

A SIMPLE INTRODUCTION TO GAUSS

GAUSS is an econometrics/statistical package available in the Department of Economics in the 68 Heady Hall lab. The purpose of this document is to provide a brief introduction on how one can access GAUSS and the basic GAUSS commands. Additional information can be found in the GAUSS Help menu and the online users manual (http://www.aptech.com/AS_resLibMF.html) Chapters 10-12 are particularly helpful.

GAUSS COMMAND LANGUAGE BASICS

All GAUSS commands end with a semicolon (;). You can have more than one command on a given line as long as semicolons separate them. In COMMAND mode, a carriage return is the same as a semicolon. Single line comments are nested between two @ signs, as in:

@ here is a single comment @

Multiple line comments are bracketed by "/*" at the beginning and "*/" at the end, written in the form:

**/* here is a multiple
line comment */**

GAUSS is a very different from the statistical packages, such as TSP and SHAZAM. GAUSS has fewer "canned" routines, requiring more programming on the part of the user, but is significantly stronger in terms of its matrix and data manipulation capabilities, allowing one to develop efficient routines for procedures not covered by TSP and SHAZAM. Furthermore, GAUSS is significantly faster, an important feature when the procedure of interest involves a lot of matrix manipulations and/or do loops.

Here are some commonly required steps in a GAUSS program and the corresponding commands.

1. Specifying and manipulating a matrix

- a. A simple matrix can be generated using the let command:

```
let x = {1 2 3, 4 5 6, 7 8 9};
```

or simply

```
x = {1 2 3, 4 5 6, 7 8 9};
```

Either of these will create the 3×3 matrix

1	2	3
4	5	6

7

8

9

- b. Notice that spaces are used to separate elements of a row and commas are used to separate rows.
- c. A matrix can also be formed by concatenating other matrices or vectors. For example, if you enter:

```
a = {1.1 3.2};
b = {2.3 -4.5};
h=a~b;
k=a|b;
```

Then h is a 1×4 row vector equal to {1.1 3.2 2.3 -4.5} and k is a 2×2 matrix equal to

```
1.1      3.2
2.3      -4.5
```

- d. Portions of a matrix can be referred to as follows:
- The (i,j)th element of the matrix **x** is referred to as **x[i,j]**.
 - The ith row of the matrix **x** is referred to as **x[i,.]**.
 - The jth column of the matrix **x** is referred to as **x[.,j]**.
 - Rows i through k of matrix **x** are referred to as **x[i:k,.]**.
- e. *Matrix versus element by element operators*: GAUSS distinguishes matrix multiplication and division from element by element operations on matrices. Thus

```
x = y*z;
```

yields the inner product of the matrices **y** and **z**, where **y** has the same number of columns as **z** has rows. In contrast,

```
x = y .* z;
```

yields the element by element multiplication of **y** and **z**, where typically **y** and **z** will either have the same number of rows and columns or one will be a scalar.

Similarly,

$x = b/A;$

yields $x = A^{-1}b$, whereas

$x = b ./ A;$

does element by element division. Similar distinctions exist for the logical operators "<", ">", "==", ">=", and "<=". A dot prior to each operator indicates that element by element comparisons are to be made, whereas without this dot, entire matrices are compared.

2. Creating and retrieving a data set

- a. Data can be read into a matrix from a file using the **LOAD** command. In particular, if you have a file **mydata** containing an $n \times m$ matrix of data, you can load this matrix into GAUSS using the command:

LOAD x[n,m] = mydata;

- b. *GAUSS data sets*: It is often convenient, if you will be referring often to a data set, to create a GAUSS data set. Creating such a data set requires a series of steps, outlined below in the annotated example, in which we are trying to save the $n \times 3$ matrix as three variables in a GAUSS data set, with names "var1", "var2" and "var3".

- Step 1: Define the name of the data set:

outfile1 = "gdata";

- Step 2: Assign names to each column of the data matrix to be saved:

vnames = { "var1" "var2" "var3" };

- Step 3: Write data to GAUSS data set:

call saved(x,outfile1,vnames);

- c. GAUSS data can be retrieved by:

- Step 1: Defining the name of the GAUSS data file:

infile1 = "gdata";

- Step 2: Opening the GAUSS data file:

open file1 = ^infile1;

- Step 3: Reading in the data using the READR command:

```
z = readr(file1,r);
```

where r denotes the number of rows to be read.

- Step 4: Retrieve the names of the variables read in:

```
vnames = getname(infile1);
```

3. Obtaining basic statistics on variables: Basic statistics can be obtained using the following commands

- a. **MEANC(x)**: returns the mean of every column of a matrix.

