are printed. If the `--save` option is given, the distances are saved to the dataset under the name `mdist` (or `mdist1`, `mdist2` and so on if there is already a variable of that name).

Caminho de Menu: /View/Mahalanobis distances

meantest

Argumentos:	*var1 var2*
Opção:	`--unequal-vars` (assume variances are unequal)

Calculates the t statistic for the null hypothesis that the population means are equal for the variables *var1* and *var2*, and shows its p-value.

By default the test statistic is calculated on the assumption that the variances are equal for the two variables; with the `--unequal-vars` option the variances are assumed to be different. This will make a difference to the test statistic only if there are different numbers of non-missing observations for the two variables.

Caminho de Menu: /Model/Bivariate tests/Difference of means

mle

Argumentos:	*log-likelihood function derivatives*
Opções:	`--vcv` (mostrar matriz de covariância)
	`--hessian` (base covariance matrix on the Hessian)
	`--robust` (QML covariance matrix)
	`--verbose` (mostrar detalhes das iterações)
Exemplo:	`weibull.inp`

Performs Maximum Likelihood (ML) estimation using the BFGS (Broyden, Fletcher, Goldfarb, Shanno) algorithm. The user must specify the log-likelihood function. The parameters of this function must be declared and given starting values (using the genr command) prior to estimation. Optionally, the user may specify the derivatives of the log-likelihood function with respect to each of the parameters; if analytical derivatives are not supplied, a numerical approximation is computed.

Simple example: Suppose we have a series `X` with values 0 or 1 and we wish to obtain the maximum likelihood estimate of the probability, `p`, that `X` = 1. (In this simple case we can guess in advance that the ML estimate of `p` will simply equal the proportion of Xs equal to 1 in the sample.)

The parameter `p` must first be added to the dataset and given an initial value. This can be done using the genr command. For example, `genr p = 0.5`.

We then construct the MLE command block:

```
mle loglik = X*log(p) + (1-X)*log(1-p)
  deriv p = X/p - (1-X)/(1-p)
end mle
```

The first line above specifies the log-likelihood function. It starts with the keyword `mle`, then a dependent variable is specified and an expression for the log-likelihood is given (using the same syntax as in the genr command). The next line (which is optional) starts with the keyword `deriv` and supplies the derivative of the log-likelihood function with respect to the parameter `p`. If no derivatives are given, you should include a statement using the keyword `params` which identifies the free parameters: these are listed on one line, separated by spaces. For example, the above could be changed to:

```
mle loglik = X*log(p) + (1-X)*log(1-p)
  params p
end mle
```

in which case numerical derivatives would be used.

Note that any option flags should be appended to the ending line of the MLE block.

By default, estimated standard errors are based on the Outer Product of the Gradient. If the `--hessian` option is given, they are instead based on the negative inverse of the Hessian (which is approximated numerically). If the `--robust` option is given, a QML estimator is used (namely, a sandwich of the negative inverse of the Hessian and the covariance matrix of the gradient).

Caminho de Menu: /Model/Maximum likelihood

modeltab

Argumentos:	*add* ou *show* ou *free*

Manipulates the gretl "model table". See *Manual de Utilização do Gretl* for details. The sub-commands have the following effects: add adds the last model estimated to the model table, if possible; show displays the model table in a window; and free clears the table.

Caminho de Menu: Session window, Model table icon

mpols

Argumentos:	*variável-dependente variáveis-independentes*
Opções:	`--vcv` (mostrar matriz de covariância)
	`--simple-print` (do not print auxiliary statistics)
	`--quiet` (não mostrar resultados da regressão)

Computes OLS estimates for the specified model using multiple precision floating-point arithmetic. This command is available only if gretl is compiled with support for the Gnu Multiple Precision (GMP) library. By default 256 bits of precision are used for the calculations, but this can be increased via the environment variable `GRETL_MP_BITS`. For example, when using the bash shell one could issue the following command, before starting gretl, to set a precision of 1024 bits.

```
export GRETL_MP_BITS=1024
```

A rather arcane option is available for this command (primarily for testing purposes): if the *variáveis-independentes* list is followed by a semicolon and a further list of numbers, those numbers are taken as powers of *x* to be added to the regression, where *x* is the last variable in *variáveis-independentes*. These additional terms are computed and stored in multiple precision. In the following example `y` is regressed on `x` and the second, third and fourth powers of `x`:

```
mpols y 0 x ; 2 3 4
```

Caminho de Menu: /Model/Other linear models/High precision OLS

nls

Argumentos:	*function* [*derivatives*]
Opções:	`--robust` (robust standard errors)
	`--vcv` (mostrar matriz de covariância)
	`--verbose` (mostrar detalhes das iterações)
Exemplo:	`wg_nls.inp`

Performs Nonlinear Least Squares (NLS) estimation using a modified version of the Levenberg–Marquandt algorithm. You must supply a function specification. The parameters of this function must be declared and given starting values (using the genr command) prior to estimation. Optionally, you may specify the derivatives of the regression function with respect to each of the parameters. If you do not supply derivatives you should instead give a list of the parameters to be estimated (separated by spaces or commas), preceded by the keyword `params`. In the latter case a numerical approximation to the Jacobian is computed.

It is easiest to show what is required by example. The following is a complete script to estimate the nonlinear consumption function set out in William Greene's *Econometric Analysis* (Chapter 11 of the 4th edition, or Chapter 9 of the 5th). The numbers to the left of the lines are for reference and are not part of the commands. Note that any option flags, such as `--vcv` for printing the covariance matrix of the parameter estimates, should be appended to the final command, `end nls`.

```
1   open greene11_3.gdt
2   ols C 0 Y
3   genr a = $coeff(0)
4   genr b = $coeff(Y)
```

```
5   genr g = 1.0
6   nls C = a + b * Y^g
7    deriv a = 1
8    deriv b = Y^g
9    deriv g = b * Y^g * log(Y)
10  end nls --vcv
```

It is often convenient to initialize the parameters by reference to a related linear model; that is accomplished here on lines 2 to 5. The parameters alpha, beta and gamma could be set to any initial values (not necessarily based on a model estimated with OLS), although convergence of the NLS procedure is not guaranteed for an arbitrary starting point.

The actual NLS commands occupy lines 6 to 10. On line 6 the nls command is given: a dependent variable is specified, followed by an equals sign, followed by a function specification. The syntax for the expression on the right is the same as that for the genr command. The next three lines specify the derivatives of the regression function with respect to each of the parameters in turn. Each line begins with the keyword deriv, gives the name of a parameter, an equals sign, and an expression whereby the derivative can be calculated (again, the syntax here is the same as for genr). As an alternative to supplying numerical derivatives, you could substitute the following for lines 7 to 9:

```
params a b g
```

Line 10, end nls, completes the command and calls for estimation.

For further details on NLS estimation please see *Manual de Utilização do Gretl*.

Caminho de Menu: /Model/Nonlinear models/Nonlinear Least Squares

normtest

Argumento:	*series*
Opções:	`--dhansen` (Doornik–Hansen test)
	`--swilk` (Shapiro–Wilk test)
	`--lillie` (Lilliefors test)
	`--jbera` (Jarque–Bera test)
	`--all` (do all tests)
	`--quiet` (suppress printed output)

Carries out a test for normality for the given *series*. The specific test is controlled by the option flags (but if no flag is given, the Doornik–Hansen test is performed). Note: the Doornik–Hansen and Shapiro–Wilk tests are recommended over the others, on account of their superior small-sample properties.

The test statistic and its p-value may be retrieved using the accessors `$test` and `$pvalue`. Please note that if the `--all` option is given, the result recorded is that from the Doornik–Hansen test.

Caminho de Menu: /Variable/Normality test

nulldata

Argumento:	*series-length*
Exemplo:	`nulldata 500`

Establishes a "blank" data set, containing only a constant and an index variable, with periodicity 1 and the specified number of observations. This may be used for simulation purposes: some of the genr commands (e.g. genr uniform(), genr normal()) will generate dummy data from scratch to fill out the data set. This command may be useful in conjunction with loop. See also the "seed" option to the set command.

Caminho de Menu: /Ficheiro/New data set

ols

Argumentos:	*variável-dependente variáveis-independentes*
Opções:	`--vcv` (mostrar matriz de covariância)
	`--robust` (robust standard errors)
	`--simple-print` (do not print auxiliary statistics)
	`--quiet` (não mostrar resultados da regressão)
	`--no-df-corr` (suppress degrees of freedom correction)
	`--print-final` (see below)
Exemplos:	`ols 1 0 2 4 6 7`
	`ols y 0 x1 x2 x3 --vcv`
	`ols y 0 x1 x2 x3 --quiet`

Computes ordinary least squares (OLS) estimates with *depvar* as the dependent variable and *variáveis-independentes* as the list of independent variables. Variables may be specified by name or number; use the number zero for a constant term.

Besides coefficient estimates and standard errors, the program also prints p-values for t (two-tailed) and F-statistics. A p-value below 0.01 indicates statistical significance at the 1 percent level and is marked with `***`. `**` indicates significance between 1 and 5 percent and `*` indicates significance between the 5 and 10 percent levels. Model selection statistics (the Akaike Information Criterion or AIC and Schwarz's Bayesian Information Criterion) are also printed. The formula used for the AIC is that given by Akaike (1974), namely minus two times the maximized log-likelihood plus two times the number of parameters estimated.

If the option `--no-df-corr` is given, the usual degrees of freedom correction is not applied when calculating the estimated error variance (and hence also the standard errors of the parameter estimates).

The option `--print-final` is applicable only in the context of a loop. It arranges for the regression to be run silently on all but the final iteration of the loop. See *Manual de Utilização do Gretl* for details.

Various internal variables may be retrieved using the genr command, provided genr is invoked immediately after this command.

The specific formula used for generating robust standard errors (when the `--robust` option is given) can be adjusted via the set command.

Caminho de Menu: /Model/Ordinary Least Squares

Acesso alternativo: Beta-hat button on toolbar

omit

Argumento:	*lista-de-variáveis*
Opções:	`--wald` (do a Wald test rather than an F-test)
	`--vcv` (mostrar matriz de covariância)
	`--quiet` (don't print estimates for reduced model)
	`--silent` (don't print anything)
	`--inst` (omit as instrument, TSLS only)
	`--both` (omit as both regressor and instrument, TSLS only)
Exemplos:	`omit 5 7 9`
	`omit seasonals --quiet`

This command must follow an estimation command. It calculates a test statistic for the joint significance of the variables in *lista-de-variáveis*, which should be a subset of the independent variables in the model previously estimated.

If the original model was estimated by OLS, the test statistic is by default an F-value. This is based on the sums of squared residuals for the restricted and unrestricted models, unless the original model was estimated with robust standard errors. In the latter case F is computed from the robust estimate of the covariance matrix for the original model. (It is the F-form of a Wald test).

For estimators other than OLS, or if the `--wald` option is given, the statistic is an asymptotic Wald chi-square value based on the covariance matrix of the original model.

By default, the restricted model is estimated, the estimates are printed, and the restricted model replaces the original as the "current model" for the purposes of, for example, retrieving the residuals as `$uhat` (or doing further tests such as add or omit).

If the `--wald` option is selected, the restricted model is not estimated (and so the current model is not replaced). The `--quiet` option suppresses the printout of the restricted model (if applicable): only the result of the test is printed. If the restricted model is both estimated and printed, the `--vcv` option has the effect of printing the covariance matrix for the coefficients in the restricted model, otherwise this option is ignored.

If the `--silent` option is given, nothing is printed; nonetheless, the results of the test can be retrieved using the special variables `$test` and `$pvalue`.

If the original model was estimated using two-stage least squares, an ambiguity arises: should the selected variables be omitted as regressors, as instruments, or as both? This is resolved as follows: by default the variables are dropped from the list of regressors, but if the `--inst` flag is given they are dropped as instruments, or if the `--both` flag is given they are dropped from the model altogether. These two options are incompatible with the `--wald` option; if one or more instruments are omitted the model must be re-estimated.

Caminho de Menu: Janela do modelo, /Testes/Omit variables

open

Argumento:	*ficheiro-de-dados*
Opção:	`--www` (use a database on the gretl server)
Exemplos:	`open data4-1`
	`open voter.dta`
	`open fedbog --www`

Opens a data file. If a data file is already open, it is replaced by the newly opened one. If a full path is not given, the program will search some relevant paths to try to find the file. If no filename suffix is given (as in the first example above), gretl assumes a native datafile with suffix `.gdt`. Based on the name of the file and various heuristics, gretl will try to detect the format of the data file (native, plain text, CSV, MS Excel, Stata, etc.).

This command can also be used to open a database (gretl or RATS 4.0) for reading. In that case it should be followed by the data command to extract particular series from the database. If the `www` option is given, the program will try to access a database of the given name on the gretl server — for instance the Federal Reserve interest rates database in the third example above.

Caminho de Menu: /Ficheiro/Open data

Acesso alternativo: Drag a data file into gretl (MS Windows or Gnome)

orthdev

Argumento:	*varlist*

Applicable with panel data only. A series of forward orthogonal deviations is obtained for each variable in *varlist* and stored in a new variable with the prefix `o_`. Thus orthdev x y creates the new variables `o_x` and `o_y`.

The values are stored one step ahead of their true temporal location (that is, `o_x` at observation t holds the deviation that, strictly speaking, belongs at $t - 1$). This is for compatibility with first differences: one loses the first observation in each time series, not the last.

outfile

Argumentos:	*filename option*
Opções:	`--append` (append to file)
	`--close` (close file)
	`--write` (overwrite file)
Exemplos:	`outfile --write regress.txt`
	`outfile --close`

Diverts output to *filename*, until further notice. Use the flag `--append` to append output to an existing file or `--write` to start a new file (or overwrite an existing one). Only one file can be opened in this way at any given time.

The `--close` flag is used to close an output file that was previously opened as above. Output will then revert to the default stream.

In the first example command above, the file `regress.txt` is opened for writing, and in the second it is closed. This would make sense as a sequence only if some commands were issued before the `--close`. For example if an ols command intervened, its output would go to `regress.txt` rather than the screen.

panel

Opções:	`--vcv` (mostrar matriz de covariância)
	`--random-effects` (random rather than fixed effects)
	`--between` (estimate the between-groups model)
	`--time-dummies` (include time dummy variables)
	`--unit-weights` (weighted least squares)
	`--iterate` (iterative estimation)
	`--quiet` (less verbose output)
	`--verbose` (more verbose output)

Estimates a panel model, by default using the fixed effects estimator. Depending on the number of cross-sectional units and the number of independent variables, this is implemented either by adding a set of dummy variables representing the units, or by subtracting the group or unit means from the original data.

If the `--random-effects` flag is given, random effects estimates are computed, using the method of Swamy and Arora.

Alternatively, if the `--unit-weights` flag is given, the model is estimated via weighted least squares, with the weights based on the residual variance for the respective cross-sectional units in the sample. In this case (only) the `--iterate` flag may be added to produce iterative estimates: if the iteration converges, the resulting estimates are Maximum Likelihood.

As a further alternative, if the `--between` flag is given, the between-groups model is estimated (that is, an OLS regression using the group means).

For more details on panel estimation, please see *Manual de Utilização do Gretl*.

Caminho de Menu: /Model/Panel

pca

Argumento:	*lista-de-variáveis*
Opções:	`--save` (Save major components)
	`--save-all` (Save all components)

Principal Components Analysis. Prints the eigenvalues of the correlation matrix for the variables in *lista-de-variáveis* along with the proportion of the joint variance accounted for by each component. Also prints the corresponding eigenvectors (or "component loadings").

If the `--save` flag is given, components with eigenvalues greater than 1.0 are saved to the dataset as variables, with names `PC1`, `PC2` and so on. These artificial variables are formed as the sum of (component loading) times (standardized `Xi`), where `Xi` denotes the ith variable in *lista-de-variáveis*.

If the `--save-all` flag is given, all of the components are saved as described above.

Caminho de Menu: /View/Principal components

Acesso alternativo: Main window pop-up (multiple selection)

pergm

Argumentos:	*nome-de-variável* `[` *bandwidth* `]`
Opções:	`--bartlett` (use Bartlett lag window)
	`--log` (use log scale)

Computes and displays (and if not in batch mode, graphs) the spectrum of the specified variable. By default the sample periodogram is given; with the `--bartlett` flag a Bartlett lag window is used in estimating the spectrum (see, for example, Greene's *Econometric Analysis* for a discussion of this). The default width of the Bartlett window is twice the square root of the sample size but this can be set manually using the *bandwidth* parameter, up to a maximum of half the sample size. If the `--log` option is given the spectrum is represented on a logarithmic scale.

