MACRO DATABASE, PROC JOIN, AND FRIENDS - SEVERAL NEW SAS MACROS, PROCEDURES, AND FUNCTIONS
Arnold W. Bragg, North Carolina State University

ABSTRACT

MACRO DATABASE is a SAS program for summarizing entire SAS databases. Unlike PROC CONTENTS which summarizes by dataset, DATABASE combines variable, device, OS attributes, and SAS attributes from each SAS dataset in the database into a single report. PROC JOIN is a SAS procedure which 'joins' two datasets in the relational sense. PROC JOIN is similar to MERGE except that one observation is produced for each possible combination of the BY variables. Eight SAS date functions and nine SAS functions which generate pseudo-random variates are also discussed, with examples.

MACRO DATABASE

SAS MACRO DATABASE is a SAS program for summarizing entire SAS databases. DATABASE gives SAS users a rudimentary data dictionary/directory facility. Unlike PROC CONTENTS which summarizes by dataset, MACRO DATABASE combines attributes from all specified SAS datasets or SAS databases into a single output dataset from which reports can be generated. MACRO DATABASE summarizes the following attributes:

variable attributes: variable name type length variable # position format informat label
device attributes: unit device type volser
OS attributes: OS dataset name
SAS attributes: SAS dataset name date created # observations (disk)

MACRO DATABASE works for disk and tape SAS datasets and single or multiple dataset databases. Because MACRO DATABASE 'reads' the output from PROC CONTENTS invocations, changes in the physical attributes or layout of SAS datasets are essentially user transparent. The dataset built by DATABASE can be sorted by any combination of the fifteen attributes listed above. For example, suppose our SAS database consists of a SAS disk database containing two SAS datasets, a SAS tape database on a standard-labeled volume, and a SAS tape dataset on a non-labeled volume:

SAS database (disk) -- 2 SAS datasets
SAS dataset (tape) -- 1 SAS dataset
SAS dataset (tape) -- 1 SAS dataset

Figure 1. illustrates the report generated as a result of executing MACRO DATABASE with program statements and datasets as indicated in the example above.
<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>TYPE &amp; VARI-</th>
<th>POSI-</th>
<th>UNIT</th>
<th>DEVICE</th>
<th>SAS DSN/ CRATED # OBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS_NO</td>
<td>NUM 8 133 1133</td>
<td>1.</td>
<td>ACCESSION NUMBER</td>
<td>DISK</td>
<td>YEAR79 7921</td>
</tr>
<tr>
<td>ACCESS_NO</td>
<td>NUM 8 133 1133</td>
<td>1.</td>
<td>ACCESSION NUMBER</td>
<td>DISK</td>
<td>YEAR80 3609</td>
</tr>
<tr>
<td>ACCESS_NO</td>
<td>NUM 8 133 1133</td>
<td>1.</td>
<td>MISCGAPS</td>
<td>TAPE</td>
<td>GAP77</td>
</tr>
<tr>
<td>ACCESS_NO</td>
<td>NUM 8 133 1133</td>
<td>1.</td>
<td>MISCGAPS</td>
<td>TAPE</td>
<td>GAPS78</td>
</tr>
<tr>
<td>DATEST</td>
<td>NUM 8 149 791</td>
<td>DATE7.</td>
<td>DISK</td>
<td>YEAR79 7921</td>
<td></td>
</tr>
<tr>
<td>DATEST</td>
<td>NUM 8 149 791</td>
<td>DATE7.</td>
<td>DISK</td>
<td>YEAR80 3609</td>
<td></td>
</tr>
<tr>
<td>DATEFIN</td>
<td>NUM 8 12 72</td>
<td>DATE7.</td>
<td>DISK</td>
<td>YEAR79 7921</td>
<td></td>
</tr>
<tr>
<td>DATEFIN</td>
<td>NUM 8 12 72</td>
<td>DATE7.</td>
<td>DISK</td>
<td>YEAR80 3609</td>
<td></td>
</tr>
<tr>
<td>DATEFIN</td>
<td>NUM 8 4 18</td>
<td>DATE7.</td>
<td>TAPE</td>
<td>GAPS77</td>
<td></td>
</tr>
<tr>
<td>DATEFIN</td>
<td>NUM 8 4 18</td>
<td>DATE7.</td>
<td>TAPE</td>
<td>GAPS78</td>
<td></td>
</tr>
<tr>
<td>EFFECTA</td>
<td>CHAR 20</td>
<td>1 4</td>
<td>MISCGAPS</td>
<td>TAPE</td>
<td>GAPS77</td>
</tr>
</tbody>
</table>

