

GAUSS TUTORIAL

GAUSS is a powerful matrix-oriented computer programming environment. It is a programming language designed for matrix-based operations and manipulations, suitable for high level statistical and econometric computations. Here are some very important tips for GAUSS users:

- Each line of code must end with a semi-colon, ";"
- GAUSS environment is case insensitive. Typing variable names in uppercase, lowercase or a combination of both does not matter. That is, character "A" is the same as "a"
- A GAUSS program must start with the statement "new;"
- A GAUSS program must end with the statement "end;"
- Any comment used in programs should be bounded either by /*comment*/ or @comment@ characters
- The ORDER of commands sometimes makes a great difference in how GAUSS processes data

GAUSS Windows - GAUSS uses six types of windows: a [Command window](#), an [Edit window](#), an [Output window](#), a [Debug window](#), a [Matrix Editor window](#), and an [HTML-based Help window](#)

Command Window - You enter interactive commands in the Command window. This window can be selected from the Windows Menu or from the Status bar.

Edit Window - An Edit window is created when you open a file. When you execute GAUSS from an Edit window, the entire file is executed. This is the equivalent of the GAUSS "run filename" statement.

Output Window - Output is written to the Output window. GAUSS commands cannot be executed from this window.

1.1 Creating Programs

Gauss can be used interactively OR in the EDIT MODE. It is recommended that you use the EDIT MODE to create your program and run the entire program. If you wish to use the program interactively, then you just enter commands at the Gauss prompt. But if you wish to create a batch (program) file you must enter the following command in order to go into the EDIT mode:

EDIT DRIVE:FILENAME

where **FILENAME** refers to the name of the file into which the program is to be stored, and **DRIVE** refers to the drive containing the computer disk. Therefore, if the disk is in drive A and the file name is CP1.PRG, then write the command as:

EDIT A:CP1.PRG

If the file already exists it will be displayed ready to be edited. But if the file does not exist, then the new file can be created using a normal editing commands on the blank screen under **NEW**.

Generating/Imputing Data

All variables in Gauss are treated as matrices. This means that all operations must adhere to the rules of matrix algebra. Suppose we have three data points for three variables X, Y, and Z. Then we can define the

matrix W containing this data in three different ways:

- 1) `W={1 4 7, 3 5 9, 2 3 7};`
- 2) `let W[3,3]=1 4 7 3 5 9 2 3 7;`
- 3) `let W=1~4~7~3~5~9~2~3~7;`

The matrix W has dimensions (3 x 3). Suppose that all the data for variable X is located in the first row, all of Y's in row 2, and all of Z's in row 3. The variables X, Y, and Z can be defined in the following way:

`X = W[1,]; Y = W[2,]; Z = W[3,];`

This says that the values of the variables X, Y, and Z are in rows one, two, and three, respectively.

Example – One of the most frequently used operations in GAUSS is Matrix multiplication. Suppose we are asked to multiply the following two matrices:

<code>1 2 3</code>	<code>1 4 7</code>
<code>A = 4 5 6</code>	<code>B = 3 5 9</code>
<code>7 8 9</code>	<code>2 3 7</code>

`new; let A[3,3]=1 2 3 4 5 6 7 8 9; let B[3,3]=1 4 7 3 5 9 2 3 7; C=A*B; Print C; end;`

Save and run the program. The following will appear on the output table:

<code>13.000000</code>	<code>23.000000</code>	<code>46.000000</code>
<code>31.000000</code>	<code>59.000000</code>	<code>115.000000</code>
<code>49.000000</code>	<code>95.000000</code>	<code>184.000000</code>

To remove the zeros add: `format /ld 7,0;` before the `end;` command

Reading Data

Data can be read into the Gauss program from either within the Gauss Program file itself or by using an external ASCII file containing the data. If the data for matrix W was present in an ASCII text file (called A:MYFILE.DAT) in the following form:

```
1 4 7
3 5 9
2 3 7
```

Then, the information could be read into Gauss using the LOAD command:

`LOAD W[3,3] = A:MYFILE.DAT;`

Data Transformation

To repeat, all variables defined in Gauss are treated as matrices. Even a variable containing a single value (a scalar) is defined as a (1 x 1) matrix. This means that operations can only be performed if matrices are conformable, and if they satisfy the conditions set by the theory of matrices. For example, two matrices can be summed only if they have the same dimensions. And, obviously, only nonsingular matrices can be inverted. Some most frequently used operations:

A = X + Y;	A is the sum of X and Y
A = X+Y[2,];	A is the sum of X and row 2 of Y
A = X - Y;	A is the difference of X and Y
A = X[.,1]-Y;	A is the difference of the first of X and Y
A = X * Y;	A is the product of X and Y
A = X'Y;	A is the product of X transpose and Y
A = X[1,]*Y;	A is the product of the first row of X and Y.
A = B';	A is the transpose of B.
Y = X!;	This computes the factorial of every element in the matrix X.