When the sample periodogram is printed, two tests for fractional integration of the series ("long memory") are given, namely the Geweke and Porter-Hudak (GPH) test and the Local Whittle Estimator. The null hypothesis in both cases is that the integration order is zero. By default the order for these tests is the lesser of T/2 and T0.6. Again, this value can be adjusted using the bandwidth parameter.

Caminho de Menu: /Variável/Spectrum

Acesso alternativo: Menu de contexto da janela principal (selecção singular)

poisson

Argumentos:	*variável-dependente variáveis-independentes* `[ ;` *offset* `]`
Opções:	`--vcv` (mostrar matriz de covariância)
	`--verbose` (mostrar detalhes das iterações)
Exemplos:	`poisson y 0 x1 x2`
	`poisson y 0 x1 x2 ; S`

Estimates a poisson regression. The dependent variable is taken to represent the occurrence of events of some sort, and must take on only non-negative integer values.

If a discrete random variable Y follows the Poisson distribution, then

$$\Pr(Y = y) = \frac{e^{-v}v^y}{y!}$$

for y = 0, 1, 2,. . . . The mean and variance of the distribution are both equal to v. In the Poisson regression model, the parameter v is represented as a function of one or more independent variables. The most common version (and the only one supported by gretl) has

$$v = \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots)$$

or in other words the log of v is a linear function of the independent variables.

Optionally, you may add an "offset" variable to the specification. This is a scale variable, the log of which is added to the linear regression function (implicitly, with a coefficient of 1.0). This makes sense if you expect the number of occurrences of the event in question to be proportional, other things equal, to some known factor. For example, the number of traffic accidents might be supposed to be proportional to traffic volume, other things equal, and in that case traffic volume could be specified as an "offset" in a Poisson model of the accident rate. The offset variable must be strictly positive.

Caminho de Menu: /Model/Nonlinear models/Poisson

plot

Argumento:	*lista-de-variáveis*
Opção:	`--one-scale` (force a single scale)

Plots the values for specified variables, for the range of observations currently in effect, using ASCII symbols. Each line stands for an observation and the values are plotted horizontally. By default the variables are scaled appropriately. See also gnuplot.

print

Argumentos:	*lista-de-variáveis* ou *string-literal*
Opções:	`--byobs` (by observations)
	`--long` (use 10 significant digits or more)
	`--no-dates` (use simple observation numbers)
Exemplos:	`print x1 x2 --byobs`
	`print "This is a string"`

If *lista-de-variáveis* is given, prints the values of the specified variables; if no list is given, prints the values of all variables in the current data file. If the `--byobs` flag is given the data are printed by observation, otherwise they are printed by variable.

If the `--long` flag is given the data are printed, by variable, to greater than usual precision. The default in this case is to show 10 significant digits but you can adjust that figure using the set command.

If the `--byobs` flag is given and the data are printed by observation, the default is to show the date (with time-series data) or the observation marker string (if any) at the start of each line. The `--no-dates` option suppresses the printing of dates or markers; a simple observation number is shown instead.

If the argument to print is a literal string (which must start with a double-quote, `"`), the string is printed as is. See also printf.

Note: a special "hack" is available with this command, in conjunction with the `--byobs` flag, which can be useful when working with missing values in a data set. If you give a list of variables followed by a semi-colon, followed by one final variable, then the final variable is not printed but is used to screen the observations to print. Any observations for which the screening variable has value 0 are not printed. As an example of use, suppose you have a daily time series `x`, and you want a list of the dates for which `x` is missing. You can do

```
genr filt = missing(x)
print x ; filt --byobs
```

Caminho de Menu: /Data/Display values

printf

Argumentos:	*format args*

Prints scalar values and/or strings under the control of a format string (providing a small subset of the `printf()` statement in the C programming language). Recognized numeric formats are `%e`, `%E`, `%f`, `%g`, `%G` and `%d`, in each case with the various modifiers available in C. Examples: the format `%.10g` prints a value to 10 significant figures; `%12.6f` prints a value to 6 decimal places, with a width of 12 characters. The format `%s` should be used for strings.

The format string itself must be enclosed in double quotes. The values to be printed must follow the format string, separated by commas. These values should take the form of either (a) the names of variables in the dataset, (b) expressions that are valid for the genr command, or (c) the special functions `varname()` or `date()`. The following example prints the values of two variables plus that of a calculated expression:

```
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
```

The next lines illustrate the use of the varname and date functions, which respectively print the name of a variable, given its ID number, and a date string, given a 1-based observation number.

```
printf "The name of variable %d is %s\n", i, varname(i)
printf "The date of observation %d is %s\n", j, date(j)
```

The maximum length of a format string is 127 characters. The escape sequences `\n` (newline), `\t` (tab), `\v` (vertical tab) and `\\` (literal backslash) are recognized. To print a literal percent sign, use `%%`.

probit

Argumentos:	*variável-dependente variáveis-independentes*
Opções:	`--robust` (robust standard errors)
	`--vcv` (mostrar matriz de covariância)
	`--verbose` (mostrar detalhes das iterações)

If the dependent variable is a binary variable (all values are 0 or 1) maximum likelihood estimates of the coefficients on *variáveis-independentes* are obtained via the "binary response model regression" (BRMR) method outlined by Davidson and MacKinnon (2004). As the model is nonlinear the slopes depend on the values of the independent variables: the reported slopes are evaluated at the means of those variables. The chi-square statistic tests the null hypothesis that all coefficients are zero apart from the constant.

By default, standard errors are computed using the negative inverse of the Hessian. If the `--robust` flag is given, then QML or Huber–White standard errors are calculated instead. In this case the estimated covariance matrix is a "sandwich" of the inverse of the estimated Hessian and the outer product of the gradient. See Davidson and MacKinnon (2004, Chapter 10) for details.

If the dependent variable is not binary, but is discrete and has a minimum value of 0, then Ordered Probit estimates are obtained. In this case robust standard errors are not yet available. (If the variable selected as dependent is not discrete, or does not have a minimum of zero, an error is flagged.)

Probit for analysis of proportions is not implemented in gretl at this point.

Caminho de Menu: /Model/Nonlinear models/Probit

pvalue

Argumentos:	*dist* `[` *params* `]` *xval*
Exemplos:	`pvalue z zscore`
	`pvalue t 25 3.0`
	`pvalue X 3 5.6`
	`pvalue F 4 58 fval`
	`pvalue G xbar varx x`
	`pvalue B bprob 10 6`

Computes the area to the right of *xval* in the specified distribution (`z` for Gaussian, `t` for Student's t, `X` for chi-square, `F` for F, `G` for gamma, or `B` for binomial).

For the t and chi-square distributions the degrees of freedom must be given; for F numerator and denominator degrees of freedom are required; for gamma the mean and variance are needed; and for the binomial distribution the "success" probability and the number of trials must be given. In each case, these extra values are provided before the *xval*.

As shown in the examples above, the numerical parameters may be given in numeric form or as the names of variables.

Caminho de Menu: /Tools/P-value finder

qlrtest

For a model estimated on time-series data via OLS, performs the Quandt likelihood ratio (QLR) test for a structural break at an unknown point in time, with 15 percent trimming at the beginning and end of the sample period.

For each potential break point within the central 70 percent of the observations, a Chow test is performed (see chow). The QLR test statistic is the maximum of the F values from these tests. It follows a non-standard distribution, the critical values of which are taken from Stock and Watson's *Introduction to Econometrics* (2003). If the QLR statistic exceeds the critical value at the chosen level of significance, one can infer that the parameters of the model are not constant. This statistic can be used to detect forms of instability other than a single discrete break (such as multiple breaks or a slow drifting of the parameters).

Caminho de Menu: Janela do modelo, /Testes/Teste QLR test

quantreg

Argumentos:	*tau depvar indepvars*
Opções:	`--robust` (robust standard errors)
	`--intervals[=`*level*`]` (compute confidence intervals)
	`--vcv` (print covariance matrix)
Exemplos:	`quantreg 0.25 y 0 xlist`
	`quantreg 0.5 y 0 xlist --intervals`
	`quantreg 0.5 y 0 xlist --intervals=.95`
	`quantreg tauvec y 0 xlist --robust`
	Ver também`mrw_qr.inp`

Quantile regression. The first argument, *tau*, is the conditional quantile for which estimates are wanted. It may be given either as a numerical value or as the name of a pre-defined scalar variable; the value must be in the range 0.01 to 0.99. (Alternatively, a vector of values may be given for *tau*; see below for details.) The second and subsequent arguments compose a regression list on the same pattern as ols.

Without the `--intervals` option, standard errors are printed for the quantile estimates. By default, these are computed according to the asymptotic formula given by Koenker and Bassett (1978), but if the `--robust` option is given, standard errors that are robust with respect to heteroskedasticity are calculated using the method of Koenker and Zhao (1994).

When the `--intervals` option is chosen, confidence intervals are given for the parameter estimates instead of standard errors. These intervals are computed using the rank inversion method, and in general they are asymmetrical about the point estimates. The specifics of the calculation are inflected by the `--robust` option: without this, the intervals are computed on the assumption of IID errors (Koenker, 1994); with it, they use the robust estimator developed by Koenker and Machado (1999).

By default, 90 percent confidence intervals are produced. You can change this by appending a confidence level (expressed as a decimal fraction) to the intervals option, as in `--intervals=0.95`.

Vector-valued *tau*: instead of supplying a scalar, you may give the name of a pre-defined matrix. In this case estimates are computed for all the given *tau* values and the results are printed in a special format, showing the sequence of quantile estimates for each regressor in turn.

Caminho de Menu: /Model/Robust estimation/Quantile regression

quit

Exits from the program, giving you the option of saving the output from the session on the way out.

Caminho de Menu: /Ficheiro/Exit

rename

Variantes:	`rename` *varnumber newname*
	`rename` *varname newname*

Changes the name of the variable with identification number *varnumber* or current name *varname* to *newname*. The new name must be of 15 characters maximum, must start with a letter, and must be composed of only letters, digits, and the underscore character.

Caminho de Menu: /Variável/Edit attributes

Acesso alternativo: Menu de contexto da janela principal (selecção singular)

reset

Must follow the estimation of a model via OLS. Carries out Ramsey's RESET test for model specification (non-linearity) by adding the square and the cube of the fitted values to the regression and calculating the F statistic for the null hypothesis that the parameters on the two added terms are zero.

Caminho de Menu: Janela do modelo, /Testes/Ramsey's RESET

restrict

Imposes a set of linear restrictions on either (a) the model last estimated or (b) a system of equations previously defined and named. The syntax and effects of the command differ slightly in the two cases.

In both cases the set of restrictions should be started with the keyword "restrict" and terminated with "end restrict". In the single equation case the restrictions are implicitly to be applied to the last model, and they are evaluated as soon as the `restrict` command is terminated. In the system case the initial "restrict" must be followed by the name of a previously defined system of equations (see system). The restrictions are evaluated when the system is next estimated, using the estimate command.

Each restriction in the set should be expressed as an equation, with a linear combination of parameters on the left and a numeric value to the right of the equals sign. In the single-equation case, parameters may be referenced in the form `b[`*i*`]`, where *i* represents the position in the list of regressors (starting at 1), or `b[`*varname*`]`, where *varname* is the name of the regressor in question. In the system case, parameters are referenced using `b` plus two numbers in square brackets. The leading number represents the position of the equation within the system and the second number indicates position in the list of regressors. For example `b[2,1]` denotes the first parameter in the second equation, and `b[3,2]` the second parameter in the third equation.

The `b` terms in the equation representing a restriction equation may be prefixed with a numeric multiplier, using `*` to represent multiplication, for example `3.5*b[4]`.

Here is an example of a set of restrictions for a previously estimated model:

```
restrict
 b[1] = 0
 b[2] - b[3] = 0
 b[4] + 2*b[5] = 1
end restrict
```

And here is an example of a set of restrictions to be applied to a named system. (If the name of the system does not contain spaces, the surrounding quotes are not required.)

```
restrict "System 1"
 b[1,1] = 0
 b[1,2] - b[2,2] = 0
 b[3,4] + 2*b[3,5] = 1
end restrict
```

In the single-equation case the restrictions are evaluated via a Wald F-test, using the coefficient covariance matrix of the model in question. By default, the restricted coefficient estimates are printed; if you just want the test statistic, you can append the `--quiet` option flag to the initial `restrict` command.

In the system case, the test statistic depends on the estimator chosen: a Likelihood Ratio test if the system is estimated using a Maximum Likelihood method, or an asymptotic F-test otherwise.

Caminho de Menu: Janela do modelo, /Testes/Linear restrictions

rhodiff

Argumentos:	*rholist* ; *lista-de-variáveis*
Exemplos:	`rhodiff .65 ; 2 3 4`
	`rhodiff r1 r2 ; x1 x2 x3`

Creates rho-differenced counterparts of the variables (given by number or by name) in *lista-de-variáveis* and adds them to the data set, using the suffix `#` for the new variables. Given variable `v1` in *lista-de-variáveis*, and entries `r1` and `r2` in *rholist*, the new variable

```
v1# = v1 - r1*v1(-1) - r2*v1(-2)
```

is created. The *rholist* entries can be given as numerical values or as the names of variables previously defined.

rmplot

Argumento: *nome-de-variável*

Range–mean plot: this command creates a simple graph to help in deciding whether a time series, y(t), has constant variance or not. We take the full sample t=1,...,T and divide it into small subsamples of arbitrary size k. The first subsample is formed by y(1),...,y(k), the second is y(k+1), ..., y(2k), and so on. For each subsample we calculate the sample mean and range (= maximum minus minimum), and we construct a graph with the means on the horizontal axis and the ranges on the vertical. So each subsample is represented by a point in this plane. If the variance of the series is constant we would expect the subsample range to be independent of the subsample mean; if we see the points approximate an upward-sloping line this suggests the variance of the series is increasing in its mean; and if the points approximate a downward sloping line this suggests the variance is decreasing in the mean.

Besides the graph, gretl displays the means and ranges for each subsample, along with the slope coefficient for an OLS regression of the range on the mean and the p-value for the null hypothesis that this slope is zero. If the slope coefficient is significant at the 10 percent significance level then the fitted line from the regression of range on mean is shown on the graph.

Caminho de Menu: /Variável/Range-mean graph

run

Argumento: *inputfile*

Execute the commands in *inputfile* then return control to the interactive prompt. This command is intended for use with the command-line program gretlcli, or at the “gretl console” in the GUI program.

See also include.

Caminho de Menu: Run icon in script window

runs

Argumento: *nome-de-variável*

Carries out the nonparametric “runs” test for randomness of the specified variable. If you want to test for randomness of deviations from the median, for a variable named `x1` with a non-zero median, you can do the following:

```
genr signx1 = x1 - median(x1)
runs signx1
```

Caminho de Menu: /Variável/Runs test

scatters

Argumentos:	*yvar* ; *xvarlist* ou *yvarlist* ; *xvar*
Opção:	`--with-lines` (create line graphs)
Exemplos:	`scatters 1 ; 2 3 4 5`
	`scatters 1 2 3 4 5 6 ; 7`
	`scatters y1 y2 y3 ; x --with-lines`

Generates pairwise graphs of *yvar* against all the variables in *xvarlist*, or of all the variables in *yvarlist* against *xvar*. The first example above puts variable 1 on the y-axis and draws four graphs, the first having variable 2 on the x-axis, the second variable 3 on the x-axis, and so on. The second example plots each of variables 1 through 6 against variable 7 on the x-axis. Scanning a set of such plots can be a useful step in exploratory data analysis. The maximum number of plots is six; any extra variable in the list will be ignored.

By default the graphs are scatterplots, but if you give the `--with-lines` flag they will be line graphs.

Caminho de Menu: /View/Multiple graphs

sdiff

Argumento:	*lista-de-variáveis*

The seasonal difference of each variable in *lista-de-variáveis* is obtained and the result stored in a new variable with the prefix `sd_`. This command is available only for seasonal time series.

Caminho de Menu: /Add/Seasonal differences of selected variables

set

Argumentos:	*variável value*
Exemplos:	`set qr on`
	`set csv_delim tab`
	`set horizon 10`

Set the values of various program parameters. The given value remains in force for the duration of the gretl session unless it is changed by a further call to set. The parameters that can be set in this way are enumerated below. Note that the settings of `hac_lag` and `hc_version` are used when the `--robust` option is given to the ols command.