**Figure 1.**
Generating pseudo-random variates from theoretical probability distributions with specified parameters is a necessary component of stochastic and probabilistic simulation modeling and analysis. SAS provides the capability of generating uniformly distributed pseudo-random variates between 0 and 1 via the UNIFORM function. These pseudo-random numbers in turn can be used to generate pseudo-random variates from nine additional distributions: exponential, gamma, normal with user specified mean and variance, triangular, uniform with user specified a and b values, Weibull, geometric (discrete), Poisson (discrete), and uniform (discrete). The gamma variates generator also can be used to emulate the Erlang-K, chi-square, beta, F, and student's t distributions with appropriate parameter selection.

The nine pseudo-random variate generators are implemented as SAS functions described in the following pages. Each function is a FORTRAN program based on algorithms or heuristics suggested by Shannon /3/, Fishman /1/, or Phillips /2/. Most function invocations require a uniformly distributed random number and one or more parameters, returning a pseudo-random variate for the theoretical distribution. For example, a variate from the gamma distribution with shape \( \alpha \) and scale \( \beta \) is generated by the following SAS code:

```sas
RN=UNIFORM(0);
ALPHA=1; BETA=0.2;
VARIATE=GAMMAV(ALPHA,BETA,RN)
```

Exceptions are NORMAV which requires a normally distributed RN value, and POISSV which requires a seed for its own internal random variates generator.

A description of the form, result, density or mass, mean, variance, restrictions, and examples of each of the nine pseudo-random variate generators can be found in the pages following.

Source listings are available on request.

Reference:

PROC JOIN RATHER THAN MERGE

Procedure JOIN merges two datasets in the relational sense. JOIN produces one observation for each possible combination of the BY variables. For example:

```
=| dataset A | dataset B |
--|--|---|--|--|
| YR | S | T | YR | S |
| 1 | A | X | 1 | A |
| 1 | B | Y | 1 | B |
| 2 | C | Z | 3 |
```

If we MERGE datasets A and B by the variable YR, the result is:

```
=| dataset C |
--|--|--|--|
| YR | S | T |
| 1 | A | X |
| 1 | B | Y |
| 2 | C | Z |
```

However, if we PROC JOIN datasets A and B by the variable YR, the result is:

```
=| dataset C |
--|--|--|--|
| YR | S | T |
| 1 | A | X |
| 2 | B | Y |
```

Notice that nonmatched observations are deleted under PROC JOIN. PROC JOIN should not be used indiscriminately because very large datasets can result. A description of the syntax, usage, and results of PROC JOIN can be found in the pages following.

NEW SAS DATE FUNCTIONS

Eight new date functions have been developed to produce fiscal year, fiscal year quarter, calendar week, and sequential weekday values. Functions for the two most common fiscal year definitions (starting July 1 and starting October 1) assist the SAS user interested in financial applications. Calendar week functions assist users whose applications are week-oriented (such as personnel payroll, facility scheduling, etc.) For example, Saturday January 3 1981 falls in the first calendar week of the year; Sunday January 4 1981 falls in the second calendar week of the year; Thursday December 31 1981 falls in the fifty-third week of the year. Sequential weekday functions return the sequential order of the weekday. For example, Thursday January 1 1981 is the first Thursday; Thursday January 8 1981 is the second Thursday; Thursday December 31 1981 is the fifty-third Thursday.

A description of the form, result, restrictions, and examples of each of the eight date functions can be found in the pages following. Source listings are available on request.
RANDOM VARIATE GENERATORS FOR CONTINUOUS DISTRIBUTIONS

NAME: EXPVN -- exponential variates generator
FORM: EXPVN(\(\theta\), RN) 2 numeric arguments
RESULT: The result of this function will be a pseudo-random variate. The variates generated by EXPVN appear to be distributed with:

DENSITY: \(f(x) = \frac{e^{-x/\theta}}{\theta}\) for \(x \geq 0\)

MEAN: \(\theta\)

VARIANCE: \(\theta^2\)

RESTRICTIONS: \(\theta > 0\), \(0 < \text{RN} \leq 1\) (uniformly distr.)

