

# Introduction to GAUSS (I): From the very beginning

Enrique Moral-Benito\*

OCTOBER 2008

## 1 What is GAUSS?

Gauss is a mathematical programming language that for the purposes of econometric analysis allows you to enter data into matrices, and to manipulate matrices. This makes it possible to translate the formulas of the typical econometric textbook directly into a GAUSS program that will implement the formula on your data. It contains some complementary pre-programmed features such as a numerical optimization routine, that become handy in many applied econometric problems. It does in general, however, not contain pre-programmed estimation procedures (as does for example STATA).

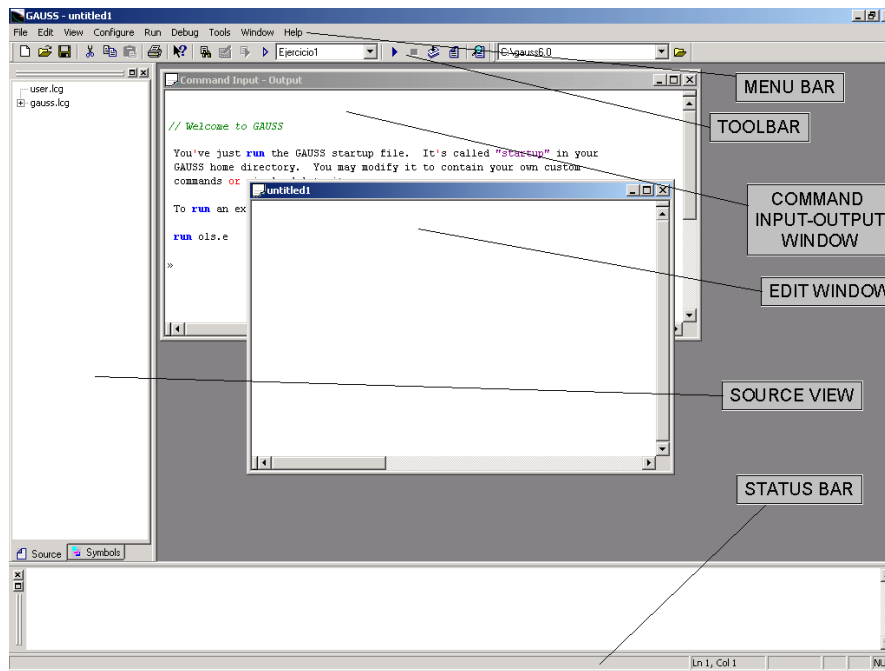
The emphasis in this brief course is on acquiring familiarity with the fundamentals of GAUSS and programming competence, rather than becoming a GAUSS guru.

---

\*E-mail: [enrique.moral@gmail.com](mailto:enrique.moral@gmail.com)

## 1.1 Workspace

In the following screenshot we can see the aspect of the computer screen once we have opened GAUSS:



### 1.1.1 Menu Bar

From the Menu Bar you have access to different menus such as File, Edit... which allow you to execute different actions (some of them can also be executed from the toolbar): edit a program, run a program, print, exit GAUSS... and what is going to be extremely useful for us, the Help Menu.

### 1.1.2 Toolbar

It includes different icons for quick access to the most frequently used commands such as new, open, cut, paste... and run. It also allows us to change the working directory.

### 1.1.3 Status Bar

It shows the status of the windows and processes on the left side. On the right side you will find the following:

- Cursor locator: number of the line and column in which the cursor is located.
- OVR: overwrite option activated.
- CAP: capital letters option activated.
- NUM: numbers option activated.

### 1.1.4 Command Input - Output Window

You can work in two different ways with GAUSS. The first one is to write the commands directly on the Command Input - Output Window and execute them by clicking the "enter" key in your keyboard. The results of the commands are shown in the same window.

### 1.1.5 Edit Window

The second (and most common) way of working with GAUSS is by running programs previously written in any text editor. GAUSS includes its own text editor. If you want to access the GAUSS text editor, there are two different ways:

1. Go to *File-New*
2. Click on the *New* button in the Toolbar

In the new window, the Edit Window, you can type the commands you want to execute. As before, the results will be shown in the Command Input - Output Window after running the program (set of commands written in the Edit Window).