Example:

```
x = {1 2 5, 2 0 6, 3 4 7};  
y = meanc(x);
```

will return a column vector:

```
y = {2, 2, 6};
```

- b. **MINC(x)**: returns a column vector containing the smallest element in each column of a matrix

Example:

```
x = {1 2 5, 2 0 6, 3 4 7};  
y = minc(x);
```

will return a column vector:

```
y = {1, 0, 5};
```

- c. **MAXC(x)**: returns a column vector containing the largest element in each column of a matrix

Example:

```
x = {1 2 5, 2 0 6, 3 4 7};  
y = maxc(x);
```

will return a column vector:

y = {3, 4, 7};

- d. **STDC(x)**: returns a column vector containing the standard deviation of each column of a matrix

Example:

x = {1 2 5, 2 0 6, 3 4 7};
y = stdc(x);

will return a column vector:

y = {1, 2, 1};

4. Plotting or graphing variables: There are a wide variety of graphing capabilities in GAUSS. The following sequence of commands illustrates the steps required to generate a simple XY plot of the variables **y1**, **y2**, and **y3** against **x**.

- a. Step 1: Activate the graphics library:

library pgraph;

- b. Step 2: Reset global graphics variables

graphset;

- c. Step 3: Combine the series of y-variables into a matrix:

y = y1~y2~y3;

- d. Step 4: Provide graphics title:

title("Plot of y1, y2, and y3 against x");

- e. Step 5: Set legend "on":

_plegctl = 1;

- f. Step 6: Call the main plotting routine:

xy(x,y);

More complex graphing capabilities and controls are detailed in Volume I, Chapter 13 of the GAUSS manuals.

5. Running a linear regression: Ordinary Least squares can be computed in a variety of fashions. Suppose we have an $n \times 1$ vector \mathbf{y} containing the dependent variables and an $n \times k$ matrix \mathbf{x} containing the independent variables. Then the ordinary least squares estimates of the parameter vector \mathbf{b} can be computed using:

a. Direct calculation:

$$\mathbf{b} = (\mathbf{x}'\mathbf{y})/(\mathbf{x}'\mathbf{x});$$

b. The **OLS** command:

output file = ols.out reset;
call ols(0,y,x);

The "0" in the first argument indicates that the data are provided directly in the program, as opposed to being read from a GAUSS file. The command **OLS** will, by default, include a constant (This can be altered by setting the global variable `__con = 0`). One can also retrieve results from the OLS regression using the format:

{vnam, m, b, stb, vc, stddev, sigma, cx, rsq, resid, dwstat} = ols(0,y,x);

instead of **call ols(0,y,x)**. The returned variables are:

vnam	$(k+2) \times 1$ vector of variable names
m	$(k+2) \times (k+2)$ moment matrix
b	$(k+1) \times 1$ least squares parameter estimates
stb	$(k+1) \times 1$ vector of t-statistics
vc	$(k+1) \times (k+1)$ variance covariance matrix estimates
stddev	$(k+1) \times 1$ vector of estimated parameter standard errors
sigma	Standard deviation of residual
cx	$k \times k$ correlation matrix of $\mathbf{y} \sim \mathbf{x}$.
rsq	R^2
resid	$n \times 1$ residual vector
dwstat	Durbin Watson statistic

See Volume II of the GAUSS manual for details regarding the format of these variables returned by the **OLS** command.

- c. OLSQR and OLSQR2 provide alternatives to OLS that are slower, but can better handle situations of near singularity.

GAUSS COMMANDS OF PARTICULAR USE IN ECON 673

Command	Purpose
CDFN(x)	Computes the cumulative distribution function (CDF) of the normal distribution
CDFNC(x)	1-CDFN(x)
CDFBNV(x,y,ρ)	Computes the cumulative distribution of the standard bivariate normal distribution
CDFNI(x)	Computes the inverse of a standard normal distribution
CDFTCI(x,d)	Computes the inverse of a standard t-distribution with degrees of freedom d
CEIL(x)	Rounds x up toward $+\infty$
COLS(x)	Computes the number of columns in x
INV(x)	Returns the inverse of an invertable matrix
ONES(r,c)	Constructs an $r \times c$ matrix of ones
PDFN(x)	Computes the standard normal probability density function
RNDN(r,c)	Creates an $r \times c$ matrix of standard normal pseudo-random numbers
RNDU(r,c)	Creates an $r \times c$ matrix of uniform pseudo-random numbers
RNSNS(r,c,s)	Creates an $r \times c$ matrix of standard normal pseudo-random numbers with seed of s
RNDUS(r,c)	Creates an $r \times c$ matrix of uniform pseudo-random numbers with seed of s
ROWS(x)	Computes the number of rows in x
SELIF(x,e)	Selects rows from a matrix x if e equals 1 for that row
ZEROS(r,c)	Constructs an $r \times c$ matrix of zeros