Gauss Introduction

Kerstin Kehrle

October 21, 2008

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

preliminary remarks

advantages of GAUSS

- GAUSS = software package for statistical and econometric purposes; quasi-standard amongst econometricians; many implemented procedures

- its programming language is easily learned and is very similar to MATLAB

- central data element is the matrix, i.e. formulae can be entered intuitively

disadvantages

- no public domain software; expensive

- relatively few books/ scarce documentation $\rightarrow$
  Use the very good User's Guide and Reference help files!!!

user interface

- input can either be entered directly in the `command input-output`
  `window` $\rightarrow$ command will be executed directly or in the `editor window`
  $\rightarrow$ complete code file will be executed by clicking on the `run` button

- the `error output-window` will become handy...

Writing a program

- programs can be written in any editor as ACII-files; use of GAUSS editor recommended, though

- program files can have any file extension, sensible choice: `.prg`

- you can define a working directory on the tool-bar. GAUSS will look for your files there

GAUSS basics

- GAUSS is *not* case-sensitive!

- some variable names are forbidden as they refer to internal procedures, e.g. abs, and, cls

- **Use** comments!!!   @ comment @ or /* comment */

- cls clears the input-output window

- new clears the memory

Generating matrices and matrix manipulation

- variables are defined 'on the go', i.e. a=5

  $\rightarrow$ scalars and variables are just special matrices

- matrices are defined row by row, i.e. `A={1 2, 3 4}` generates

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} ; B = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$$

B is generated accordingly

- individual elements e_{ij} of a matrix can be addressed by
  `newvarname=varname[i,j]` where i and j are the row and column
  position
  example: 3 is the e_{21} element of $\mathbf{A}$ and can be extracted by
  `newvarname=varname[2,1]`

- whole columns can be addressed by `varname[.,1]`

- **WARNING:** GAUSS overwrites preexisting variables without a prior
  warning or error message!!!

- sometimes, it is useful to generate an empty matrix by `u={}`

- horizontal concatenation of two matrices: a~b; vertical concatenation by a|b

  example: c=a~b returns

$$C = \begin{pmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \end{pmatrix}$$

d=a|b returns

$$D = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{pmatrix}$$

- `cols(varname)` and `rows(varname)` returns the number of columns and rows of a matrix

  example: `cols(`C`)` returns 4; `rows(`C`)` returns 2

- Hint: Always make sure that you know the dimension of the output of GAUSS procedures. Is the result of a row or a column vector?

Operators, matrix algebra

operators for scalars

- all basic mathematical operators like +,-,*,/ can be used

- variables can be raised to power `n` by ^n, faculty uses !

- basic calculus rules known from algebra apply

relational operators

== (eq); /= (ne); > (gt);
< (lt); >= (ge); <= (le);

- The result of a logical operation of the form `t=a op b` (where `op` is one of the aforementioned operators) is a logical variable → false=0 and true=1

logical operators

`and`, `or` and `not` can also be used

Matrix algebra

- standard matrix algebra calculus rules apply

- $\mathbf{A}'$ yields the transpose of $\mathbf{A}$

$$\mathbf{A}' = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$$

- element-wise operations: adding and subtracting matrices by +,-, multiplying and dividing a matrix by a scalar *,/

- Remember: Matrix multiplication and inversion of matrices is not an

element-wise operation!

`inv(matname)` and `*` follow laws of matrix calculus

- elementwise operations are possible! `a.*b` yields

$$\left(\begin{array}{cc} a_{11} \cdot b_{11} & a_{12} \cdot b_{12} \\ a_{21} \cdot b_{21} & a_{22} \cdot b_{22} \end{array} \right) = \left(\begin{array}{cc} 1 \cdot 5 & 2 \cdot 6 \\ 3 \cdot 7 & 4 \cdot 8 \end{array} \right)$$