If the set command is given without any parameters, the current settings for all the relevant variables are printed.

- `echo`: `off` or `on` (the default). Suppress or resume the echoing of commands in gretl's output.
- `messages`: `off` or `on` (the default). Suppress or resume the printing of non-error messages associated with various commands, for example when a new variable is generated or when the sample range is changed.
- `nls_toler`: a floating-point value (the default is the machine precision to the power 3/4). Sets the tolerance used in judging whether or not convergence has occurred in nonlinear least squares estimation using the nls command.
- `qr`: `on` or `off` (the default). Use QR rather than Cholesky decomposition in calculating OLS estimates.
- `seed`: an unsigned integer. Sets the seed for the pseudo-random number generator. By default this is set from the system time; if you want to generate repeatable sequences of random numbers you must set the seed manually.
- `hac_lag`: `nw1` (the default) or `nw2`, or an integer. Sets the maximum lag value, p, used when calculating HAC (Heteroskedasticity and Autocorrelation Consistent) standard errors using the Newey-West approach, for time series data. `nw1` and `nw2` represent two variant automatic calculations based on the sample size, T: for nw1, $p = 0.75 \times T^{1/3}$, and for nw2, $p = 4 \times (T/100)^{2/9}$.

- `hc_version`: 0 (the default), 1, 2 or 3. Sets the variant used when calculating Heteroskedasticity Consistent standard errors with cross-sectional data. The options correspond to the HC0, HC1, HC2 and HC3 discussed by Davidson and MacKinnon in *Econometric Theory and Methods*, chapter 5. HC0 produces what are usually called "White's standard errors".

- `force_hc`: `off` (the default) or `on`. By default, with time-series data and when the `--robust` option is given with `ols`, the HAC estimator is used. If you set `force_hc` to "on", this forces calculation of the regular Heteroskedasticity Consistent Covariance Matrix (which does not take autocorrelation into account).

- `garch_vcv`: `unset`, `hessian`, `im` (information matrix) , `op` (outer product matrix), `qml` (QML estimator), `bw` (Bollerslev–Wooldridge). Specifies the variant that will be used for estimating the coefficient covariance matrix, for GARCH models. If `unset` is given (the default) then the Hessian is used unless the "robust" option is given for the garch command, in which case QML is used.

- `hp_lambda`: `auto` (the default), or a numerical value. Sets the smoothing parameter for the Hodrick–Prescott filter (see the `hpfilt` function under the `genr` command). The default is to use 100 times the square of the periodicity, which gives 100 for annual data, 1600 for quarterly data, and so on.

- `bkbp_limits`: two integers, the second greater than the first (the defaults are 8 and 32). Sets the frequency bounds for the Baxter–King bandpass filter (see the `bkfilt` function under the `genr` command).

- `bkbp_k`: one integer (the default is 8). Sets the approximation order for the Baxter–King bandpass filter.

- `horizon`: one integer (the default is based on the frequency of the data). Sets the horizon for impulse responses and forecast variance decompositions in the context of vector autoregressions.

- `csv_delim`: either `comma` (the default), `space` or `tab`. Sets the column delimiter used when saving data to file in CSV format.

- `bhhh_maxiter`: one integer, maximum number of iterations for gretl's internal BHHH routine, which is used in the arma command for conditional ML estimation. If convergence is not achieved after `bhhh_maxiter`, the program returns an error. The default is set at 500.

- `bhhh_toler`: one floating point value, or the string `default`. This is used in gretl's internal BHHH routine to check if convergence has occurred. The algorithm stops iterating as soon as the increment in the log-likelihood between iterations is smaller than `bhhh_toler`. The default value is 1.0E−06; this value may be re-established by typing `default` in place of a numeric value.

- `longdigits`: one positive integer value, less than or equal to 20. Determines the number of digits of precision used when printing the values of variables using the `--long` option (see print).

- `initvals`: a pre-specified matrix. Allows manual setting of the initial parameter estimates for ARMA estimation. For details see *Manual de Utilização do Gretl*.

setinfo

Argumentos:	*nome-de-variável* `-d` *description* `-n` *displayname*
Opções:	`--discrete` (mark variable as discrete)
	`--continuous` (mark variable as continuous)
Exemplos:	`setinfo x1 -d "Description of x1-n "Graph name"`
	`setinfo z --discrete`

Sets up to three attributes of the named variable, as follows.

If the `-d` flag is given followed by a string in double quotes, that string is used to set the variable's descriptive label. This label is shown in response to the labels command, and is also shown in the main window of the GUI program.

If the `-n` flag is given followed by a quoted string, that string is used to set the variable's "display name", which is then used in place of the variable's name in graphs.

If one or other of the `--discrete` or `--continuous` option flags is given, the variable's numerical character is set accordingly. The default is to treat all variables as continuous; setting a variable as discrete affects the way the variable is handled in frequency plots.

Caminho de Menu: /Variável/Edit attributes

Acesso alternativo: Menu de contexto da janela principal

setobs

Variantes:	setobs *periodicity startobs*
	setobs *unitvar timevar*
Opções:	`--cross-section` (interpret as cross section)
	`--time-series` (interpret as time series)
	`--stacked-cross-section` (interpret as panel data)
	`--stacked-time-series` (interpret as panel data)
	`--panel-vars` (use index variables (see below))
Exemplos:	`setobs 4 1990:1 --time-series`
	`setobs 12 1978:03`
	`setobs 1 1 --cross-section`
	`setobs 20 1:1 --stacked-time-series`
	`setobs unit year --panel-vars`

Force the program to interpret the current data set as having a specified structure.

In the first form of the command the *periodicity*, which must be an integer, represents frequency in the case of time-series data (1 = annual; 4 = quarterly; 12 = monthly; 52 = weekly; 5, 6, or 7 = daily; 24 = hourly). In the case of panel data the periodicity means the number of lines per data block: this corresponds to the number of cross-sectional units in the case of stacked cross-sections, or the number of time periods in the case of stacked time series. In the case of simple cross-sectional data the periodicity should be set to 1.

The starting observation represents the starting date in the case of time series data. Years may be given with two or four digits; subperiods (for example, quarters or months) should be separated from the year with a colon. In the case of panel data the starting observation should be given as 1:1; and in the case of cross-sectional data, as 1. Starting observations for daily or weekly data should be given in the form YY/MM/DD or YYYY/MM/DD (or simply as 1 for undated data).

The second form of the command (which requires the `--panel-vars` flag) may be used to impose a panel interpretation when the data set contains variables that uniquely identify the cross-sectional units and the time periods. The data set will be sorted as stacked time series, by ascending values of the units variable, *unitvar*.

If no explicit option flag is given to indicate the structure of the data the program will attempt to guess the structure from the information given.

Caminho de Menu: /Data/Dataset structure

setmiss

Argumentos:	*value* [*lista-de-variáveis*]
Exemplos:	`setmiss -1`
	`setmiss 100 x2`

Get the program to interpret some specific numerical data value (the first parameter to the command) as a code for "missing", in the case of imported data. If this value is the only parameter, as in the first example above, the interpretation will be applied to all series in the data set. If *value* is followed by a list of variables, by name or number, the interpretation is confined to the specified variable(s). Thus in the second example the data value 100 is interpreted as a code for "missing", but only for the variable `x2`.

Caminho de Menu: /Sample/Set missing value code

shell

Argumento: *shellcommand*

Exemplos:
```
! ls -al
! notepad
launch notepad
```

A !, or the keyword launch, at the beginning of a command line is interpreted as an escape to the user's shell. Thus arbitrary shell commands can be executed from within gretl. When ! is used, the external command is executed synchronously. That is, gretl waits for it to complete before proceeding. If you want to start another program from within gretl and not wait for its completion (asynchronous operation), use launch instead.

For reasons of security this facility is not enabled by default. To activate it, check the box titled "Allow shell commands" under the File, Preferences menu in the GUI program. This also makes shell commands available in the command-line program (and is the only way to do so).

smpl

Variantes:
`smpl` *startobs endobs*
`smpl` *+i -j*
`smpl` *dumvar* `--dummy`
`smpl` *condition* `--restrict`
`smpl --no-missing [` *lista-de-variáveis* `]`
`smpl` *n* `--random`
`smpl full`

Exemplos:
```
smpl 3 10
smpl 1960:2 1982:4
smpl +1 -1
smpl x > 3000 --restrict
smpl y > 3000 --restrict --replace
smpl 100 --random
```

Resets the sample range. The new range can be defined in several ways. In the first alternate form (and the first two examples) above, *startobs* and *endobs* must be consistent with the periodicity of the data. Either one may be replaced by a semicolon to leave the value unchanged. In the second form, the integers *i* and *j* (which may be positive or negative, and should be signed) are taken as offsets relative to the existing sample range. In the third form *dummyvar* must be an indicator variable with values 0 or 1 at each observation; the sample will be restricted to observations where the value is 1. The fourth form, using `--restrict`, restricts the sample to observations that satisfy the given Boolean condition (which is specified according to the syntax of the genr command).

With the `--no-missing` form, if *lista-de-variáveis* is specified observations are selected on condition that all variables in *lista-de-variáveis* have valid values at that observation; otherwise, if no *lista-de-variáveis* is given, observations are selected on condition that *all* variables have valid (non-missing) values.

With the `--random` flag, the specified number of cases are selected from the full dataset at random. If you wish to be able to replicate this selection you should set the seed for the random number generator first (see the set command).

The final form, `smpl full`, restores the full data range.

Note that sample restrictions are, by default, cumulative: the baseline for any `smpl` command is the current sample. If you wish the command to act so as to replace any existing restriction you can add the option flag `--replace` to the end of the command.

The internal variable `obs` may be used with the `--restrict` form of `smpl` to exclude particular observations from the sample. For example

```
smpl obs!=4 --restrict
```

will drop just the fourth observation. If the data points are identified by labels,

```
smpl obs!="USA" --restrict
```

will drop the observation with label "USA".

One point should be noted about the `--dummy`, `--restrict` and `--no-missing` forms of `smpl`: Any "structural" information in the data file (regarding the time series or panel nature of the data) is lost when this command is issued. You may reimpose structure with the setobs command.

Please see *Manual de Utilização do Gretl* for further details.

Caminho de Menu: /Sample

spearman

Argumentos:	*x y*
Opção:	`--verbose` (print ranked data)

Prints Spearman's rank correlation coefficient for the two variables x and y. The variables do not have to be ranked manually in advance; the function takes care of this.

The automatic ranking is from largest to smallest (i.e. the largest data value gets rank 1). If you need to invert this ranking, create a new variable which is the negative of the original first. For example:

```
genr altx = -x
spearman altx y
```

Caminho de Menu: /Model/Robust estimation/Rank correlation

sprintf

Argumentos: *stringvar format args*

This command works exactly like the printf command, printing the given arguments under the control of the format string, except that the result is written into the named string, *stringvar*.

square

Argumento:	*lista-de-variáveis*
Opção:	`--cross` (generate cross-products as well as squares)

Generates new variables which are squares of the variables in *lista-de-variáveis* (plus cross-products if the `--cross` option is given). For example, square x y will generate `sq_x` = `x` squared, `sq_y` = `y` squared and (optionally) `x_y` = `x` times `y`. If a particular variable is a dummy variable it is not squared because we will get the same variable.

Caminho de Menu: /Add/Squares of selected variables

sscanf

Argumentos: *source* , *format* , *args*

Scans the string *source* under the control of *format*, assigning zero or more values to the given *args*. This is a simplifed version of the `sscanf` function in the C programming language.

source may be either a literal string, enclosed in double quotes, or the name of a predefined string variable. *format* is defined similarly to the format string in printf (more on this below). *args* should be a comma-separated list containing the names of pre-defined variables: these are the targets of conversion from *source*. (For those used to C: one can prefix the names of numerical variables with `&` but this is not required.)

Literal text in *format* is matched against *source*. Conversion specifiers start with `%`, and recognized conversions include `%f`, `%g` or `%lf` for floating-point numbers; `%d` for integers; `%s` for strings; and `%m` for matrices. You may insert a positive integer after the percent sign: this sets the maximum number of characters to read for the given conversion (or the maximum number of rows in the case of matrix conversion). Alternatively, you can insert a literal `*` after the percent to suppress the conversion (thereby skipping any characters that would otherwise have been converted for the given type). For example, `%3d` converts the next 3 characters in *source* to an integer, if possible; `%*g` skips as many characters in *source* as could be converted to a single floating-point number.

Matrix conversion works thus: the scanner reads a line of input and counts the (space- or tab-separated) number of numeric fields. This defines the number of columns in the matrix. By default, reading then proceeds for as many lines (rows) as contain the same number of numeric columns, but the maximum number of rows to read can be limited as described above.

In addition to `%s` conversion for strings, a simplified version of the C format `%`*N*`[`*chars*`]` is available. In this format *N* is the maximum number of characters to read and *chars* is a set of acceptable characters, enclosed in square brackets: reading stops if *N* is reached or if a character not in *chars* is encountered. The function of *chars* can be reversed by giving a circumflex, `^`, as the first character; in that case reading stops if a character in the given set is found. (Unlike C, the hyphen does not play a special role in the *chars* set.)

If the source string does not (fully) match the format, the number of conversions may fall short of the number of arguments given. This is not in itself an error so far as gretl is concerned. However, you may wish to check the number of conversions performed; this is given by the built-in scalar variable `$nscan`, whose value is refreshed every time `sscanf` is called. The number of conversions is also printed, in interactive use.

Some examples follow:

```
scalar x
scalar y
sscanf "123456", "%3d%3d", x, y

sprintf S "1 2 3 4\n5 6 7 8"
S
matrix m
sscanf S, "%m", m
print m
```

store

Argumentos:	*ficheiro-de-dados* [*lista-de-variáveis*]
Opções:	`--csv` (use CSV format)
	`--omit-obs` (see below, on CSV format)
	`--gnu-octave` (use GNU Octave format)
	`--gnu-R` (use GNU R format)
	`--traditional` (use traditional ESL format)
	`--gzipped` (apply gzip compression)
	`--jmulti` (use JMulti ASCII format)
	`--dat` (use PcGive ASCII format)
	`--database` (use gretl database format)
	`--overwrite` (see below, on database format)

Saves either the entire dataset or, if a *lista-de-variáveis* is supplied, a specified subset of the variables in the current dataset, to the file given by *datafile*.

By default the data are saved in "native" gretl format, but the option flags permit saving in several alternative formats. CSV (Comma-Separated Values) data may be read into spreadsheet programs, and can also be manipulated using a text editor. The formats of Octave, R and PcGive are designed for use with the respective programs. Gzip compression may be useful for large datasets. See *Manual de Utilização do Gretl* for details on the various formats.

The option flag `--omit-obs` is applicable only when saving data in CSV format. By default, if the data are time series or panel or if the dataset includes specific observation markers, the CSV file includes a first column identifying the observations (e.g. by date). If the `--omit-obs` flag is given this column is omitted; only the actual data are printed.

Note that any scalar variables will not be saved automatically: if you wish to save scalars you must explicitly list them in *lista-de-variáveis*.

The option of saving in gretl database format is intended to help with the construction of large sets of series, possibly having mixed frequencies and ranges of observations. At present this option is available only for annual, quarterly or monthly time-series data. If you save to a file that already exists, the default action is to append the newly saved series to the existing content of the database. In this context it is an error if one or more of the variables to be saved has the same name as a variable that is already present in the database. The `--overwrite` flag has the effect that, if there are variable names in common, the newly saved variable replaces the variable of the same name in the original dataset.

Caminho de Menu: /Ficheiro/Save data; /File/Export data

summary

Argumento:	`[` *lista-de-variáveis* `]`

Print summary statistics for the variables in *lista-de-variáveis*, or for all the variables in the data set if *lista-de-variáveis* is omitted. Output consists of the mean, standard deviation (sd), coefficient of variation (= sd/mean), median, minimum, maximum, skewness coefficient, and excess kurtosis.

Caminho de Menu: /View/Summary statistics

Acesso alternativo: Menu de contexto da janela principal

system

Variantes:	`system method=`*estimator*
	`system name=`*sysname*
Argumento:	*savevars*
Exemplos:	`system name="Klein Model 1"`
	`system method=sur`
	`system method=sur save=resids`
	`system method=3sls save=resids,fitted`
	Ver também`klein.inp`, `kmenta.inp`

Starts a system of equations. Either of two forms of the command may be given, depending on whether you wish to save the system for estimation in more than one way or just estimate the system once.