### 1.1.6 Source View

There are two different tabs that allow us to access the files and symbols related to our working space:

- Source Tab: it includes a list of active libraries and their source files. You can access these files by clicking the right button of your mouse.
- Symbols Tab: it includes the symbols of the working space according to their type. (the most used is *matrices*)

## 2 Data Input

There are two basic ways of data input in GAUSS:

1. To create directly vectors and matrices of data in the program
2. To read the data stored in another file

## 2.1 Variable Definition

### 2.1.1 Scalars

To create a variable in GAUSS we only have to introduce the name of the variable and its value<sup>1</sup>. For example, the command:

```
a=3
```

creates a scalar variable with value equal to 3. Any posterior command can make use of this variable, for example:

```
b=sqrt(a)
```

generates a new variable called **b** whose value is the square root of the variable **a**.

### 2.1.2 Vectors and Matrices

We can also define vectors and matrices in a similar way than scalars. If we want to create the following matrix:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

we type either:

```
A = {1 2,3 4,5 6}
```

or:

```
let A[3,2] = 1 2 3 4 5 6
```

The commands `rows(A)` and `cols(A)` show the number of rows and columns of the matrix **A**.

Standard matrices can be defined as follows:

- Matrix **B** of ones with dimension  $N_1 \times N_2$ : `B = ones(N1,N2);`
- Matrix **C** of zeros with dimension  $N_1 \times N_2$ : `C = zeros(N1,N2);`
- Identity matrix **D** with dimension  $N_1 \times N_1$ : `D = eye(N1);`
- Matrix **F** of threes with dimension  $N_1 \times N_2$ : `let F[N1,N2] = 3;`

A vector **y** whose elements follow an arithmetic progression can be created as follows:

```
y = seqa(initial value,increment,N)
```

---

<sup>1</sup>It is important to note that GAUSS is not case-sensitive, i.e., it does not distinguish between upper-case and lower-case letters.

where  $N$  is the number of elements in the sequence (number of rows). For example:

$$y = \text{seqa}(1, 0.25, 4) \Rightarrow y = \begin{pmatrix} 1.00 \\ 1.25 \\ 1.50 \\ 1.75 \end{pmatrix}$$

## 2.2 Reading Data Files

There are several alternatives but here we will explain only the most basic way of loading data in GAUSS from an external file. We need to know the number of observations and variables there are in the data file and use the command:

```
load x[r,c]=filename
```

where  $x$  is the name of the matrix that will contain the data in GAUSS,  $r$  is the number of rows (observations) and  $c$  the number of columns (variables). For example: `load x[10,3]=c:\documents\data.csv`

If we do not know the number of observations but we do know the number of variables, we can load the data as follows:

```
load x[] = c:\documents\data.csv;
y = reshape(x, rows(x)/m, m);
```

where  $m$  is the number of variables.

## 3 Basic Operations with Matrices

### 3.1 Elements of a matrix

Given the matrix:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

- To access the element  $(i, j)$  of  $A$  we type `A[i, j]` or `z=A[i, j]` if we want to assign the value of that element to a new scalar variable called  $z$ .
- To access the row  $i$  of the  $A$  matrix  $\Rightarrow A[i, .]$ ; For example: `A[1, .] = ( 1 2 )`
- To access the column  $j$  of the  $A$  matrix  $\Rightarrow A[. , j]$ ; For example: `A[. , 1] = ( 1 )`  
`( 3 )`  
`( 5 )`

- We can also have access to different sub-matrices, for example:

$$A[2:3,1:1] = \begin{pmatrix} 3 \\ 5 \end{pmatrix}$$

## 3.2 Operators

Given the matrices:

$$A = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \text{ and } B = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$$

- Horizontal concatenation<sup>2</sup>:  $C = A \sim B \Rightarrow C = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$
- Vertical concatenation:  $D = A | B \Rightarrow D = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$
- Multiplication, addition and subtraction:  $C * A$ ,  $A + B$ ,  $A - B$  (we should always check that the dimension of the matrices is correct)
- Inverse: `inv(C)`
- Determinant: `det(C)`
- Transpose:  $C'$
- Eigenvalues: `eig(C)`
- Elements of the diagonal: `diag(C)`
- Element by element operation:  $F = A .* B \Rightarrow F = \begin{pmatrix} 3 \\ 8 \end{pmatrix}$  or  $G = A ./ B \Rightarrow G = \begin{pmatrix} 0.33 \\ 0.50 \end{pmatrix}$

## 4 Graphs

Any program drawing graphs should have the lines:

```
library pgraph;
graphset;
```

ideally at the start of the program. "library pgraph" enables GAUSS to know where all the specialized graph-drawing routines are to be found. Note that graphs cannot be drawn if this line is omitted. "graphset" resets the graphical global variables to the default state.