- further element-wise operators are `./` and `.^`

- Matrices can also be multiplied element-wise by a vector

- All *logical* and *relational* operators can also be applied element-wise: `.op`

overview **GAUSS** operations

	mathematical operation	implementation in GAUSS
matrix multiplication:	C=A $\cdot$ B $\rightarrow$ inner product	`>> C=A*B`
element-wise multiplication:	does not exist here!	`>> C=A.*B` equals $c_{ij} = a_{ij} \cdot b_{ij}$
raising to power:	$A^2 = A \cdot A$	`>> AQ=A^2`
element-wise raising to power	does not exist here!	`>> AQ=A.^2` equals $c_{ij} = a_{ij}^2$
element-wise division:	does not exist here!	`>> AD=A./B` equals $c_{ij} = a_{ij}/b_{ij}$

Matrix addition/subtraction are element-wise operations: `C=A+B` equals $c_{ij} = a_{ij} + b_{ij}$

further matrix commands

further commands for generating special matrices

- `seqa`(start,incr,times)

- `ones`(rows,cols), `zeros`(rows,cols), `eye`(dim)

further matrix functions

- `sumc`(matname), `cumsumc`(matname)

- `selif`(matname,e), `delif`(matname,e)

- `counts(matname,v)`

- `diag(matname)`

- `rank(matname)`, `det(matname)`, `eig(matname)`

Procedures

- efficient way of programming

- procedures are used to repeat program code with different arguments

- the difference between local and global variables is important. Local variables are only used within the procedure. Global variables are used everywhere in the program.

Syntax of a Procedure

`proc`(number of outputarguments) = procname(inputarg1,inputarg2,...);

`local` locavar1,locavar2,locavar3...;

< `proc` body operations >;

`retp`(outputarg1,outputarg2,...);

`endp`;

call the procedure:

`call` procname(inputarguments); /*ignores outputarguments */

outputarg1 = procname(inputarguments);

{outputarg1,outputarg2,...} = procname(inputarguments);

Exercise: Write a procedure that multiplies two matrices X and Y and that checks the conformity.

`if-else-elseif-endif`...**conditional branching**

`if` condition1;

$<$ do this 1 $>$;

`elseif` condition2;

$<$ do this 2 $>$;

...

`else`;

$<$ do this 3 $>$;

`endif`;

The `else` and `elseif` statements are optional.

Exercise: There are three states (1,2,3) a scalar can take. Either the scalar is negative, it is within 0 and 1 or it is greater than one. Test your program code with different scalar values.

`while`, **and** `until`, `for`...**loop statements**

Syntax of a loop statement: `for`

for i (start, stop, step);
< do this >;
endfor;

unconditional loop statement

used if the number of repetitions is certain and does not depend on a condition

example: fixed number of simulations

Exercise: create a zero matrix of dimension 10 times 4 and write in each row the actual time using a `for` statement. Hint: `time` delivers the actual time.

`while`, **and** `until`, `for`...**loop statements**

Syntax of a loop statement: `while` **and** `until`

`do while condition;`	`do until condition;`
$<$ do this $>$;	$<$ do this $>$;
`endo;`	`endo;`

conditional loop statement

`do while` is executed as long as the condition is true

(loop action is skipped as soon the condition is wrong)

`do until` is executed as long as the condition is wrong

(loop action is skipped as soon the condition is true)

syntax for `do until` and `do while` is identical

Exercise: As previous exercise, but use a `do while` and `do until` statement instead.

`while`, **and** `do until`, `for`...**loop statements**

Hint: Use vector operations instead of loops if possible. Exercise: create a data vector of natural numbers from 1 to 100000. Compute the sum of this rows (a) with a for loop and (b) with a conditional loop and (c) with an inner product. Measure the time difference between the three variants. Hint: Use the function `hsec`.

save and load... GAUSS data format

- there are many different ways to load and save data in GAUSS

- supported formats are e.g.: .asc .txt .xls .fmt .dat ...

- load syntax for .txt or .asc: load varname[r,c] = filename.txt or load varname[] = filename.txt

- load syntax for .fmt: load varname = filename.fmt

- save syntax: save filename = varname (GAUSS saves in a fmt format)

Exercise: Save and load a self-designed matrix as .fmt

Statistical Functions in GAUSS

- `meanc(varname)` and `stdc(varname)`

- `median(varname)` and `quantile(varname,probs)`

- `corrx(varname)`

- `ols()`

Exercise: use a data matrix and try all statistical functions

Graphs in GAUSS

- complex and extensive graphics features possible

- it is recommended to use a more sophisticated graphical software

- "quick and dirty" graphs

- activation of graphics tool by calling the library by the command: `library pgraph; graphset;` The second command resets all global variables in the library to the default values.