To save the system you should give it a name, as in the first example (if the name contains spaces it must be surrounded by double quotes). In this case you estimate the system using the estimate command. With a saved system of equations, you are able to impose restrictions (including cross-equation restrictions) using the restrict command.

Alternatively you can specify an estimator for the system using `method=` followed by a string identifying one of the supported estimators: ols (Ordinary Least Squares), tsls (Two-Stage Least Squares) sur (Seemingly Unrelated Regressions), 3sls (Three-Stage Least Squares), fiml (Full Information Maximum Likelihood) or liml (Limited Information Maximum Likelihood). In this case the system is estimated once its definition is complete.

An equation system is terminated by the line end system. Within the system four sorts of statement may be given, as follows.

- equation: specify an equation within the system. At least two such statements must be provided.
- instr: for a system to be estimated via Three-Stage Least Squares, a list of instruments (by variable name or number). Alternatively, you can put this information into the equation line using the same syntax as in the tsls command.

- endog: for a system of simultaneous equations, a list of endogenous variables. This is primarily intended for use with FIML estimation, but with Three-Stage Least Squares this approach may be used instead of giving an instr list; then all the variables not identified as endogenous will be used as instruments.
- identity: for use with FIML, an identity linking two or more of the variables in the system. This sort of statement is ignored when an estimator other than FIML is used.

In the optional save= field of the command you can specify whether to save the residuals (resids) and/or the fitted values (fitted).

For examples of the specification and estimation of systems of equations, please see the scripts `klein.inp`, `kmenta.inp` and `greene14_2.inp` (supplied with the gretl distribution).

Caminho de Menu: /Model/Simultaneous equations

tabprint

Argumento:	`[` *-f filename* `]`
Opções:	`--complete` (Create a complete document)
	`--format="f1|f2|f3|f4"` (Specify a custom format)

Must follow the estimation of a model. Prints the estimated model in the form of a LaTeX table. If a filename is specified using the `-f` flag output goes to that file, otherwise it goes to a file with a name of the form `model_N.tex`, where `N` is the number of models estimated to date in the current session. See also eqnprint.

If the `--complete` flag is given the LaTeX file is a complete document, ready for processing; otherwise it must be included in a document.

If you wish alter the appearance of the tabular output, you can specify a custom row format using the `--format` flag. The format string must be enclosed in double quotes and must be tied to the flag with an equals sign. The pattern for the format string is as follows. There are four fields, representing the coefficient, standard error, t-ratio and p-value respectively. These fields should be separated by vertical bars; they may contain a `printf`-type specification for the formatting of the numeric value in question, or may be left blank to suppress the printing of that column (subject to the constraint that you can't leave all the columns blank). Here are a few examples:

```
--format="%.4f|%.4f|%.4f|%.4f"
--format="%.4f|%.4f|%.3f|"
--format="%.5f|%.4f||%.4f"
--format="%.8g|%.8g||%.4f"
```

The first of these specifications prints the values in all columns using 4 decimal places. The second suppresses the p-value and prints the t-ratio to 3 places. The third omits the t-ratio. The last one again omits the t, and prints both coefficient and standard error to 8 significant figures.

Once you set a custom format in this way, it is remembered and used for the duration of the gretl session. To revert to the default format you can use the special variant `--format=default`.

Caminho de Menu: Janela do modelo, /LaTeX

testuhat

Must follow a model estimation command. Gives the frequency distribution for the residual from the model along with a chi-square test for normality, based on the procedure suggested by Doornik and Hansen (1984).

Caminho de Menu: Janela do modelo, /Testes/Normality of residual

tobit

Argumentos:	*variável-dependente variáveis-independentes*
Opções:	`--vcv` (mostrar matriz de covariância)
	`--verbose` (mostrar detalhes das iterações)

Estimates a Tobit model. This model may be appropriate when the dependent variable is "censored". For example, positive and zero values of purchases of durable goods on the part of individual households are observed, and no negative values, yet decisions on such purchases may be thought of as outcomes of an underlying, unobserved disposition to purchase that may be negative in some cases. For details see Greene's *Econometric Analysis*, Chapter 20.

Caminho de Menu: /Model/Nonlinear models/Tobit

tsls

Argumentos:	*variável-dependente variáveis-independentes* ; *instruments*
Opções:	`--vcv` (mostrar matriz de covariância)
	`--robust` (robust standard errors)
Exemplo:	`tsls y1 0 y2 y3 x1 x2 ; 0 x1 x2 x3 x4 x5 x6`

Computes two-stage least squares (TSLS or IV) estimates: *depvar* is the dependent variable, *variáveis-independentes* is the list of independent variables (including right-hand side endogenous variables) in the structural equation for which TSLS estimates are needed; and *instruments* is the combined list of exogenous and predetermined variables in all the equations. If the *instruments* list is not at least as long as *variáveis-independentes*, the model is not identified.

In the above example, the `y`s are the endogenous variables and the `x`s are the exogenous and predetermined variables.

Output includes the Hausman test and, if the model is over-identified, the Sargan over-identification test. In the Hausman test, the null hypothesis is that OLS estimates are consistent, or in other words estimation by means of instrumental variables is not required. A model of this sort is over-identified if there are more instruments than are strictly required. The Sargan test is based on an auxiliary regression of the residuals from the two-stage least squares model on the full list of instruments. The null hypothesis is that all the instruments are valid, and suspicion is thrown on this hypothesis if the auxiliary regression has a significant degree of explanatory power. Davidson and MacKinnon (2004, chapter 8) give a good explanation of both tests.

Caminho de Menu: /Model/Other linear models/Two-Stage least Squares

var

Argumentos:	*ordem lista-de-variáveis* [; *exolist*]
Opções:	`--nc` (não incluir uma constante)
	`--seasonals` (incluir variáveis sazonais auxiliares)
	`--robust` (robust standard errors)
	`--impulse-responses` (print impulse responses)
	`--variance-decomp` (print forecast variance decompositions)
	`--lagselect` (show information criteria for lag selection)
Exemplos:	`var 4 x1 x2 x3 ; time mydum`
	`var 4 x1 x2 x3 --seasonals`
	`var 12 x1 x2 x3 --lagselect`

Sets up and estimates (using OLS) a vector autoregression (VAR). The first argument specifies the lag order — or the maximum lag order in case the `--lagselect` option is given (see below). The order may be given numerically, or as the name of a pre-existing scalar variable. Then follows the setup for the first equation. Don't include lags among the elements of *lista-de-variáveis* — they will be added automatically. The semi-colon separates the stochastic variables, for which *order* lags will be included, from any exogenous variables in *exolist*. Note that a constant is included automatically unless you give the `--nc` flag, a trend can be added with the `--trend` flag, and seasonal dummy variables may be added using the `--seasonals` flag.

A separate regression is run for each variable in *lista-de-variáveis*. Output for each equation includes F-tests for zero restrictions on all lags of each of the variables, an F-test for the significance of the maximum lag, and, if the `--impulse-responses` flag is given, forecast variance decompositions and impulse responses.

Forecast variance decompositions and impulse responses are based on the Cholesky decomposition of the contemporaneous covariance matrix, and in this context the order in which the (stochastic) variables are given matters. The first variable in the list is assumed to be "most exogenous" within-period. The horizon for variance decompositions and impulse responses can be set using the set command.

If the `--lagselect` option is given, the first parameter to the `var` command is taken as the maximum lag order. Output consists of a table showing the values of the Akaike (AIC), Schwartz (BIC) and Hannan–Quinn (HQC) information criteria computed from VARs of order 1 to the given maximum. This is intended to help with the selection of the optimal lag order. The usual VAR output is not presented.

Caminho de Menu: /Modelo/Série temporal/Vector autoregression

varlist

Prints a listing of variables currently available. list and ls are synonyms.

vartest

Argumentos:	*var1 var2*

Calculates the F statistic for the null hypothesis that the population variances for the variables *var1* and *var2* are equal, and shows its p-value.

Caminho de Menu: /Model/Bivariate tests/Difference of variances

vecm

Argumentos:	*ordem rank lista-de-variáveis*
Opções:	`--nc` (sem constante)
	`--rc` (constante restringida)
	`--crt` (constante e tendência restringida)
	`--ct` (constante e tendência não restringida)
	`--seasonals` (incluir auxiliares sazonais centradas)
	`--impulse-responses` (print impulse responses)
	`--variance-decomp` (print forecast variance decompositions)
Exemplos:	`vecm 4 1 Y1 Y2 Y3`
	`vecm 3 2 Y1 Y2 Y3 --rc`
	Ver também`denmark.inp`, `hamilton.inp`

A VECM is a form of vector autoregression or VAR (see var), applicable where the variables in the model are individually integrated of order 1 (that is, are random walks, with or without drift), but exhibit cointegration. This command is closely related to the Johansen test for cointegration (see coint2).

The *order* parameter to this command represents the lag order of the VAR system. The number of lags in the VECM itself (where the dependent variable is given as a first difference) is one less than *order*.

The *rank* parameter represents the cointegration rank, or in other words the number of cointegrating vectors. This must be greater than zero and less than or equal to (generally, less than) the number of endogenous variables given in *lista-de-variáveis*.

lista-de-variáveis supplies the list of endogenous variables, in levels. The inclusion of deterministic terms in the model is controlled by the option flags. The default if no option is specified is to include an "unrestricted constant", which allows for the presence of a non-zero intercept in the cointegrating relations as well as a trend in the levels of the endogenous variables. In the literature stemming from the work of Johensen (see for example his 1995 book) this is often referred to as "case 3". The first four options given above, which are mutually exclusive, produce cases 1, 2, 4 and 5 respectively. The meaning of these cases and the criteria for selecting a case are explained in *Manual de Utilização do Gretl*.

The `--seasonals` option, which may be combined with any of the other options, specifies the inclusion of a set of centered seasonal dummy variables. This option is available only for quarterly or monthly data.

The first example above specifies a VECM with lag order 4 and a single cointegrating vector. The endogenous variables are `Y1`, `Y2` and `Y3`. The second example uses the same variables but specifies a lag order of 3 and two cointegrating vectors; it also specifies a "restricted constant", which is appropriate if the cointegrating vectors may have a non-zero intercept but the `Y` variables have no trend.

Caminho de Menu: /Modelo/Série temporal/VECM

vif

Must follow the estimation of a model which includes at least two independent variables. Calculates and displays the Variance Inflation Factors (VIFs) for the regressors. The VIF for regressor j is defined as

$$\frac{1}{1 - R_j^2}$$

where R_j is the coefficient of multiple correlation between regressor j and the other regressors. The factor has a minimum value of 1.0 when the variable in question is orthogonal to the other independent variables. Neter, Wasserman, and Kutner (1990) suggest inspecting the largest VIF as a diagnostic for collinearity; a value greater than 10 is sometimes taken as indicating a problematic degree of collinearity.

Caminho de Menu: Janela do modelo, /Testes/Collinearity

wls

Argumentos:	*wtvar variável-dependente variáveis-independentes*
Opções:	`--vcv` (mostrar matriz de covariância)
	`--robust` (robust standard errors)
	`--quiet` (não mostrar resultados da regressão)

Computes weighted least squares (WLS) estimates using *wtvar* as the weight, *depvar* as the dependent variable, and *variáveis-independentes* as the list of independent variables. Let *w* denote the positive square root of `wtvar`; then WLS is basically equivalent to an OLS regression of *w* `*` *depvar* on *w* `*` *variáveis-independentes*. The *R*-squared, however, is calculated in a special manner, namely as

$$R^2 = 1 - \frac{\text{ESS}}{\text{WTSS}}$$

where ESS is the error sum of squares (sum of squared residuals) from the weighted regression and WTSS denotes the "weighted total sum of squares", which equals the sum of squared residuals from a regression of the weighted dependent variable on the weighted constant alone.

If *wtvar* is a dummy variable, WLS estimation is equivalent to eliminating all observations with value zero for *wtvar*.

Caminho de Menu: /Model/Other linear models/Weighted Least Squares

xcorrgm

Argumentos:	*var1 var2* `[` *maxlag* `]`
Exemplo:	`xcorrgm x y 12`

Prints and graphs the cross-correlogram for variables *var1* and *var2*, which may be specified by name or number. The values are the sample correlation coefficients between the current value of *var1* and successive leads and lags of *var2*.

If a *maxlag* value is specified the length of the cross-correlogram is limited to at most that number of leads and lags, otherwise the length is determined automatically, as a function of the frequency of the data and the number of observations.

Caminho de Menu: /View/Cross-correlogram

Acesso alternativo: Menu de contexto da janela principal (multiple selection)

xtab

Argumentos:	*ylist* ; *xlist*
Opções:	`--row` (display row percentages)
	`--column` (display column percentages)
	`--zeros` (display zero entries)

With no options given, displays a contingency table or cross-tabulation for each variable in *ylist* (by row) against each variable in *xlist* (by column). Variables in these lists can be referenced by name or by number. Note that all the variables must have been marked as discrete.

The `--row` and `--column` options are mutually exclusive, and instead of the frequency count yield the percentages for each row or column, respectively. The `--zeros` option may be useful for importing the table into another program, such as a spreadsheet.

Pearson's chi-square test for independence is displayed if the expected frequency under independence is at least 1.0e-7 for all cells. A common rule of thumb for the validity of this statistic is that at least 80 percent of cells should have expected frequencies of 5 or greater; if this criterion is not met a warning is printed.

1.3 Commands by topic

The following sections show the available commands grouped by topic.

Estimação

`ar`	Estimação autoregressiva	`ar1`	AR(1) estimation
`arbond`	Modelos de painel dinâmico	`arima`	Modelo ARMA
`equation`	Define equation within a system	`estimate`	Estimate system of equations
`garch`	GARCH model	`gmm`	GMM estimation
`hccm`	HCCM estimation	`heckit`	Heckman selection model
`hsk`	Heteroskedasticity-corrected estimates	`lad`	Least Absolute Deviation estimation
`logistic`	Logistic regression	`logit`	Logit regression
`mle`	Maximum likelihood estimation	`mpols`	Multiple-precision OLS
`nls`	Nonlinear Least Squares	`ols`	Ordinary Least Squares
`panel`	Panel models	`poisson`	Poisson estimation
`probit`	Probit model	`quantreg`	Quantile regression
`system`	Systems of equations	`tobit`	Tobit model
`tsls`	Two-Stage Least Squares	`var`	Vector Autoregression
`vecm`	Vector Error Correction Model	`wls`	Weighted Least Squares

Testes

`add`	Acrescentar variáveis ao modelo	`adf`	Teste de Dickey-Fuller aumentado
`arch`	Teste ARCH	`chow`	Teste de Chow
`coeffsum`	Soma de coeficientes	`coint`	Teste de cointegração Engle-Granger
`coint2`	Teste de cointegração de Johansen	`cusum`	Teste CUSUM
`difftest`	Nonparametric tests for differences	`hausman`	Panel diagnostics
`kpss`	KPSS stationarity test	`leverage`	Influential observations
`lmtest`	LM tests	`meantest`	Difference of means
`normtest`	Normality test	`omit`	Omit variables
`qlrtest`	Quandt likelihood ratio test	`reset`	Ramsey's RESET
`restrict`	Linear restrictions	`runs`	Runs test
`testuhat`	Normality of residual	`vartest`	Difference of variances
`vif`	Variance Inflation Factors		

Transformações

diff	First differences	discrete	Mark variables as discrete
dummify	Create sets of dummies	lags	Create lags
ldiff	Log-differences	logs	Create logs
orthdev	Orthogonal deviations	rhodiff	Quasi-differencing
sdiff	Seasonal differencing	square	Create squares of variables

Estatísticas

corr	Coeficientes de correlação	corrgm	Correlograma
freq	Frequency distribution	hurst	Hurst exponent
mahal	Mahalanobis distances	pca	Principal Components Analysis
pergm	Periodogram	spearman	Spearmans's rank correlation
summary	Descriptive statistics	xcorrgm	Cross-correlogram
xtab	Cross-tabulate variables		

Conjunto de Dados

append	Acrescentar dados	data	Import from database
dataset	Manipulate the dataset	delete	Delete variables
genr	Generate a new variable	info	Information on data set
labels	Print labels for variables	nulldata	Creating a blank dataset
open	Open a data file	rename	Rename variables
setinfo	Edit attributes of variable	setobs	Set frequency and starting observation
setmiss	Missing value code	smpl	Set the sample range
store	Save data	varlist	Listing of variables

Gráficos

boxplot	Gráficos caixa-com-bigodes	gnuplot	Create a gnuplot graph
graph	Create ASCII graph	plot	ASCII plot
rmplot	Range-mean plot	scatters	Multiple pairwise graphs

Impressão

eqnprint	Print model as equation	outfile	Direct printing to file
print	Print data or strings	printf	Formatted printing
sprintf	Printing to a string	tabprint	Print model in tabular form

Predição

fcast	Generate forecasts

Programação

break	Sair de um ciclo	elif	Flow control
else		end	End block of commands
endif	Flow control	endloop	End a command loop
foreign	Non-native script	function	Define a function
if	Flow control	include	Include function definitions
loop	Start a command loop	run	Execute a script

set	Set program parameters	sscanf	Scanning a string

Utilidades

criteria	Critério de selecção de modelos	help	Help on commands
modeltab	The model table	pvalue	Compute p-values
quit	Exit the program	shell	Execute shell commands

Capítulo 2

Gretl functions

2.1 Introduction

This chapter presents two alphabetical listings: first, the "accessors" which enable the user to retrieve the values of internal variables; and second, the functions proper that are available in the context of the `genr` command.

