

Some preliminaries:

- The relevant objects in GAUSS are matrices -- a term encompassing both vectors and scalars.
- Matrices can be either string or numeric. We will try to deal mostly with numeric.
- GAUSS can be operated at the command line (like STATA) or from command files.
- GaUSS Is CaSe iNsEnSiTiVe.
- GAUSS for Windows has some residual UNIX-ish characteristics. The most important for our purposes is the use of the forward slash (/) to denote file paths, e.g.  
"c:/gauss36/mydata.dat"

## I. Matrices

### A. Declarations and assignments

Let  $x = \{1\ 2\ 3,\ 4\ 5\ 6,\ 7\ 8\ 9\}$  produces a 3x3 matrix

Let  $x = \{1,2,3,4,5,6,7,8,9\}$  produces a 9x1 column vector

Let  $x = \{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\}$  produces a 1x9 row vector

$Y=x$  produces a new matrix,  $y$ , equal to  $x$

If you have initialized a matrix (i.e. declared it), you can always have GAUSS output the matrix by typing its name.

### B. Concatenation and transposition

Vertical concatenation: | (bar):  $y=x1|x2$ ;  $y=1|2$

Horizontal concatenation: ~ (tilde):  $y=x1~x2$ ;  $y=(1|2)~(3|4)$

Transpose operator: ' (apostrophe):  $y=x'$

### C. Special matrices

Zeros(rows, columns)

Ones(rows, columns)

#### Matrices of constants

Eye(rows) -- Why only one argument?

Seqa(start, interval, length)

Seqm(start, factor, length)

### D. Some random number matrices

Rndu(rows, columns)

Rndn(rows, columns)

### E. Describing matrices

Rows(matrix), Cols(matrix)

Sumc(matrix) -- How would you obtain the row sum?

Meanc(matrix), Median(matrix), Stdc(matrix)

Minc(matrix), Maxc(matrix)

Corrx(matrix)

### F. Indexing and submatrix extraction

- Matrices are indexed with square brackets.  $X[3, 4]$ , for example, returns the third row, fourth column element of  $x$ .
- Submatrix arrays are indexed with a colon, e.g.  $x[25:50, 2:4]$

- To extract all rows or all columns, use a period, e.g. `x[:, 3:4]` (returns a matrix consisting of the third and fourth columns of `x`).
- Vectors (column or row) require only one index, e.g. `x[4]` returns the fourth element of the vector `x`.
- To sort on the  $c^{\text{th}}$  column, type `sortc(x,c)`;

#### G. Operators

1. Scalar Operators: what you're used to
2. Matrix operators: Matrix multiplication
3. Matrix operators: Element-by-element, addition and subtraction
4. Matrix operators: Element-by-element, multiplication and division
5. Relational operators: "dot" and non-dot versions
6. Logical operators: dot and non-dot versions
7. Other important matrix manipulations:  
`Diag`, `inv` and `invpd`, `det`, `chol`

#### H. File I/O

##### Output files

- `Output file = output.out`

##### ASCII file input:

- `load x[]=data.asc` produces an  $N \times 1$  vector. This can be reshaped into an  $N_2 \times K$  vector with `x=reshape(x,N2,K)`
- Alternatively, type `load x[N,K] = data.asc` to cut out the second step.

##### GAUSS datasets

##### Basic file input:

1. Create a string variable representing the file name, e.g.  
`file1 = "mydata"`
2. Specify a file handle:  
`open f1 = ^file1` OR `open f1 = mydata.dat`
3. Get variable names: `mynames = getname("mydata")`.  
Type `$mynames` to view the names
4. If you don't know the number of observations, find out: `obs = rowsf(f1)`;
5. Create a matrix using the data file:  
`X = readr(f1,obs)`
6. Important: If you only read some of the observations in, you must reset GAUSS's "pointer" for that particular file. Otherwise, if you read ten lines of data to see what it looked like, you will start reading on the eleventh line the next time you use `readr`. Do so by typing `seekr(f1,1)`.