EXAMPLES: EXPVN(2, 0.01) is 9.2103
EXPVN(2, 0.70) is 0.71335

REFERENCES: /3/ p 203-4, /7/ p 181-8, /10/ p 444

NAME: GAMNAV -- gamma variates generator
FORM: GAMNAV(\(\alpha\), \(\beta\), RN) 3 numeric arguments
RESULT: The result of this function will be a pseudo-random variate. The variates generated by GAMNAV appear to be distributed with:

DENSITY: \(f(x) = \frac{\alpha^{\alpha-1} e^{-x/\beta}}{\Gamma(\alpha)}\)

MEAN: \(\frac{\alpha}{\beta}\)

VARIANCE: \(\frac{\alpha}{\beta^2}\)

EXAMPLES: GAMNAV(1, 1, 0.45) is 0.800470
GAMNAV(3, 0.2, 0.45) is 0.595131

REFERENCES: /1/ p 203-4, /3/ p 360-3, /4/ p 444

NAME: NORMAV -- normal variates generator
FORM: NORMAV(M, S, RN) 3 numeric arguments
RESULT: The result of this function will be a pseudo-random variate. The variates generated by NORMAV appear to be distributed with:

DENSITY: \(f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-M)^2}{2S^2}}\)

MEAN: \(M\)

VARIANCE: \(S^2\)

EXAMPLES: NORMAV(1, 2, 0.00) is 1.00
NORMAV(1, 2, 0.67) is 2.34

REFERENCES: /1/ p 211-3, /3/ p 359-60

NAME: TRIANAV -- triangular variates generator
FORM: TRIANAV(a, b, c, RN) 4 numeric arguments
RESULT: The result of this function will be a pseudo-random variate. The variates generated by TRIANAV appear to be distributed with:

DENSITY: \(f(x) = \begin{cases} \frac{2(x-a)}{(b-a)(c-a)} & \text{for } a \leq x \leq b \\ \frac{2(c-x)}{(c-b)(c-a)} & \text{for } b \leq x \leq c \end{cases}\)

EXAMPLES: TRIANAV(1, 3, 5, 0.20) is 2.26491
TRIANAV(1, 3, 5, 0.50) is 3.0

REFERENCES: /1/ p 202-4

NAME: UNIFOV -- uniform variates generator
FORM: UNIFOV(a, b, RN) 3 numeric arguments
RESULT: The result of this function will be a pseudo-random variate. The variates generated by UNIFOV appear to be distributed with:

DENSITY: \(f(x) = \frac{1}{b-a}\)

MEAN: \(\frac{a+b}{2}\)

VARIANCE: \(\frac{(b-a)^2}{12}\)

EXAMPLES: UNIFOV(1, 5, 0.20) is 1.80
UNIFOV(1, 5, 1.00) is 5.00

REFERENCES: /1/ p 200-2, /3/ p 357

NAME: WEIBUV -- weibull variates generator
FORM: WEIBUV(\(\alpha\), RN) 3 numeric arguments
RESULT: The result of this function will be a pseudo-random variate. The variates generated by WEIBUV appear to be distributed with:

DENSITY: \(f(x) = \frac{\alpha x^{\alpha-1} e^{-x/\beta}}{\beta\Gamma(\alpha)}\)

MEAN: \(\beta\Gamma\left(\frac{\alpha}{\beta} + 1\right)\)

VARIANCE: \(\beta^2\left(\Gamma\left(\frac{\alpha}{\beta} + 2\right) - \beta^2\Gamma\left(\frac{\alpha}{\beta} + 1\right)\right)\)

REFERENCES: /1/ p 357

NAME: NORMAV -- normal variates generator
FORM: NORMAV(M, S, RN) 3 numeric arguments
RESULT: The result of this function will be a pseudo-random variate. The variates generated by NORMAV appear to be distributed with:

DENSITY: \(f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-M)^2}{2S^2}}\)

MEAN: \(M\)

VARIANCE: \(S^2\)

EXAMPLES: NORMAV(1, 2, 0.00) is 1.00
NORMAV(1, 2, 0.67) is 2.34

REFERENCES: /1/ p 211-3, /3/ p 359-60

NAME: TRIANAV -- triangular variates generator
FORM: TRIANAV(a, b, c, RN) 4 numeric arguments
RESULT: The result of this function will be a pseudo-random variate. The variates generated by TRIANAV appear to be distributed with:

DENSITY: \(f(x) = \begin{cases} \frac{2(x-a)}{(b-a)(c-a)} & \text{for } a \leq x \leq b \\ \frac{2(c-x)}{(c-b)(c-a)} & \text{for } b \leq x \leq c \end{cases}\)

EXAMPLES: TRIANAV(1, 3, 5, 0.20) is 2.26491
TRIANAV(1, 3, 5, 0.50) is 3.0

REFERENCES: /1/ p 202-4

NAME: UNIFOV -- uniform variates generator
FORM: UNIFOV(a, b, RN) 3 numeric arguments
RESULT: The result of this function will be a pseudo-random variate. The variates generated by UNIFOV appear to be distributed with:

DENSITY: \(f(x) = \frac{1}{b-a}\)

MEAN: \(\frac{a+b}{2}\)

VARIANCE: \(\frac{(b-a)^2}{12}\)

EXAMPLES: UNIFOV(1, 5, 0.20) is 1.80
UNIFOV(1, 5, 1.00) is 5.00

REFERENCES: /1/ p 200-2, /3/ p 357

NAME: WEIBUV -- weibull variates generator
FORM: WEIBUV(\(\alpha\), RN) 3 numeric arguments
RESULT: The result of this function will be a pseudo-random variate. The variates generated by WEIBUV appear to be distributed with:

DENSITY: \(f(x) = \frac{\alpha x^{\alpha-1} e^{-x/\beta}}{\beta\Gamma(\alpha)}\)

MEAN: \(\beta\Gamma\left(\frac{\alpha}{\beta} + 1\right)\)

VARIANCE: \(\beta^2\left(\Gamma\left(\frac{\alpha}{\beta} + 2\right) - \beta^2\Gamma\left(\frac{\alpha}{\beta} + 1\right)\right)\)

REFERENCES: /1/ p 357
EXAMPLES: WEIBULL(3,4,0.01) is 6.6549
          WEIBULL(3,6,0.70) is 2.63673

REFERENCES: /1/ p 211

RANDOM VARIATE GENERATORS FOR DISCRETE DISTRIBUTIONS

NAME: GEOMEV -- geometric variates generator
FORM: GEOMEV(p,RN) 2 numeric arguments
RESULT: The result of this function will be an integer pseudo-random variate. The variates generated by GEOMEV appear to be distributed with:

MASS: \( f(x=p) = p \times (1-p)^x \) for \( x=0,1,\ldots \)

MEAN: \( p/(1-p) \)

VARIANCE: \( p/(1-p)^2 \)

RESTRICTIONS: 0< p<1, 0<RN<1 (uniformly distr.)

EXAMPLES: GEOMEV(0.5,0.1) is 3
          GEOMEV(0.5,0.5) is 1

REFERENCES: /1/ p 222-4

NAME: POISSV -- poisson variates generator
FORM: POISSV(\lambda,seed) 2 numeric arguments
RESULT: The result of this function will be an integer pseudo-random variate. The variates generated by POISSV appear to be distributed with:

MASS: \( f(x=0) = e^{-\lambda}/x! \) for \( x=0,1,\ldots \)

MEAN: \( \lambda \)

VARIANCE: \( \lambda \)

POISSV generates its own subsequent uniformly distributed random variates based on the initial seed value using a multiplicative congruential method. These variates are for internal use. The user must alter the seed prior to each call.

RESTRICTIONS: \( \lambda > 0 \); seed must be odd integer of nine or fewer digits

EXAMPLES: POISSV(3,281) is 0
          POISSV(3,801) is 3

REFERENCES: /1/ p 224-5, /3/ p 352-3, 358-9

NAME: UNIFDV -- uniform discrete variates generator
FORM: UNIFDV(a,b,RN) 3 numeric arguments
RESULT: The result of this function will be an integer pseudo-random variate. The variates generated by UNIFDV appear to be distributed with:

MASS: \( f(x=a,\ldots,b) \) where \( x=a,\ldots,b \) (Integ.)

MEAN: \( (b+a)/2 \)

VARIANCE: \( ((b-a)(b-a+2))/12 \)

RESTRICTIONS: b>a, a and b integer
0< RN<1 (uniformly distr.)

EXAMPLES: UNIFDV(1,5,0.2) is 1
          UNIFDV(1,5,0.21) is 2

REFERENCES: /1/ p 219

NEW DATE FUNCTIONS

NAME: FY -- fiscal year (starting July 1)
FORM: FY(month, calendar year) 2 numeric arguments
RESULT: The result of this function will be the fiscal year based on a starting date of July 1. The resulting fiscal year will be the same as or one greater than the argument 2 year value.

RESTRICTIONS: 1< month<12, year>0

EXAMPLES: FY(6,74) is 74
          FY(7,74) is 75
          FY(1,75) is 75

NAME: FFY -- fiscal year (starting October 1)
FORM: FFY(month, calendar year) 2 numeric arguments
RESULT: The result of this function will be the fiscal year based on a starting date of October 1. The resulting fiscal year will be the same as or one greater than the argument 2 year value.