2.2 Accessors

$ahat

Output: series

Must follow the estimation of a fixed-effect panel data model. Returns the estimates of individual fixed effects (per-unit intercepts).

$aic

Output: scalar

Returns the Akaike Information Criterion for the last estimated model.

$bic

Output: scalar

Returns Schwarz's Bayesian Information Criterion for the last estimated model.

$coeff

Output:	scalar or matrix
Argument:	*s* (name of coefficient, optional)

The `$coeff` accessor can be used in two ways: with no arguments, it returns a column vector containing the estimated coefficients for the last model. With the optional argument, it returns a scalar, which is the estimated parameter named *s*. See also $stderr, $vcv.

Example:

```
open bjg
arima 0 1 1 ; 0 1 1 ; lg
b = $coeff
macoef = $coeff(theta_1)
```

If the "model" in question is actually a system (a VAR or VECM, or system of simultaneous equations), `$coeff` with no parameters returns the matrix of coefficients, one column per equation.

$compan

Output: matrix

Must follow the estimation of a VAR or a VECM; returns the companion matrix.

$datatype

Output: scalar

Returns an integer value representing the sort of dataset that is currently loaded: 0 = no data; 1 = cross-sectional (undated) data; 2 = time-series data; 3 = panel data.

$df

Output: scalar

Returns the degrees of freedom of the last estimated model.

$ess

Output: scalar

Returns the error sum of squares of the last estimated model.

$gmmcrit

Output: scalar

Must follow a `gmm` block. Returns the value of the objective function at its minimum.

$h

Output: series

Must follow a `garch` command. Returns the estimated conditional variance.

$hausman

Output: row vector

Must follow a `tsls` command. Returns a 1×3 vector, containing the value of the Hausman test statistic, the corresponding degrees of freedom and p-value, in this order.

$hqc

Output: scalar

Returns the Hannan-Quinn Information Criterion for the last estimated model.

$jalpha

Output: matrix

Must follow the estimation of a VECM, and returns the loadings matrix. It has as many rows as variables in the VECM and as many columns as the cointegration rank.

$jbeta

Output: matrix

Must follow the estimation of a VECM, and returns the cointegration matrix. It has as many rows as variables in the VECM (plus the number of exogenous variables that are restricted to the cointegration space, if any), and as many columns as the cointegration rank.

$jvbeta

Output: square matrix

Must follow the estimation of a VECM, and returns the estimated covariance matrix for the elements of the cointegration vectors.

In the case of unrestricted estimation, it has a number of rows equal to the unrestricted elements of the cointegration space after the Phillips normalization. If, however, a restricted system is estimated via the `restrict` command with the `--full` option, a singular matrix with $(n + m)r$ rows will be returned (*n* being the number of endogenous variables, *m* the number of exogenous variables that are restricted to the cointegration space, and *r* the cointegration rank).

Example: the code

```
open denmark.gdt
vecm 2 1 LRM LRY IBO IDE --rc --seasonals -q
s0 = $jvbeta

restrict --full
b[1,1] = 1
b[1,2] = -1
b[1,3] + b[1,4] = 0
end restrict
s1 = $jvbeta

print s0
print s1
```

produces the following output.

```
s0 (4 x 4)

  0.019751      0.029816  -0.00044837     -0.12227
  0.029816       0.31005     -0.45823     -0.18526
-0.00044837     -0.45823       1.2169    -0.035437
  -0.12227      -0.18526    -0.035437      0.76062

s1 (5 x 5)

0.0000       0.0000       0.0000       0.0000       0.0000
0.0000       0.0000       0.0000       0.0000       0.0000
0.0000       0.0000      0.27398     -0.27398    -0.019059
0.0000       0.0000     -0.27398      0.27398     0.019059
0.0000       0.0000    -0.019059     0.019059    0.0014180
```

$lnl

Output: scalar

Returns the log-likelihood for the last estimated model (where applicable).

$ncoeff

Output: scalar

Total number of coefficients estimated in the last model.

$nobs

Output: scalar

Returns the number of observations in the currently selected sample.

$nvars

Output: scalar

Returns the number of variables in the dataset (including the constant).

$pd

Output: scalar

Returns the frequency or periodicity of the data (e.g. 4 for quarterly data).

$pvalue

Output: scalar

Returns the p-value of the test statistic that was generated by the last explicit hypothesis-testing command, if any (e.g. `chow`). See *Manual de Utilização do Gretl* for details. See also $test.

$rho

Output:	scalar
Argument:	n (scalar, optional)

Without arguments, returns the first-order autoregressive coefficient for the residuals of the last model. After estimating a model via the `ar` command, the syntax `$rho(n)` returns the corresponding estimate of $\rho(n)$.

$rsq

Output: scalar

Returns the unadjusted R^2 from the last estimated model.

$sample

Output: series

Must follow estimation of a single-equation model. Returns a dummy series with value 1 for observations used in estimation, 0 for observations within the currently defined sample range but not used (presumably because of missing values), and NA for observations outside of the current range.

If you wish to compute statistics based on the sample that was used for a given model, you can do, for example:

```
ols y 0 xlist
genr sdum = $sample
smpl sdum --dummy
```

$sargan

Output: row vector

Must follow a `tsls` command. Returns a 1×3 vector, containing the value of the Sargan over-identification test statistic, the corresponding degrees of freedom and p-value, in this order.

$sigma

Output: scalar

Returns the standard error of the residuals (or Standard Error of Estimate) from the last model.

$stderr

Output:	scalar or matrix
Argument:	s (name of coefficient, optional)

The `$stderr` accessor can be used in two ways: with no arguments, it returns a column vector containing the standard error of the coefficients for the last model. With the optional argument, it returns a scalar, namely the standard error of the parameter called s. See also $coeff, $vcv.

$stopwatch

Output: scalar

Must be preceded by `set stopwatch`, which activates the measurement of CPU time. The first use of this accessor yields the seconds of CPU time that have elapsed since the `set stopwatch` command.

At each access the clock is reset, so subsequent uses of `$stopwatch` yield the seconds of CPU time since the previous access.

$T

Output: scalar

Number of observations used in estimating the last model.

$t1

Output: scalar

The 1-based index of the first observation in the currently selected sample.

$t2

Output: scalar

The 1-based index of the last observation in the currently selected sample.

$test

Output: scalar

Returns the value of the test statistic that was generated by the last explicit hypothesis-testing command, if any (e.g. `chow`). See *Manual de Utilização do Gretl* for details. See also $pvalue.

$trsq

Output: scalar

Returns TR^2 (sample size times R-squared) from the last model.

$uhat

Output: series

Returns the residuals from the last model. This may have different meanings for different estimators. For example, after an ARMA estimation `$uhat` will contain the one-step-ahead forecast error; after a probit model, it will contain the generalized residuals.

If the "model" in question is actually a system (a VAR or VECM, or system of simultaneous equations), `$uhat` with no parameters retrieves the matrix of residuals, one column per equation.

$unit

Output: series

Valid for panel datasets only. Returns a series with value 1 for all observations on the first unit or group, 2 for observations on the second unit, and so on.

$vcv

Output:	scalar or matrix
Arguments:	*s1* (name of coefficient, optional)
	s2 (name of coefficient, optional)

The `$stderr` accessor can be used in two ways: with no arguments, it returns a square matrix containing the estimated covariance matrix for the coefficients of the last model. With the optional arguments, it returns a scalar, which is the estimated covariance between the parameters named *s1* and *s2*. See also $coeff, $stderr.

If the "model" in question is actually a system (a VAR or VECM, or system of simultaneous equations), `$vcv` with no parameters returns the cross-equation covariance matrix.

$version

Output:	scalar

Returns an integer value that codes for the program version. The gretl version string takes the form `x.y.z` (for example, 1.7.6). The return value from this accessor is formed as `10000*x + 100*y + z`, so that 1.7.6 translates as 10706.

$windows

Output:	scalar

Returns 1 if gretl is running on MS Windows, otherwise 0. By conditioning on the value of this variable you can write shell calls that are portable across different operating systems.

Also see the shell command.

$xlist

Output:	list

Returns the list of regressors from the last model (for single-equation models only).

$yhat

Output:	series

Returns the fitted values from the last regression.

2.3 Functions proper

abs

Output:	same type as input
Argument:	*x* (scalar, series or matrix)

Absolute value.

acos

Output:	same type as input
Argument:	*x* (scalar, series or matrix)

The arc cosine of *x*, that is, the value whose cosine is *x*. The result is in radians; the input should be in the range -1 to 1.

asin

Output:	same type as input
Argument:	*x* (scalar, series or matrix)

The arc sine of *x*, that is, the value whose sine is *x*. The result is in radians; the input should be in the range -1 to 1.

atan

Output:	same type as input
Argument:	*x* (scalar, series or matrix)

The arc tangent of *x*, that is, the value whose tangent is *x*. The result is in radians.

BFGSmax

Output:	scalar
Arguments:	*b* (vector)
	s (string)

Numerical maximization via the method of Broyden, Fletcher, Goldfarb and Shanno. The vector b should hold the initial values of a set of parameters, and the string s should specify a call to a function that calculates the (scalar) criterion to be maximized, given the current parameter values and any other relevant data. If the object is in fact minimization, this function should return the negative of the criterion. On successful completion, `BFGSmax` returns the maximized value of the criterion, and b holds the parameter values which produce the maximum.

For more details and examples see the chapter on special functions in `genr` in *Manual de Utilização do Gretl*. See also fdjac.

bkfilt

Output:	series
Argument:	y (series)

Extracts the cyclical component of series y via the Baxter–King bandpass filter, a two-sided symmetric filter. See *Manual de Utilização do Gretl* for details. See also hpfilt.

cdemean

Output:	matrix
Argument:	X (matrix)

Centers the columns of matrix X around their means.

cdf

Output:	same type as input
Arguments:	c (character)
	... (see below)
	x (scalar, series or matrix)
Examples:	`p1 = cdf(N, -2.5)`
	`p2 = cdf(X, 3, 5.67)`
	`p3 = cdf(D, 0.25, -1, 1)`

Cumulative distribution function calculator. Returns $P(X \leq x)$, where the distribution X is determined by the character c. Between the arguments c and x, zero or more additional arguments are required to specify the parameters of the distribution, as follows.

Distribution	*c*	*Arg 2*	*Arg 3*
Standard normal	`z`, `n` or `N`	–	–
Bivariate normal	`D`	ρ	–
Student's t (central)	`t`	degrees of freedom	–
Chi square	`c`, `x` or `X`	degrees of freedom	–
Snedecor's F	`f` or `F`	df (num.)	df (den.)
Gamma	`g` or `G`	shape	scale
Binomial	`b` or `B`	probability	trials
Poisson	`p` or `P`	mean	–
Weibull	`w` or `W`	shape	scale

Note that most cases have aliases to help memorizing the codes. The bivariate normal case is special: the syntax is `x = cdf(D, rho, z1, z2)` where `rho` is the correlation between the variables `z1` and `z2`.

The parametrization gretl uses for the Gamma random variate implies that its density function can be written as

$$f(x; k, \theta) = \frac{x^{k-1}}{\theta^k} \frac{e^{-x/\theta}}{\Gamma(k)}$$

where $k > 0$ is the shape parameter and $\theta > 0$ is the scale parameter.

See also pdf, critical, invcdf, pvalue.

cdiv

Output:	matrix
Arguments:	X (matrix)
	Y (matrix)

Complex division. The two arguments must have the same number of rows, n, and either one or two columns. The first column contains the real part and the second (if present) the imaginary part. The return value is an $n \times 2$ matrix or, if the result has no imaginary part, an n-vector. See also cmult.

ceil

Output:	same type as input
Argument:	x (scalar, series or matrix)

Ceiling function: returns the smallest integer greater than or equal to x. See also floor, int.

cholesky

Output:	square matrix
Argument:	A (square matrix)

Peforms a Cholesky decomposition of the matrix A, which is assumed to be symmetric and positive definite. The result is a lower-triangular matrix K which satisfies $A = KK'$. The function will fail if A is not symmetric or not positive definite.

cmult

Output:	matrix
Arguments:	X (matrix)
	Y (matrix)

Complex multiplication. The two arguments must have the same number of rows, n, and either one or two columns. The first column contains the real part and the second (if present) the imaginary part. The return value is an $n \times 2$ matrix, or, if the result has no imaginary part, an n-vector. See also cdiv.

cnorm

Output:	same type as input
Argument:	x (scalar, series or matrix)

Returns the cumulative distribution function for a standard normal. See also dnorm, qnorm.

colnames

Output:	scalar
Arguments:	M (matrix)
	s (named list or string)

Attaches names to the columns of the $T \times k$ matrix M. If s is a named list, the column names are copied from the names of the variables; the list must have k members. If s is a string, it should contain k space-separated sub-strings. The return value is 0 on successful completion, non-zero on error.

cols

Output:	scalar
Argument:	X (matrix)

The number of columns of X. See also mshape, rows, unvech, vec, vech.

corr

Output:	scalar
Arguments:	*y1* (series)
	y2 (series)

Computes the correlation coefficient between *y1* and *y2*. See also cov, mcov, mcorr.

cos

Output:	same type as input
Argument:	*x* (scalar, series or matrix)

Cosine.

cov

Output:	scalar
Arguments:	*y1* (series)
	y2 (series)

Computes the covariance between *y1* and *y2*. See also corr, mcov, mcorr.

critical

Output:	same type as input
Arguments:	*c* (character)
	... (see below)
	p (scalar, series or matrix)
Examples:	`c1 = critical(t, 20, 0.025)`
	`c2 = critical(F, 4, 48, 0.05)`

Critical value calculator. Returns x such that $P(X > x) = p$, where the distribution X is determined by the character c. Between the arguments c and p, zero or more additional arguments are required to specify the parameters of the distribution, as follows.

Distribution	*c*	*Arg 2*	*Arg 3*
Standard normal	`z`, `n` or `N`	–	–
Student's t (central)	`t`	degrees of freedom	–
Chi square	`c`, `x` or `X`	degrees of freedom	–
Snedecor's F	`f` or `F`	df (num.)	df (den.)
Binomial	`b` or `B`	p	n

See also cdf, invcdf, pvalue.

cum

Output:	same type as input
Argument:	*x* (series or matrix)

Cumulates *x*. When x is a series, produces a series $y_t = \sum_{s=m}^{t} x_s$; the starting point of the summation, m, is the first non-missing observation of the currently selected sample. If any missing values are encountered in x, subsequent values of y will be set to missing. When x is a matrix, its elements are cumulated by columns.

See also diff.

det

Output:	scalar
Argument:	*A* (square matrix)

Returns the determinant of A, computed via the LU factorization. See also ldet, rcond.

diag

Output:	matrix
Argument:	X (matrix)

Returns the principal diagonal of X in a column vector. Note: if X is an $m \times n$ marix, the number of elements of the output vector is $\min(m, n)$. See also tr.

diff

Output:	same type as input
Argument:	y (series, matrix or list)

Computes first differences. If y is a series, or a list of series, starting values are set to `NA`. If y is a matrix, differencing is done by columns and starting values are set to 0.

When a list is returned, the individual variables are automatically named according to the template `d_`*varname* where *varname* is the name of the original series. The name is truncated if necessary, and may be adjusted in case of non-uniqueness in the set of names thus constructed.