Graphics commands in GAUSS

- `xy(x,y)` twodimensional coordinate axes with x and y on x- and y-axis

- `bar(varname,ht)` bar chart

- `hist(varname,bp)` histogram

- `xyz(x,y,z)` three dimensional graphic

More graphics details

- all global variables in the `pgraph` look like `_p`...

- consider `xy(x,y)`: `_plctrl` controls the lines between the points x and y if `_plctrl=0` (default) the points are connected by lines; if `_plctrl=` -1 the points are not connected.

- `_plwidth` controls the line width

- note: the new settings are valid until they are either changed or reset by `graphset`.

- with the command `window(r,c,t)`, it is possible to combine graphics to one graph r and c are the number of rows and columns, respectively and t the character of the graphic (transparent or not)

- note: There are some graphics examples in `c:\gauss6.0\examples`

- `title("` graph name `"); xlabel("` label name `")`

Probability Distributions

- there are many commands for the computation of distribution functions, inverse distribution function, complement of a distribution function (1-F(x)) and probability functions

- example: standard normal probability function `pdfn(x)` with associated distribution function `cdfn(x)`, complement `cdfnc(x)`

Exercise: Plot the density function of a N(0,1) random number in a graph.

Random Numbers

- there are GAUSS commands for the computation of random numbers

- example: standard normal-, uniform-, gamma-, poission-, negative binomial- random numbers,...

- syntax example for drawing a uniform random number: u = rndu(r,c)
  r and c are the number of rows and columns

Exercise: Random Numbers and Graphs

Exercise: Write a procedure that produces a $(N \times 1)$ vector of a t_ν-distributed random variable.

Create a t_ν-distributed random variable Y (sample size N=10000) from a normally distributed r.v. X and a χ^2_ν-distributed r.v. $U = \sum_{i=1}^{\nu} Z_i^2$ where Z is normally distributed. X and Z have to be independent. For example, Y could be computed as:

$$Y = \frac{X}{\sqrt{U/\nu}}$$

Plot the empirical probability function of the produced t_ν-distributed variable in a graph.

Exercise: OLS

- Import to Gauss the file `wages1_data.txt` (download from homepage) which contains 3296 observations taken from a US Household Panel. Overview of the imported data: column structure from first to fourth column `exper`: experience in years `male`: 1 if male, 0 otherwise `educ`: years of schooling `wage`: wage (in $) per hour

- Reproduce the descriptive statistics in the table below.

Variable	Obs	Mean	Std. Dev.	Min	Max
exper	3296	8.041869	2.290855	1	18
male	3296	.5239684	.499501	0	1
educ	3296	11.63016	1.657114	3	16
wage	3296	5.816391	4.054694	.0765556	112.7919

- Plot the histogram of `wage` and save the graph.

- Estimate the following regression models by OLS and interpret β_2 (the returns of schooling).

a) $\texttt{wage}_i = \beta_1 + \beta_2\texttt{educ}_i + \beta_3\texttt{exper}_i + \varepsilon_i$

b) $\texttt{lnwage}_i = \beta_1 + \beta_2\texttt{educ}_i + \beta_3\texttt{exper}_i + \varepsilon_i$

Hints: write a OLS procedure that includes the following:

1) Define y and x.

2) compute the ols estimator: $\mathbf{b} = (\mathbf{X'X})^{-1}\mathbf{X'y}$

3) Compute the Variance-Covariance matrix of the OLS estimator: $Var(b) = s^2(X'X)^{-1}$ with $s^2 = \sum e_i^2/(n-k)$

4) Compute the t-test with $H_0 : \bar{\beta}_k = 0$ $t_k = \frac{\beta_k - \bar{\beta}_k}{se(b_k)}$

5) Print the output of coefficient, standard error and t-test in the Output window.

- compare your results with the OLS Gauss procedure results