---

<sup>2</sup>The symbol  $\sim$  can be obtained by typing Alt+126

Two dimensions graphs are drawn with the command:

```
xy(x,y);
```

which draws  $x$  in the X-axis and  $y$  in the Y-axis. This statement must be written after having defined all the labels, titles and other options that can be modified for a particular graph. Next I will present some of these options:

- To add a title to the graph:

```
title("graph1");
```

- To label the axes:

```
xlabel("x label name");
```

```
ylabel("y label name");
```

- To define the scale of the axes:

```
xtics(min,max,step,minordiv);
```

```
ytics(min,max,step,minordiv);
```

For example, if we want to define a yearly time scale for the X-axis between the years 1970 and 2000, we will write:

```
xtics(1970,2000,1,1);
```

- To change the color of the graph lines we modify the value of the global variable `_pcolor`. The color is defined by an integer between 0 and 15. By default, GAUSS sets `_pcolor=0`.
- By default, GAUSS saves all the graphs in the same file, "graphic.tkf". Therefore, if we plot more than one graph that we want to save, we need to assign a name to every file in order to avoid that GAUSS overwrites the files. This is done by typing:

```
_ptek="filename.tkf";
```

You can obtain more information about graphs in GAUSS by typing "Using Publication Quality Graphics" in the *Search* tab of the *User Guide* in the Help Menu.

## EXERCISES LESSON 1: INTRODUCTION TO GAUSS (I)

Now we all have some basic notions of the GAUSS programming language. Therefore, we are ready to practise and to do some very easy exercises.

If you have some previous experience in any programming language and you find the exercises too easy, you can help your mates once you have finished the exercises.

### Exercise 1: Matrices

Generate the following matrices:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \quad B = \begin{pmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{pmatrix} \quad C = \begin{pmatrix} 2 & 8 & 4 \\ 9 & 3 & 6 \\ 1 & 5 & 2 \end{pmatrix}$$

and compute:

1.  $D = B * A$
2.  $E = D * A'$
3.  $F = B + C$
4.  $G = B. * C$
5. Try to compute  $A * B$ ,  $A. * C$  or  $A + E$ . What happens?
6. Calculate the determinant of  $C$  in two different ways and check that you obtain the same result. (*Hint: use the eigenvalues of  $C$* )
7. Compute the inverse of  $H = A * A'$ . What happen? Why?<sup>3</sup>

### Exercise 2: Random Number Generator

GAUSS has its own random number generator. This will be very useful for us when we do any kind of simulation. For example, we can generate a vector or a matrix of normal random numbers with the following command:

`rndn(r,c)`

where  $\mathbf{r}$  and  $\mathbf{c}$  are the numbers of rows and columns of the matrix of normal random numbers.

---

<sup>3</sup>Note that  $H$  can be written as  $A[:, 1] * A[:, 1]' + A[:, 2] * A[:, 2]'$ .

1. Create a scalar variable called **N** with the value 1,000,000.
2. Generate a  $N \times 1$  vector of normal random numbers and call it **z**.
3. Compute the mean and the standard deviation of **z** and call them **muz** ( $\mu_z$ ) and **stdz** ( $\sigma_z$ ).
4. Using **z**, generate a new vector **z1** with standard deviation 2 and mean 0
5. Generate a  $N \times 1$  vector of normal random numbers called **y** with mean to 3 and standard deviation 0.5. ( $\mu_y = 3, \sigma_y = 0.5$ )
6. Check that the mean and the standard deviation of **y** are 3 and 0.5 respectively.
7. Do the exercise again but using  $N = 1,000$ . What happens?

### Exercise 3: Graphics

Now we will plot the vectors of random numbers that we have generated in the previous exercise.

1. Load the **pgraph** library and reset all the graphical global variables to the default values.
2. Create the vector for the X-axes with the command **seqa**.
3. Plot the **z** vector of  $N(0, 1)$  random numbers and add the title: *Graph 1*
4. Save the graph in your computer with the name "**graph1.tkf**"
5. Plot the three series, **z**, **z1** and **y**, together (*Hint: horizontal concatenation*) in the same graph called *Graph 2*
6. Plot two different random draws with size 1,000 of a  $N(0, 1)$  variable in the same graph.
7. By using the Help menu plot the histogram of the three series