See also cum, ldiff, sdiff.

dnorm

Output:	same type as input
Argument:	x (scalar, series or matrix)

Returns the density of the standard normal distribution at x. To get the density for a non-standard normal distribution at x, pass the z-score of x to the `dnorm` function and multiply the result by the Jacobian of the z transformation, namely 1 over σ, as illustrated below:

```
mu = 100
sigma = 5
x = 109
fx = (1/sigma) * dnorm((x-mu)/sigma)
```

See also cnorm, qnorm.

dsort

Output:	same type as input
Argument:	x (series or vector)

Sorts x in descending order, skipping observations with missing values when x is a series. See also sort, values.

dummify

Output:	list
Argument:	x (series or list)

The argument x should be a discrete series, or list of such series. This function creates a set of dummy variables coding for the distinct values in the series; the smallest value is taken as the omitted category and is not explicitly represented.

The generated variables are automatically named according to the template `D`*varname*_*i* where *varname* is the name of the original series and i is a 1-based index. The original portion of the name is truncated if necessary, and may be adjusted in case of non-uniqueness in the set of names thus constructed.

eigengen

Output:	matrix
Arguments:	A (square matrix)
	$\&U$ (reference to matrix, or `null`)

Computes the eigenvalues, and optionally the right eigenvectors, of the $n \times n$ matrix A. If all the eigenvalues are real, an $n \times 1$ matrix is returned; otherwise, the result is an $n \times 2$ matrix, the first column holding the real components and the second column the imaginary components.

The second argument must be either the name of an existing matrix preceded by `&` (to indicate the "address" of the matrix in question), in which case an auxiliary result is written to that matrix, or the keyword `null`, in which case the auxiliary result is not produced.

If a non-null second argument is given, the specified matrix will be over-written with the auxiliary result. (It is not required that the existing matrix be of the right dimensions to receive the result.) It will be organized as follows:

- If the i-th eigenvalue is real, the i-th column of U will contain the corresponding eigenvector;
- If the i-th eigenvalue is complex, the i-th column of U will contain the real part of the corresponding eigenvector and the next column the imaginary part. The eigenvector for the conjugate eigenvalue is the conjugate of the eigenvector.

In other words, the eigenvectors are stored in the same order as the eigenvalues, but the real eigenvectors occupy one column, whereas complex eigenvectors take two (the real part comes first); the total number of columns is still n, because the conjugate eigenvector is skipped.

See also eigensym, qrdecomp, svd.

eigensym

Output:	column vector
Arguments:	A (square matrix)
	$\&U$ (reference to matrix, or `null`)

Computes the eigenvalues, and optionally the right eigenvectors, of the $n \times n$ symmetrix matrix A; the second argument must be either the name of an existing matrix preceded by `&` (to indicate the "address" of the matrix in question), in which case an auxiliary result is written to that matrix, or the keyword `null`, in which case the auxiliary result is not produced.

If the second argument is not `null`, the specified matrix will be over-written with the auxiliary result. (It is not required that the existing matrix be of the right dimensions to receive the result.)

See also eigengen, qrdecomp, svd.

exp

Output:	same type as input
Argument:	x (scalar, series or matrix)

Exponential. Note: in case of matrices, the function acts element by element. For the matrix exponential function, see mexp.

fdjac

Output:	matrix
Arguments:	b (column vector)
	s (string)

Calculates the (forward-difference approximation to the) Jacobian associated with the vector b and the transformation function defined by the function call in the string s. For more details and examples see the chapter on special functions in `genr` in *Manual de Utilização do Gretl*.

See also BFGSmax.

fft

Output:	matrix
Argument:	X (matrix)

Discrete real Fourier transform. If the input matrix X has n columns, the output has $2n$ columns, where the real parts are stored in the odd columns and the complex parts in the even ones.

Should it be necessary to compute the Fourier transform on several vectors with the same number of elements, it is numerically more efficient to group them into a matrix rather than invoking `fft` for each vector separately. See also ffti.

ffti

Output:	matrix
Argument:	X (matrix)

Inverse discrete real Fourier transform. It is assumed that X contains n complex column vectors, with the real part in the odd columns and the imaginary part in the even ones, so the total number of columns should be $2n$. A matrix with n columns is returned.

Should it be necessary to compute the inverse Fourier transform on several vectors with the same number of elements, it is numerically more efficient to group them into a matrix rather than invoking `ffti` for each vector separately. See also fft.

firstobs

Output:	scalar
Argument:	y (series)

First non-missing observation for the variable y. Note that if some form of subsampling is in effect, the value returned may be smaller than the dollar variable $t1. See also lastobs.

floor

Output:	same type as input
Argument:	y (scalar, series or matrix)

Floor function: returns the greatest integer less than or equal to x. Note: int and `floor` differ in their effect for negative arguments: `int(-3.5)` gives -3, while `floor(-3.5)` gives -4.

fracdiff

Output:	series
Arguments:	y (series)
	d (scalar)

$$\Delta^d y_t = y_t - \sum_{i=1}^{\infty} \psi_i y_{t-i}$$

where

$$\psi_i = \frac{\Gamma(i-d)}{\Gamma(-d)\Gamma(i+1)}$$

Note that in theory fractional differentiation is an infinitely long filter. In practice, presample values of y_t are assumed to be zero.

gammafun

Output:	same type as input
Argument:	x (scalar, series or matrix)

Returns the gamma function of x.

genpois

Output:	series
Argument:	μ (scalar or series)

Generates a series of Poisson pseudo-random variates. If μ is a scalar, all the elements are drawn from the same distribution

$$P(x_t = a) = e^{-\mu} \frac{\mu^a}{a!}$$

Otherwise, if μ is a series, the above becomes

$$P(x_t = a) = e^{-\mu_t} \frac{\mu_t^a}{a!}$$

See also randgen, normal, uniform, mnormal, muniform.

getenv

Output:	string
Argument:	s (string)

If an environment variable by the name of s is defined, returns the value of that variable, otherwise returns an empty string.

gini

Output:	scalar
Argument:	y (series)

Returns Gini's inequality index for the series y.

ginv

Output:	matrix
Argument:	A (matrix)

Returns A^+, the Moore–Penrose or generalized inverse of A, computed via the singular value decomposition.

This matrix has the properties

$$\begin{aligned} AA^+A &= A \\ A^+AA^+ &= A^+ \end{aligned}$$

Moreover, the products A^+A and AA^+ are symmetric by construction.

See also inv, svd.

hpfilt

Output:	series
Argument:	y (series)

Returns the cycle from the Hodrick–Prescott filter applied to series y. See *Manual de Utilização do Gretl* for details. See also bkfilt.

I

Output:	square matrix
Argument:	n (scalar)

Returns an identity matrix with n rows and columns.

imaxc

Output:	row vector
Argument:	X (matrix)

Returns the row indices of the maxima of the columns of X.

See also imaxr, iminc, maxc.

imaxr

Output:	column vector
Argument:	X (matrix)

Returns the column indices of the maxima of the rows of X.

See also imaxc, iminr, maxr.

iminc

Output:	row vector
Argument:	X (matrix)

Returns the row indices of the minima of the columns of X.

See also iminr, imaxc, minc.

iminr

Output:	column vector
Argument:	X (matrix)

Returns the column indices of the mimima of the rows of X.

See also iminc, imaxr, minr.

infnorm

Output:	scalar
Argument:	X (matrix)

Returns the ∞-norm of the $r \times c$ matrix X, namely,

$$\|X\|_\infty = \max_i \sum_{j=1}^{c} |X_{ij}|$$

See also onenorm.

int

Output:	same type as input
Argument:	x (scalar, series or matrix)

Truncates the fractional part of x. Note: `int` and floor differ in their effect for negative arguments: `int(-3.5)` gives -3, while `floor(-3.5)` gives -4. See also ceil.

inv

Output:	matrix
Argument:	A (square matrix)

Returns the inverse of A. If A is singular or not square, an error message is produced and nothing is returned. Note that gretl checks automatically the structure of A and uses the most efficient numerical procedure to perform the inversion.

The matrix types gretl checks for are: identity; diagonal; symmetric and positive definite; symmetric but not positive definite; and triangular.

See also ginv, invpd.

invcdf

Output:	same type as input
Arguments:	c (character)
	... (see below)
	p (scalar, series or matrix)

Inverse cumulative distribution function calculator. Returns x such that $P(X \leq x) = p$, where the distribution X is determined by the character c; Between the arguments c and p, zero or more additional arguments are required to specify the parameters of the distribution, as follows.

Distribution	code, c	*Arg* 2	*Arg* 3
Standard normal	`z`, `n` or `N`	–	–
Student's t (central)	`t`	degrees of freedom	–
Chi square	`c`, `x` or `X`	degrees of freedom	–
Snedecor's F	`f` or `F`	df (num.)	df (den.)
Binomial	`b` or `B`	p	n

See also cdf, critical, pvalue.

invpd

Output:	square matrix
Argument:	A (symmetric matrix)

Returns the inverse of the symmetric, positive definite matrix A. This function is slightly faster than inv for large matrices, since no check for symmetry is performed; for that reason it should be used with care.

islist

Output:	scalar
Argument:	s (string)

Returns 1 if s is the identifier for a currently defined list, otherwise 0. See also isnull, isseries, isstring.

isnull

Output:	scalar
Argument:	s (string)

Returns 0 if s is the identifier for a currently defined object, be it a scalar, a series, a matrix, list or string; otherwise returns 1. See also islist, isseries, isstring.

isseries

Output:	scalar
Argument:	s (string)

Returns 1 if s is the identifier for a currently defined series, otherwise 0. See also islist, isnull, isstring.

isstring

Output:	scalar
Argument:	s (string)

Returns 1 if s is the identifier for a currently defined string, otherwise 0. See also islist, isnull, isseries.

lags

Output:	list
Arguments:	p (scalar)
	y (series or list)

Generates lags 1 to p of the series y, or if y is a list, of all variables in the list. If $p = 0$, the maximum lag defaults to the periodicity of the data; otherwise p must be positive.

The generated variables are automatically named according to the template *varname*_*i* where *varname* is the name of the original series and i is the specific lag. The original portion of the name is truncated if necessary, and may be adjusted in case of non-uniqueness in the set of names thus constructed.

lastobs

Output:	scalar
Argument:	y (series)

Last non-missing observation for the variable y. Note that if some form of subsampling is in effect, the value returned may be larger than the dollar variable $t2. See also firstobs.

ldet

Output:	scalar
Argument:	A (square matrix)

Returns the natural log of the determinant of A, computed via the LU factorization. See also det, rcond.

ldiff

Output:	same type as input
Argument:	y (series or list)

Computes log differences; starting values are set to `NA`.

When a list is returned, the individual variables are automatically named according to the template `ld_`*varname* where *varname* is the name of the original series. The name is truncated if necessary, and may be adjusted in case of non-uniqueness in the set of names thus constructed.

See also diff, sdiff.

lincomb

Output:	series
Arguments:	L (list)
	b (vector)

Computes a new series as a linear combination of the series in the list L. The coefficients are given by the vector b, which must have length equal to the number of series in L.

See also wmean.

ljungbox

Output:	scalar
Arguments:	y (series)
	p (scalar)

Computes the Ljung–Box Q' statistic for the series y using lag order p. The currently defined sample range is used. The lag order must be greater than or equal to 1 and less than the number of available observations.

This statistic may be referred to the chi-square distribution with p degrees of freedom as a test of the null hypothesis that the series y is serially independent. See also pvalue.

lngamma

Output:	same type as input
Argument:	x (scalar, series or matrix)

Log of the gamma function of x.

log

Output:	same type as input
Argument:	x (scalar, series, matrix or list)

Natural logarithm; produces `NA` for non-positive values. Note: `ln` is an acceptable alias for `log`.

When a list is returned, the individual variables are automatically named according to the template `l_`*varname* where *varname* is the name of the original series. The name is truncated if necessary, and may be adjusted in case of non-uniqueness in the set of names thus constructed.

log10

Output:	same type as input
Argument:	x (scalar, series or matrix)

Base-10 logarithm; produces `NA` for non-positive values.

log2

Output:	same type as input
Argument:	x (scalar, series or matrix)

Base-2 logarithm; produces `NA` for non-positive values.

lower

Output:	square matrix
Argument:	A (matrix)

Returns an $n \times n$ lower triangular matrix B for which $B_{ij} = A_{ij}$ if $i \geq j$, and 0 otherwise.

See also upper.

lrvar

Output:	scalar
Arguments:	y (series)
	k (scalar)

Returns the long-run variance of y, calculated using a Bartlett kernel with window size k. If k is negative, `int(T^(1/3))` is used.

In formulae:

$$\hat{\omega}^2(k) = \frac{1}{T}\sum_{t=k}^{T-k}\left[\sum_{i=-k}^{k} w_i (y_t - \bar{X})(y_{t-i} - \bar{Y})\right]$$

with

$$w_i = 1 - \frac{|i|}{k+1}$$

makemask

Output:	column vector
Argument:	y (series)

Produces a column vector containing the observation numbers corresponding to the non-zero entries in the series y. This function is typically useful for filtering out rows of a matrix built from data series.

max

Output:	scalar or series
Argument:	y (series or list)

If the argument y is a series, returns the (scalar) maximum of the non-missing observations in the series. If the argument is a list, returns a series each of whose elements is the maximum of the values of the listed variables at the given observation.

maxc

Output:	row vector
Argument:	X (matrix)

Returns the maxima of the columns of X.

See also imaxc, maxr, minc.

maxr

Output:	column vector
Argument:	X (matrix)

Returns the maxima of the rows of X.

See also imaxc, maxc, minr.

mcorr

Output:	matrix
Argument:	X (matrix)

Computes a correlation matrix treating each column of X as a variable. See also corr, cov, mcov.

mcov

Output:	matrix
Argument:	X (matrix)

Computes a covariance matrix treating each column of X as a variable. See also corr, cov, mcorr.

mean

Output:	scalar or series
Argument:	x (series or list)

If x is a series, returns the (scalar) sample mean, skipping any missing observations.

If x is a list, returns a series y such that y_t is the mean of the values of the variables in the list at observation t, or `NA` if there are any missing values at t.

meanc

Output:	row vector
Argument:	X (matrix)

Returns the means of the columns of X. See also meanr, sumc, sdc.

meanr

Output:	column vector
Argument:	X (matrix)

Returns the means of the rows of X. See also meanc, sumr.

median

Output:	scalar
Argument:	y (series)

The median of the non-missing observations in series y. See also quantile.

mexp

Output:	square matrix
Argument:	A (square matrix)

Matrix exponential,

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!} = \frac{I}{0!} + \frac{A}{1!} + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots$$

(This series is sure to converge.) The algorithm used is 11.3.1 from Golub and Van Loan (1996).

min

Output:	scalar or series
Argument:	y (series or list)

If the argument y is a series, returns the (scalar) minimum of the non-missing observations in the series. If the argument is a list, returns a series each of whose elements is the minimum of the values of the listed variables at the given observation.

minc

Output:	row vector
Argument:	X (matrix)

Returns the minima of the columns of X.

See also iminc, maxc, minr.

minr

Output:	column vector
Argument:	X (matrix)

Returns the minima of the rows of X.

See also iminr, maxr, minc.

missing

Output:	same type as input
Argument:	x (scalar or series)

Returns a binary variable holding 1 if x is `NA`. If x is a series, the comparison is done element by element. See also misszero, ok, zeromiss.

misszero

Output:	same type as input
Argument:	x (scalar or series)

Converts `NA`s to zeros. If x is a series, the conversion is done element by element. See also missing, ok, zeromiss.

mlag

Output:	matrix
Arguments:	X (matrix)
	p (scalar)

Shifts up or down the elements of X. If $p > 0$ the returned matrix Y has typical element $Y_{i,j} = X_{i-p,j}$ for $i \geq p$ and zero otherwise. In other words, the columns of X are shifted down by p rows and the first p rows are filled with zeros. If p is a negative number, X is shifted up and the last rows are filled with zeros.

mnormal

Output:	matrix
Arguments:	r (scalar)
	c (scalar)

Returns a matrix with r rows and c columns, filled with standard normal pseudo-random variates. See also normal, muniform.

mols

Output:	matrix
Arguments:	Y (matrix)
	X (matrix)
	$\&U$ (reference to matrix, or `null`)

Returns a $k \times n$ matrix of parameter estimates obtained by OLS regression of the $T \times n$ matrix Y on the $T \times k$ matrix X. The Cholesky decomposition is used. If the third argument is not `null`, the $T \times n$ matrix U will contain the residuals.

movavg

Output:	series
Arguments:	x (series)
	p (scalar)

Computes the p-term moving average for the series x, that is $y_t = \frac{1}{p}\sum_{i=0}^{p-1} x_{t-i}$.