##### Basic file output

Suppose you want to save a  $100 \times 3$  matrix 'x' with variable names `x1`, `x2`, and `x3`:

1. Create a string variable with the dataset name, e.g.  
`Dataset = "c:/myfolder/mydata"`
2. Identity the variable names:

- Let vnames = x1 x2 x3
- 3. Create the dataset
- Saved(x, dataset, vnames)

## II. Repetition of tedious tasks: Do-Loops

### A. Find “machine zero”

```
i=1;
do until i+1 == 1;
    epsilon = i;
    i=i/2;
endo;
```

Note that we would likely write such a program in a command file. In this mode, each line must end with semi-colons.

### B. Draw a 100x1 standard normal vector, and save the median. Repeat 1000 times.

```
result = zeros(1000,1);
i=1;
do until i > 1000; /* OR do while i<=1000; */
    draw = rndn(100,1);
    meddraw = median(draw);
    result[i] = meddraw;
    i=i+1;
endo;
```

## III. Conditional branching: If, then, else, elseif, endif;

Sometimes, we want GAUSS to determine whether a statement is true or false, and do something depending on the truth of the statement. Here is a stupid example:

```
x=rndu(100,1);
meanx=meanc(x);
if meanx<0.33;
    print "Unusually low";
elseif 0<meanx and meanx<0.66;
    print "Just right";
else;
    print "Unusually high";
endif;
```

## IV. Vectorize whenever possible.

Do-loops are well and good, but they are slow. Whenever possible, conduct operations with matrices and vectors rather than looping with scalars.

## V. Procedures: Programs within programs

- A. Elements of a procedure: proc, local, retp, endp.
- B. Examples:
  - 1. Procedures with one return
    - a. One argument

```
proc sumr(x); /* Returns a row sum */
```

```

    local xtransp, res;
    xtransp=x';
    res = sumc(xtransp);
    retp(res);
endp;

```

### b. More than one argument

```

proc sampmo(x,y); /* Calculates the raw sample moment matrix */
    local n,mo,sm;
    n=rows(x);
    mo = (x'x);
    sm = (x'x)/n;
    retp(sm);
endp;

```

## 2. A procedure with multiple returns

```

proc (2) = rootpow(x,y); /* Calculates x^y and y^x */
    local r1, r2;
    r1 = x^y;
    r2 = y^x;
    retp(r1,r2);
endp;

```

### C. Defining matrices in terms of procedures:

```

a={1,2,3,4};
b={4,3,2,1};
{rp1,rp2}=rootpow(a,b);

```

## VI. Programming Etiquette:

In Gauss, remarks are delineated in one of two ways:

Multi-line comments begin with `/*` and end with `*/`

Single-line comments can begin and end with `@`

1. Your file should have a header. Here is an example of a header for a file I wrote for this session:

```

/*****
* File:          Tutor2.prg
*                for Gauss (3.5)
* Date:          January 8, 2002
* Author:        Sanford Gordon
* Purpose:       This file is intended as a tutorial for
*                procedures, loops, and branching operations
*                in GAUSS.
* Data used:     All internal
* Output file:   None
* Data output:   None
*****/

```

Note the use of the `/*` and `*/`; The other asterisks are aesthetic (so my eye picks up that this is a header right away).

2. All programs should be comment-rich. Every line of code does not need a comment, but every unique task that is not transparently obvious should have a comment. Some tasks take multiple lines, others only one.

3. Use tabs and spacing. Consider the following useless bit of code, which generates 1000 standard normal variates, and saves the maximum, one hundred times:

```
i = 1; do until i > 100; /* Starts a do loop */
    x = rndn(1000,1); /* 1000x1 standard normal draw */
    maxx = maxc(x); /* Find the maximum in x */
    if i == 1; /* conditional statement */
        results = maxx;
    else;
        results = results|maxx;
    endif;
    i = i+1;
endo;
```