Procedures

Sometimes we are given large computing tasks for which it is difficult to write programs as a whole. We need to use procedures to separate large tasks into collection of smaller tasks by using specific codes repeatedly. Any procedure should contain the following parts:

1. **Procedure declaration** – done by GAUSS command **proc**
2. **Local variable declaration** – done by GAUSS command **local**
Note: WE should distinguish local variables from global variables. Variables that do not affect other parts of the program are considered to be local
3. **Body procedure** – describe smaller tasks that will be repeated, we call this **looping**
4. **Return from procedure** - by GAUSS command **retp** we want the smaller task to repeat as many times as needed to complete the main task
5. **End of procedure** – we require the procedure to end by GAUSS command **endp**
Note: GAUSS procedures are recursive, so that it can repeat as long as there is a logic to prevent from infinite recursion

Example – Suppose we are asked to calculate the n^{th} power of any square matrix. We need to use a procedure since taking n^{th} power of matrix is a multiple task to be performed. The following is an example of the syntax used to define this procedure:

```

proc(1)=pow(A,n);          /*declares the procedure-we can give any name to procedure*/
    local xn;              /*the local variable is the matrix A*/
    xn=A;                  /*start looping-start taking power of matrix A for n times*/
                                /*note: start from 2 b/c we have already assign A to local variable xn*/
for ctr(2,n,1);
A=xn*A;
endfor;                  /*the loop end here*/
retp(A);                  /*return to the procedure*/
endp;

```

Graphics

Gauss is capable of making complex graphs and can create a variety of Graph types. Each of them has a distinct pre-defined procedure name and hence must be *called up* as one. Following is the list of graph types that Gauss can create and the procedure names associated with each:

BAR	Generate bar graph
BOX	Graph data using box graph percentile method
CONTOUR	Graph contour data
DRAW	Supply additional graphic elements to graphs
HIST	Compute and graph frequency histogram
HISTF	Graph histogram given vector of frequencies
HISTP	Graph percent frequency histogram of a vector
LOGLOG	Scatter plot with logarithmic X and Y axes
LOGX	Scatter plot with logarithmic X axis
LOGY	Scatter plot with logarithmic Y axis
POLAR	Polar plots
SURFACE	Graph 3-D surface
XY	Scatter plot
XYZ	Graph 3-D scatter plot

Before we can create a Graph in Gauss, we need to invoke the Graphics Library by using the command **library pgraph**; Also all data (or matrices/vectors) that are used in the graph generation must be defined before using them. Once these pre-requisites are met, the graphing procedure can be *called up* with the required syntax.

Example – Plot the graph of the following function: $y=x^2$ (x to the power of 2)

```
library pgraph;
fn parabola(x)=x^2;          /*let domain be (-2,2) and let applied range be (-4,0)*/
d=seqa(.2,.01,401);        /*divide entire range into small units, .01 is the size of increment*/
xy(d, parabola(d));        /*U-shaped parabola with minima at the origin (0,0)*/
```

Statistical Analysis Using Gauss Procedures

Below is an example of a multivariate regression where the dependent variable is a 100 by 1 uniform random variable, and the independent variables are from a 100 by 5 matrix of normal random numbers:

```
new;
x = rndn(100,5);
y = rndu(100,1);
call ols("",y,x);
end;
```

We can run this either interactively by simply typing these commands in Gauss, or non-interactively by entering these commands in a command file and then executing them. Below is the descriptive statistics that will appear on the output window:

Valid cases:	100	Dependent variable:	Y
Missing cases:	0	Deletion method:	None
Total SS:	8.460	Degrees of freedom:	94
R-squared:	0.032	Rbar-squared:	-0.019
Residual SS:	8.187	Std error of est:	0.295
F(5,94):	0.627	Probability of F:	0.679

Variable	Estimate	Standard Error	t-value	Prob > t	Standardized Estimate	Cor with Dep Var
CONSTANT	0.481277	0.030025	16.029103	0.000	---	---
X1	0.002540	0.029176	0.087042	0.931	0.008989	0.005182

X2	-0.026961	0.031554	-0.854458		0.395	-0.088646	-0.072611
X3	0.044416	0.033791	1.314448	0.192	0.138098	0.144467	
X4	0.021887	0.029694	0.737095	0.463	0.077260	0.079122	
X5	-0.002981	0.031551	-0.094486		0.925	-0.009913	0.025793

Saving and Printing Results

It is good programming practice, and often necessary, to save the output of a program onto a file. In order to accomplish this in GAUSS, use the command : **output file=capture.out [reset][on]**; All the output that appears on the screen after this line is written in the text file **capture.out**. The command **reset** erases all previous content in the file. The command **on** appends to the file. To close the file, use **output off**

Note: files must be closed before they can be printed or edited. A closed file can be re-opened with **output on**. The file **capture.out** is in ASCII format and can be viewed and printed with any text editor.

How To Use Help On Gauss

Getting general help for Gauss is obtained by pressing ALT-H, then typing ? will give you the index of help, you can keep pressing pagedown keypad till you see the information you need. Typing ?? will give you list of all the operators in Gauss, type **h** and it will ask for the name of operator, put in the name of the operator and it will show the detail information on the operator. The help system also includes context sensitive help while in all of the pop-up menus and input screen.

◆ THE END ◆