Note that the result is not centered. If you want a centered moving average, you can use the lead operator on the returned series. Example:

```
tmp = movavg(x,3)
y = tmp(+1)
```

mread

Output:	matrix
Argument:	s (string)

Reads a matrix from a text file. The string s must contain the name of the (plain text) file from which the matrix is to be read. The file in question must conform to the following rules:

- The columns must be separated by spaces or tab characters.
- The decimal separator must be the dot character, "`.`".
- The first line in the file must contain two integers, separated by a space or a tab, indicating the number of rows and columns, respectively.

Should an error occur (such as the file being badly formatted or inaccessible), an empty matrix is returned.

See also mwrite.

mshape

Output:	matrix
Arguments:	X (matrix)
	r (scalar)
	c (scalar)

Rearranges the elements of X into a matrix with r rows and c columns. Elements are read from X and written to the target in column-major order. If X contains fewer than $k = rc$ elements, the elements are repeated cyclically; otherwise, if X has more elements, only the first k are used.

See also cols, rows, unvech, vec, vech.

msortby

Output:	matrix
Arguments:	X (matrix)
	j (scalar)

Returns a matrix in which the rows of X are reordered by increasing value of the elements in column j.

muniform

Output:	matrix
Arguments:	r (scalar)
	c (scalar)

Returns a matrix with r rows and c columns, filled with uniform (0,1) pseudo-random variates. Note: the preferred method for generating a scalar uniform r.v. is recasting the output of `muniform` to a scalar, as in

```
scalar x = muniform(1,1)
```

See also mnormal, uniform.

mwrite

Output:	scalar
Arguments:	X (matrix)
	s (string)

Writes the matrix X to a plain text file named s. The file will contain on the first line two integers, separated by a tab character, with the number of rows and columns; on the next lines, the matrix elements in scientific notation, separated by tabs (one line per row).

If file s already exists, it will be overwritten. The return value is 0 on successful completion; if an error occurs, such as the file being unwritable, the return value will be non-zero.

Matrices stored via the `mwrite` command can be easily read by other programs; see *Manual de Utilização do Gretl* for details.

See also mread.

mxtab

Output:	matrix
Arguments:	x (series or vector)
	y (series or vector)

Returns a matrix holding the cross tabulation of the values contained in x (by row) and y (by column). The two arguments should be of the same type (both series or both column vectors), and because of the typical usage of this function, are assumed to contain integer values only.

See also values.

nelem

Output:	scalar
Argument:	L (list)

Returns the number of items in list L.

nobs

Output:	scalar
Argument:	y (series)

Returns the number of non-missing observations for the variable y in the currently selected sample.

normal

Output:	series
Arguments:	μ (scalar)
	σ (scalar)

Generates a series of Gaussian pseudo-random variates with mean μ and standard deviation σ. If no arguments are supplied, standard normal variates $N(0,1)$ are produced.

See also randgen, normal, genpois, mnormal, muniform.

nullspace

Output:	matrix
Argument:	A (matrix)

Computes the right nullspace of A, via the singular value decomposition: the result is a matrix B such that $AB = [0]$, except when A has full column rank, in which case an empty matrix is returned. Otherwise, if A is $m \times n$, B will be an $n \times (n - r)$ matrix, where r is the rank of A.

See also rank, svd.

obs

Output:	series

Returns a series of consecutive integers, setting 1 at the start of the dataset. Note that the result is invariant to subsampling. This function is especially useful with time-series datasets. Note: you can write `t` instead of `obs` with the same effect.

See also obsnum.

obsnum

Output:	scalar
Argument:	s (string)

Returns an integer corresponding to the observation specified by the string s. Note that the result is invariant to subsampling. This function is especially useful with time-series datasets. For example, the following code

```
open denmark
k = obsnum(1980:1)
```

yields `k = 25`, indicating that the first quarter of 1980 is the 25th observation in the `denmark` dataset.

See also obs.

ok

Output:	same type as input
Argument:	x (scalar, series or list)

Returns a binary variable holding 1 if x is not `NA`. If x is a series, the comparison is done element by element. If x is a list of series, the output is a series with 0 at the observations for which at least one series in the list is missing, and 1 otherwise.

See also missing, misszero, zeromiss.

onenorm

Output:	scalar
Argument:	X (matrix)

Returns the 1-norm of the $r \times c$ matrix X:

$$\|X\|_1 = \max_j \sum_{i=1}^{r} |X_{ij}|$$

See also infnorm, rcond.

ones

Output:	matrix
Arguments:	r (scalar)
	c (scalar)

Outputs a matrix with r rows and c columns, filled with ones.

See also seq, zeros.

orthdev

Output:	series
Argument:	y (series)

Only applicable if the currently open dataset has a panel structure. Computes the forward orthogonal deviations for variable y, that is

$$\tilde{y}_{i,t} = \sqrt{\frac{T_i - t}{T_i - t + 1}} \left(y_{i,t} - \frac{1}{T_i - t} \sum_{s=t+1}^{T_i} y_{i,s} \right)$$

This transformation is sometimes used instead of differencing to remove individual effects from panel data. For compatibility with first differences, the deviations are stored one step ahead of their true temporal location (that is, the value at observation t is the deviation that, strictly speaking, belongs at $t - 1$). That way one loses the first observation in each time series, not the last.

See also diff.

pdf

Output:	same type as input
Arguments:	c (character)
	. . . (see below)
	x (scalar, series or matrix)
Examples:	`f1 = pdf(N, -2.5)`
	`f2 = pdf(X, 3, y)`
	`f3 = pdf(W, shape, scale, y)`

Probability density function calculator. Returns the density at x of the distribution identified by the code c. See cdf for details of the required arguments. The distributions supported by the `pdf` function are the normal, Student's t, chi-square, F, Gamma and Weibull.

For the normal distribution, see also dnorm.

pmax

Output:	series
Argument:	y (series)

Only applicable if the currently open dataset has a panel structure. Returns the per-unit maximum for variable y.

Missing values are skipped. See also pmin, pmean, pnobs, psd.

pmean

Output:	series
Argument:	y (series)

Only applicable if the currently open dataset has a panel structure. Computes the per-unit mean for variable y; that is,

$$\bar{y}_i = \frac{1}{T_i}\sum_{t=1}^{T_i} y_{i,t}$$

where T_i is the number of valid observations for unit i.

Missing values are skipped. See also pmax, pmin, pnobs, psd.

pmin

Output:	series
Argument:	y (series)

Only applicable if the currently open dataset has a panel structure. Returns the per-unit minimum for variable y.

Missing values are skipped. See also pmax, pmean, pnobs, psd.

pnobs

Output:	series
Argument:	y (series)

Only applicable if the currently open dataset has a panel structure. Returns for each unit the number of non-missing cases for the variable y.

Missing values are skipped. See also pmax, pmin, pmean, psd.

polroots

Output:	matrix
Argument:	a (vector)

Finds the roots of a polynomial. If the polynomial is of degree p, the vector a should contain $p + 1$ coefficients in ascending order, i.e. starting with the constant and ending with the coefficient on x^p.

If all the roots are real they are returned in a column vector of length p, otherwise a $p \times 2$ matrix is returned, the real parts in the first column and the imaginary parts in the second.

princomp

Output:	matrix
Arguments:	X (matrix)
	p (scalar)

Let the matrix X be $T \times k$, containing T observations on k variables. The argument p must be a positive integer less than or equal to k. This function returns a $T \times p$ matrix, P, holding the first p principal components of X.

The elements of P are computed as

$$P_{tj} = \sum_{i=1}^{k} Z_{ti}\, v_i^{(j)}$$

where Z_{ti} is the standardized value of variable i at observation t, $Z_{ti} = (X_{ti} - \bar{X}_i)/\hat{\sigma}_i$, and $v^{(j)}$ is the jth eigenvector of the correlation matrix of the X_is, with the eigenvectors ordered by decreasing value of the corresponding eigenvalues.

See also eigensym.

psd

Output:	series
Argument:	y (series)

Only applicable if the currently open dataset has a panel structure. Computes the per-unit sample standard deviation for variable y, that is

$$\sigma_i = \sqrt{\frac{1}{T_i - 1} \sum_{t=1}^{T_i} (y_{i,t} - \bar{y}_i)^2}$$

The above formula holds for $T_i \geq 2$, where T_i is the number of valid observations for unit i; if $T_i = 0$, `NA` is returned; if $T_i = 1$, 0 is returned.

Note: this function makes it possible to check whether a given variable (say, `X`) is time-invariant via the condition `max(psd(X)) = 0`.

See also pmax, pmin, pmean, pnobs.

pvalue

Output:	same type as input
Arguments:	c (character)
	... (see below)
	x (scalar, series or matrix)
Examples:	`p1 = pvalue(z, 2.2)`
	`p2 = pvalue(X, 3, 5.67)`
	`p2 = pvalue(F, 3, 30, 5.67)`

P-value calculator. Returns $P(X > x)$, where the distribution X is determined by the character c. Between the arguments c and x, zero or more additional arguments are required to specify the parameters of the distribution; see cdf for details. The distributions supported by the `pval` function are the standard normal, t, Chi square, F, gamma, binomial, Poisson and Weibull.

See also critical, invcdf.

qform

Output:	matrix
Arguments:	x (matrix)
	A (symmetric matrix)

Computes the quadratic form $Y = xAx'$. Using this function instead of ordinary matrix multiplication guarantees more speed and better accuracy. If x and A are not conformable, or A is not symmetric, an error is returned.

qnorm

Output:	same type as input
Argument:	x (scalar, series or matrix)

Returns quantiles for the standard normal distribution. If x is not between 0 and 1, `NA` is returned. See also cnorm, dnorm.

qrdecomp

Output:	matrix
Arguments:	X (matrix)
	$\&R$ (reference to matrix, or `null`)

Computes the QR decomposition of an $m \times n$ matrix X, that is $X = QR$ where Q is an $m \times n$ orthogonal matrix and R is an $n \times n$ upper triangular matrix. The matrix Q is returned directly, while R can be retrieved via the optional second argument.

See also eigengen, eigensym, svd.

quantile

Output:	scalar or row vector
Arguments:	y (series or matrix)
	p (scalar between 0 and 1)

Given a series argument, returns the p-quantile for the series. For example, when $p = 0.5$, the median is returned. Given a matrix argument, returns a row vector containing the p-quantiles for the columns of y; that is, each column is treated as a series.

For a series of length n, the p-quantile, q, is defined as:

$$q = y_{[k]} + (n \cdot p - k)(y_{[k+1]} - y_{[k]})$$

where k is the integer part of $n \cdot p$ and $y_{[i]}$ is the i-th element of the series when sorted from smallest to largest.

rank

Output:	scalar
Argument:	X (matrix)

Returns the rank of X, numerically computed via the singular value decomposition. See also svd.

ranking

Output:	series
Argument:	y (series)

Returns a series with the ranks of y. The rank for observation i is the number of elements in the series that are less than y_i plus one half the number of elements in the series that are equal to y_i. (Intuitively, you may think of chess points, where victory gives you one point and a draw gives you half a point.) One is added so the lowest rank is 1 instead of 0.

Formally,

$$\mathrm{rank}(y_i) = 1 + \sum_{j \neq i} \left[I(y_j < y_i) + 0.5 \cdot I(y_j = y_i) \right]$$

where I denotes the indicator function.

See also sort, sortby.

randgen

Output:	series
Arguments:	*c* (character)
	a (scalar)
	b (scalar)
Examples:	`series x = randgen(u, 0, 100)`
	`series t14 = randgen(t, 14)`
	`series y = randgen(B, 0.6, 30)`
	`series g = randgen(G, 1, 1)`

All-purpose random number generator. The parameter *c* is a character, which specifies from which distribution the pseudo-random numbers should be drawn; *a* and, in some cases, *b* gauge the shape of the distribution.

Distribution	*c*	a	b
Uniform (continuous)	`u` or `U`	minimum	maximum
Normal	`z`, `n` or `N`	mean	standard deviation
Student's *t*	`t`	degrees of freedom	–
Chi square	`c`, `x` or `X`	degrees of freedom	–
Snedecor's *F*	`f` or `F`	df (num.)	df (den.)
Gamma	`g` or `G`	shape	scale
Binomial	`b` or `B`	*p*	*n*
Poisson	`p` or `P`	mean	–
Weibull	`w` or `W`	shape	scale

See also normal, uniform, genpois.

rcond

Output:	scalar
Argument:	*A* (square matrix)

Returns the reciprocal condition number for *A* with respect to the 1-norm. In many circumstances, this is a better measure of the sensitivity of *A* to numerical operations such as inversion than the determinant.

Given that *A* is non-singular, we may define

$$\kappa(A) = ||A||_1 \cdot ||A^{-1}||_1$$

This function returns $\kappa(A)^{-1}$.

See also det, ldet, onenorm.

readfile

Output:	string
Argument:	*fname* (string)

If a file by the name of *fname* exists and is readable, returns a string containing the content of this file, otherwise flags an error.

Also see the sscanf command.

resample

Output:	same type as input
Argument:	*x* (series or matrix)

Resamples from x with replacement. In the case of a series argument, each value of the returned series, y_t, is drawn from among all the values of x_t with equal probability. When a matrix argument is given, each row of the returned matrix is drawn from the rows of x with equal probability.

round

Output:	same type as input
Argument:	x (scalar, series or matrix)

Rounds to the nearest integer. Note that when x lies halfway between two integers, rounding is done "away from zero", so for example 2.5 rounds to 3, but `round(-3.5)` gives -4. This is a common convention in spreadsheet programs, but other software may yield different results. See also ceil, floor, int.

rows

Output:	scalar
Argument:	X (matrix)

Number of rows of the matrix X. See also cols, mshape, unvech, vec, vech.

sd

Output:	scalar or series
Argument:	x (series or list)

If x is a series, returns the (scalar) sample standard deviation, skipping any missing observations.

If x is a list, returns a series y such that y_t is the sample standard deviation of the values of the variables in the list at observation t, or `NA` if there are any missing values at t.

See also var.

sdc

Output:	row vector
Argument:	X (matrix)

Returns the standard deviations of the columns of X (with no degrees of freedom correction). See also meanc, sumc.

sdiff

Output:	same type as input
Argument:	y (series or list)

Computes seasonal differences: $y_t - y_{t-k}$, where k is the periodicity of the current dataset (see $pd). Starting values are set to `NA`.

When a list is returned, the individual variables are automatically named according to the template `sd_`*varname* where *varname* is the name of the original series. The name is truncated if necessary, and may be adjusted in case of non-uniqueness in the set of names thus constructed.

See also diff, ldiff.

selifc

Output:	matrix
Arguments:	A (matrix)
	b (row vector)

Selects from A only the columns for which the corresponding element of b is non-zero. b must be a row vector with the same number of columns as A.

See also selifr.

selifr

Output:	matrix
Arguments:	*A* (matrix)
	b (column vector)

Selects from *A* only the rows for which the corresponding element of *b* is non-zero. *b* must be a column vector with the same number of rows as *A*.

See also selifc, trimr.

seq

Output:	row vector
Arguments:	*a* (scalar)
	b (scalar)

Returns a row vector filled with consecutive integers, with *a* as first element and *b* last. If *a* is greater than *b* the sequence will be decreasing. If either argument is not integral its fractional part is discarded.

See also ones, zeros.

sin

Output:	same type as input
Argument:	*x* (scalar, series or matrix)

Sine. See also cos, tan, atan.

sort

Output:	same type as input
Argument:	*x* (series or vector)

Sorts *x* in ascending order, skipping observations with missing values when *x* is a series. See also dsort, values. For matrices specifically, see msortby.

sortby

Output:	series
Arguments:	*y1* (series)
	y2 (series)

Returns a series containing the elements of *y2* sorted by increasing value of the first argument, *y1*. See also sort, ranking.

sqrt

Output:	same type as input
Argument:	*x* (scalar, series or matrix)

Square root of *x*; produces `NA` for negative values.

sst

Output:	scalar
Argument:	*y* (series)

Sum of squared deviations from the mean for the non-missing observations in series *y*. See also var.

strlen

Output:	scalar
Argument:	*s* (string)

Returns the number of characters in *s*.

strstr

Output:	string
Arguments:	*s1* (string)
	s2 (string)

Searches *s1* for an occurrence of the string *s2*. If a match is found, returns a copy of the portion of *s1* that starts with *s2*, otherwise returns an empty string.

sum

Output:	scalar
Argument:	*y* (series)

Sum of the non-missing observations in series *y*.

sumc

Output:	row vector
Argument:	*X* (matrix)

Returns the sums of the columns of *X*. See also meanc, sumr.

sumr

Output:	column vector
Argument:	*X* (matrix)

Returns the sums of the rows of *X*. See also meanr, sumc.

svd

Output:	row vector
Arguments:	*X* (matrix)
	&U (reference to matrix, or `null`)
	&V (reference to matrix, or `null`)

Performs the singular values decomposition of the $r \times c$ matrix X:

$$X = U \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n, \end{bmatrix} V$$

where $n = \min(r, c)$. U is $r \times n$ and V is $n \times c$, with $U'U = I$ and $VV' = I$.

The singular values are returned in a row vector. The left and/or right singular vectors U and V may be obtained by supplying non-null values for arguments 2 and 3, respectively. For any matrix `A`, the code

```
s = svd(A, &U, &V)
B = (U .* s) * V
```

should yield `B` identical to `A` (apart from machine precision).

See also eigengen, eigensym, qrdecomp.

tan

Output:	same type as input
Argument:	x (scalar, series or matrix)

Tangent.

tr

Output:	scalar
Argument:	A (square matrix)

Returns the trace of the square matrix A, namely $\sum_i A_{ii}$. See also diag.

transp

Output:	matrix
Argument:	X (matrix)

Matrix transposition. Note: this is rarely used; in order to get the transpose of a matrix, in most cases you can just use the prime operator: `X'`.

trimr

Output:	matrix
Arguments:	X (matrix)
	ttop (scalar)
	tbot (scalar)

Returns a matrix that is a copy of X with *ttop* rows trimmed at the top and *tbot* rows trimmed at the bottom. The latter two arguments must be non-negative, and must sum to less than the total rows of X.

See also selifr.

uniform

Output:	series
Arguments:	a (scalar)
	b (scalar)

Generates a series of uniform pseudo-random variates in the interval (a, b), or, if no arguments are supplied, in the interval (0,1). The algorithm used is the Mersenne Twister by Matsumoto and Nishimura (1998).

See also randgen, normal, genpois, mnormal, muniform.

unvech

Output:	square matrix
Arguments:	v (vector)
	b (scalar)

Returns an $n \times n$ symmetric matrix obtained by rearranging the elements of v. The number of elements in v must be a triangular integer — i.e., a number k such that an integer n exists with the property $k = n(n+1)/2$. This is the inverse of the function vech.

See also mshape.

upper

Output:	square matrix
Argument:	A (square matrix)

Returns an $n \times n$ upper triangular matrix B for which $B_{ij} = A_{ij}$ if $i \leq j$ and 0 otherwise.

See also lower.

values

Output:	column vector
Argument:	x (series or vector)

Returns a vector containing the distinct elements of x sorted in ascending order. If you wish to truncate the values to integers before applying this function, use the expression `values(int(x))`.

See also dsort, sort.

var

Output:	scalar or series
Argument:	x (series or list)

If x is a series, returns the (scalar) sample variance, skipping any missing observations.

If x is a list, returns a series y such that y_t is the sample variance of the values of the variables in the list at observation t, or `NA` if there are any missing values at t.

In each case the sum of squared deviations from the mean is divided by $(n - 1)$ for $n > 1$. Otherwise the variance is given as zero if $n = 1$, or as `NA` if $n = 0$.

See also sd.

varname

Output:	string
Argument:	i (scalar)

Returns the name of the variable with ID number i, or generates an error if there is no such variable.

varnum

Output:	scalar
Argument:	*varname* (string)

Returns the ID number of the variable called *varname*, or NA is there is no such variable.

vec

Output:	column vector
Argument:	X (matrix)

Stacks the columns of X as a column vector. See also mshape, unvech, vech.

vech

Output:	column vector
Argument:	A (square matrix)

Returns in a column vector the elements of A on and above the diagonal. Typically, this function is used on symmetric matrices; in this case, it can be undone by the function unvech. See also vec.

wmean

Output:	series
Arguments:	Y (list)
	W (list)

Returns a series y such that y_t is the weighted mean of the values of the variables in list Y at observation t, the respective weights given by the values of the variables in list W at t. The weights can therefore be time-varying. The lists Y and W must be of the same length and the weights must be non-negative.

See also wsd, wvar.

wsd

Output:	series
Arguments:	Y (list)
	W (list)

Returns a series y such that y_t is the weighted sample standard deviation of the values of the variables in list Y at observation t, the respective weights given by the values of the variables in list W at t. The weights can therefore be time-varying. The lists Y and W must be of the same length and the weights must be non-negative.

See also wmean, wvar.

wvar

Output:	series
Arguments:	X (list)
	W (list)

Returns a series y such that y_t is the weighted sample variance of the values of the variables in list X at observation t, the respective weights given by the values of the variables in list W at t. The weights can therefore be time-varying. The lists Y and W must be of the same length and the weights must be non-negative.

The weighted sample variance is computed as

$$s_w^2 = \frac{n'}{n'-1} \frac{\sum_{i=1}^n w_i (x_i - \bar{x}_w)^2}{\sum_{i=1}^n w_i}$$

where n' is the number of non-zero weights and $\bar{x}_w$ is the weighted mean.

See also wmean, wsd.

xpx

Output:	list
Argument:	L (list)

Returns a list that references the squares and cross-products of the variables in list L. Squares are named on the pattern `sq_`*varname* and cross-products on the pattern *var1_var2*. The input variable names are truncated if need be, and the output names may be adjusted in case of duplication of names in the returned list.

zeromiss

Output:	same type as input
Argument:	x (scalar or series)

Converts zeros to `NA`s. If x is a series, the conversion is done element by element. See also missing, misszero, ok.

zeros

Output:	matrix
Arguments:	r (scalar)
	c (scalar)

Outputs a zero matrix with r rows and c columns. See also ones, seq.

Capítulo 3

Comments in scripts

When a script does anything non-obvious, it's a good idea to add comments explaining what's going on. This is particularly useful if you plan to share the script with others, but it's also useful as a reminder to yourself — when you revisit a script some months later and wonder what it was supposed to be doing.

The comment mechanism can also be helpful when you're developing a script. There may come a point where you want to execute a script, but bypass execution of some portion of it. Obviously you could delete the portion you wish to bypass, but rather than lose that section you can "comment it out" so that it is ignored by gretl.

Two sorts of comments are supported by gretl. The simpler one is this:

- If a hash mark, `#`, is encountered in a gretl script, everything from that point to the end of the current line is treated as a comment, and ignored.

If you wish to "comment out" several lines using this mode, you'll have to place a hash mark at the start of each line.

The second sort of comment is patterned after the C programming language:

- If the sequence `/*` is encountered in a script, all the following input is treated as a comment until the sequence `*/` is found.

Comments of this sort can extend over several lines. Using this mode it is easy to add lengthy explanatory text, or to get gretl to ignore substantial blocks of commands. As in C, comments of this type cannot be nested.

How does these two comment modes interact? You can think of gretl as starting at the top of a script and trying to decide at each point whether it should or should not be in "ignore mode". In doing so it follows these rules:

- If we're not in ignore mode, then `#` puts us into ignore mode till the end of the current line.
- If we're not in ignore mode, then `/*` puts us into ignore mode until `*/` is found.

This means that each sort of comment can be masked by the other.

- If `/*` follows `#` on a given line which does not already start in ignore mode, then there's nothing special about `/*`, it's just part of a `#`-style comment.
- If `#` occurs when we're already in ignore mode, it is just part of a comment.

A few examples follow.

```
/* multi-line comment
   # hello
   # hello */
```

In the above example the hash marks are not special; in particular the hash mark on the third line does not prevent the multi-line comment from terminating at `*/`.

```
# single-line comment /* hello
```

Assuming we were not in ignore mode before the line shown above, it is just a single-line comment: the `/*` is masked, and does not open a multi-line comment.

You can append a comment to a command:

```
ols 1 0 2 3 # estimate the baseline model
```

Example of "commenting out":

```
/*
# let's skip this for now
ols 1 0 2 3 4
omit 3 4
*/
```

Capítulo 4

Options, arguments and path-searching

4.1 Invoking gretl

`gretl` (under MS Windows, `gretlw32.exe`)[1].

— Opens the program and waits for user input.

`gretl` *datafile*

— Starts the program with the specified datafile in its workspace. The data file may be in any of several formats (see the *Gretl User's Guide*); the program will try to detect the format of the file and treat it appropriately. See also Section 4.4 below for path-searching behavior.

`gretl -help` (or `gretl -h`)

— Print a brief summary of usage and exit.

`gretl -version` (or `gretl -v`)

— Print version identification for the program and exit.

`gretl -english` (or `gretl -e`)

— Force use of English instead of translation.

`gretl -run` *scriptfile* (or `gretl -r` *scriptfile*)

— Start the program and open a window displaying the specified script file, ready to run. See Section 4.4 below for path-searching behavior.

`gretl -db` *database* (or `gretl -d` *database*)

— Start the program and open a window displaying the specified database. If the database files (the `.bin` file and its accompanying `.idx` file) are not in the default system database directory, you must specify the full path. See also the *Gretl User's Guide* for details on databases.

`gretl -dump` (or `gretl -c`)

— Dump the program's configuration information to a plain text file (the name of the file is printed on standard output). May be useful for trouble-shooting.

`gretlw32 -debug` (or `gretlw32 -g`)

— (MS Windows only) Open a console window to display any messages sent to the "standard output" or "standard error" streams. Such messages are not usually visible on Windows; this may be useful for trouble-shooting.

4.2 Preferences dialog

Various things in gretl are configurable under the "Tools, Preferences" menu. Separate menu items are devoted to the choice of the monospaced font to be used in gretl screen output, and, on some platforms, the font used for menus and other messages. The other options are organized under five tabs, as follows.

General: Here you can configure the base directory for gretl's shared files. In addition there are several check boxes. Checking "Tell me about gretl updates" makes gretl attempt to query the update server at start-up. If your native language setting is not English and the local decimal point character is not the period ("`.`"), unchecking "Use locale setting for decimal point" will make gretl use the period regardless. Checking "Allow shell commands" makes it possible to invoke shell commands in scripts and in the gretl console (this facility is disabled by default for security reasons).

[1] On Linux, a "wrapper" script named `gretl` is installed. This script checks whether the `DISPLAY` environment variable is set; if so, it launches the GUI program, `gretl_x11`, and if not it launches the command-line program, `gretlcli`

Databases tab: You can select the directory in which to start looking for native gretl databases; the directory in which to start looking for RATS 4 databases; the host name of the gretl database server to access; and the IP number and port number of the HTTP proxy server to use when contacting the database server (if you're behind a firewall).

Programs tab: You can specify the names or paths to various third-party programs that may called by gretl under certain conditions. Note that the item "Command to compile TEX files" can be set to either latex or pdflatex; if latex is selected, TEX output will be previewed in DVI format; if pdflatex is selected, the preview will be in PDF format.

HCCME tab: Set preferences regarding robust covariance matrix estimation. See the *Gretl User's Guide* for details.

Manuals tab: Select your preferred language for the full gretl documentation in PDF format (currently only English and Italian are supported). When using the English documentation you can also choose between US letter paper and A4 paper.

Settings chosen via the Preferences dialog are stored from one gretl session to the next. Under MS Windows they are stored in the Windows registry; on other platforms they are stored in a plain text file named `.gretlrc` in the user's home directory.

4.3 Invoking gretlcli

`gretlcli`

— Opens the program and waits for user input.

`gretlcli` *datafile*

— Starts the program with the specified datafile in its workspace. The data file may be in any format supported by gretl (see the *Gretl User's Guide* for details). The program will try to detect the format of the file and treat it appropriately. See also Section 4.4 for path-searching behavior.

`gretlcli -help` (or `gretlcli -h`)

— Prints a brief summary of usage.

`gretlcli -version` (or `gretlcli -v`)

— Prints version identification for the program.

`gretlcli -english` (or `gretlcli -e`)

— Force use of English instead of translation.

`gretlcli -run` *scriptfile* (or `gretlcli -r` *scriptfile*)

— Execute the commands in *scriptfile* then hand over input to the command line. See Section 4.4 for path-searching behavior.

`gretlcli -batch` *scriptfile* (or `gretlcli -b` *scriptfile*)

— Execute the commands in *scriptfile* then exit. When using this option you will probably want to redirect output to a file. See Section 4.4 for path-searching behavior.

When using the `-run` and `-batch` options, the script file in question must call for a data file to be opened. This can be done using the `open` command within the script.

4.4 Path searching

When the name of a data file or script file is supplied to gretl or gretlcli on the command line, the file is looked for as follows:

1. "As is". That is, in the current working directory or, if a full path is specified, at the specified location.

2. In the user's gretl directory (see Table 4.1 for the default values; note that `PERSONAL` is a placeholder that is expanded by Windows in a user- and language-specific way, typically involving "My Documents" on English-language systems).

3. In any immediate sub-directory of the user's gretl directory.

4. In the case of a data file, search continues with the main gretl data directory. In the case of a script file, the search proceeds to the system script directory. See Table 4.1 for the default settings. (`PREFIX` denotes the base directory chosen at the time gretl is installed.)
5. In the case of data files the search then proceeds to all immediate sub-directories of the main data directory.

Tabela 4.1: Default path settings

	Linux	*MS Windows*
User directory	`$HOME/gretl`	`PERSONAL\gretl`
System data directory	`PREFIX/share/gretl/data`	`PREFIX\gretl\data`
System script directory	`PREFIX/share/gretl/scripts`	`PREFIX\gretl\scripts`

Thus it is not necessary to specify the full path for a data or script file unless you wish to override the automatic searching mechanism. (This also applies within gretlcli, when you supply a filename as an argument to the `open` or `run` commands.)

When a command script contains an instruction to open a data file, the search order for the data file is as stated above, except that the directory containing the script is also searched, immediately after trying to find the data file "as is".

MS Windows

Under MS Windows configuration information for gretl and gretlcli is stored in the Windows registry. A suitable set of registry entries is created when gretl is first installed, and the settings can be changed under gretl's "Tools, Preferences" menu. In case anyone needs to make manual adjustments to this information, the entries can be found (using the standard Windows program regedit.exe) under `Software\gretl` in `HKEY_LOCAL_MACHINE` (the main gretl directory and the paths to various auxiliary programs) and `HKEY_CURRENT_USER` (all other configurable variables).

Capítulo 5

Reserved Words

Reserved words, which cannot be used as the names of variables, fall into the following categories:

- Names of constants: `CONST`, `NA`, `const`, `null`, `pi`.
- Names of internal variables and data types: `matrix`, `obs`, `scalar`, `series`, `t`.
- Names of functions, as shown in Table 5.1.

Tabela 5.1: Function names

BFGSmax	I	abs	acos	argname	asin	atan	bkfilt
cdemean	cdf	cdiv	ceil	cholesky	cmult	cnorm	colnames
cols	corr	cos	cov	critical	cum	det	diag
diff	dnorm	dsort	dummify	eigengen	eigensym	exp	fdjac
fft	ffti	filter	firstobs	floor	fracdiff	gammafun	genpois
getenv	gini	ginv	grab	hpfilt	imaxc	imaxr	iminc
iminr	infnorm	int	inv	invcdf	invpd	islist	isnull
isscalar	isseries	isstring	lags	lastobs	ldet	ldiff	lincomb
ljungbox	ln	lngamma	log	log10	log2	lower	lrvar
makemask	max	maxc	maxr	mcorr	mcov	mean	meanc
meanr	median	mexp	min	minc	minr	missing	misszero
mlag	mnormal	mols	movavg	mread	mshape	msortby	muniform
mwrite	mxtab	nelem	nobs	normal	nullspace	obslabel	obsnum
ok	onenorm	ones	orthdev	pdf	pmax	pmean	pmin
pnobs	polroots	princomp	psd	pvalue	qform	qnorm	qrdecomp
quantile	randgen	rank	ranking	rcond	readfile	resample	round
rows	sd	sdc	sdiff	selifc	selifr	seq	sin
sort	sortby	sqrt	sst	strlen	strstr	sum	sumc
sumr	svd	tan	tr	transp	trimr	uniform	unvech
upper	values	var	varname	varnum	vec	vech	wmean
wsd	wvar	xpx	zeromiss	zeros			