RESTRICTIONS: 1< month<12, year>0

EXAMPLES: FFY(9,74) is 74
          FFY(10,74) is 75
          FFY(1,75) is 75

NAME: FQTR -- fiscal quarter (starting July 1)
FORM: FQTR(calendar quarter) 1 numeric argument
RESULT: The result of this function will be the fiscal quarter based on a starting date of July 1. The function returns a 1, 2, 3, or 4.

RESTRICTIONS: 1< quarter<4

EXAMPLES: FQTR(1) is 3
          FQTR(4) is 2

NAME: FFQTR -- fiscal quarter (starting October 1)
FORM: FFQTR(calendar quarter) 1 numeric argument
RESULT: The result of this function will be the fiscal quarter based on a starting date of October 1. The function returns a 1, 2, 3, or 4.

RESTRICTIONS: 1< quarter<4

EXAMPLES: FFQTR(2) is 3

NAME: PFWTR -- fiscal week (starting Monday)
FORM: PFWTR(calendar week) 1 numeric argument
RESULT: The result of this function will be the fiscal week based on a starting date of Monday. The function returns a 1, 2, 3, or 4.

RESTRICTIONS: 1< week<4

EXAMPLES: PFWTR(1) is 3
          PFWTR(4) is 2

NAME: PFWT -- fiscal week (starting Thursday)
FORM: PFWT(calendar week) 1 numeric argument
RESULT: The result of this function will be the fiscal week based on a starting date of Thursday. The function returns a 1, 2, 3, or 4.

RESTRICTIONS: 1< week<4

EXAMPLES: PFWT(1) is 3
          PFWT(4) is 2

NAME: PFWQ -- fiscal quarter (starting Monday)
FORM: PFWQ(calendar quarter) 1 numeric argument
RESULT: The result of this function will be the fiscal quarter based on a starting date of Monday. The function returns a 1, 2, 3, or 4.

RESTRICTIONS: 1< quarter<4

EXAMPLES: PFWQ(2) is 3
**NAME:** CALWKY -- calendar week of the year  
**FORM:** CALWKY(date value for first day of year, date value) 2 numeric SAS date values  
**RESULT:** The result of this function will be the physical calendar week of the year in which the date value (argument 2) falls, based on the specified first day of the year (argument 1). For example,  
A=MDY(1,1,81) Jan 1  
B=MDY(2,3,81) Feb 3  
C=MDY(12,31,81) Dec 31  
D=MDY(1,2,81) Jan 2  
E=MDY(1,4,81) Jan 4  
Thus, CALWKY(A,B) is week 6  
CALWKY(A,C) is week 53  
CALWKY(A,D) is week 1  
CALWKY(A,E) is week 2  
**RESTRICTIONS:** both date values ≥ 0, so dates prior to Jan 1 1960 are not evaluated

**NAME:** SEQWDY -- sequential weekday of the year  
**FORM:** SEQWDY(date value for first day of year, date value) 2 numeric SAS date values  
**RESULT:** The result of this function will be the sequential weekday of the year corresponding to the weekday of the date value (argument 2) based on the first day of the year (argument 1). For example,  
A=MDY(1,1,81) Jan 1  
B=MDY(1,31,81) Jan 31  
C=MDY(1,20,81) Jan 20  
D=MDY(12,31,81) Dec 31  
Thus, SEQWDY(A,B) is 5 (5th Saturday)  
SEQWDY(A,C) is 3 (3rd Tuesday)  
SEQWDY(A,D) is 1 (1st Thursday)  
SEQWDY(A,E) is 2 (2nd Thursday)  
SEQWDY(A,F) is 53 (53rd Thursday)  
**RESTRICTIONS:** both date values ≥ 0, so dates prior to Jan 1 1960 are not evaluated

**NAME:** SEQWDY -- sequential weekday of the month  
**FORM:** same as for CALWKY  
**RESULT:** same as for SEQWDM except that first day of month is used for argument 1. The function is an alias for SEQWDM and as such evaluates identically.  
**RESTRICTIONS:** same as for SEQWDM

---

**PROCEDURE JOIN**  
JOIN merges two datasets in the relational sense. The procedure requires a BY statement and optionally one or two variable lists. The procedure produces no printed output, but generates an output dataset. JOIN can produce very large datasets because all possible combinations of the BY variables are considered.  
**USAGE:**  
PROC JOIN DATA=dataset1 DATA=dataset2  
OUT=output dataset;  
BY variable list (common variables);  
VAR1 variables list from dataset 1;  
VAR2 variables list from dataset 2;  
**RESULT:** The output dataset will contain all the BY variables + variables in VAR1 list + variables in VAR2 list. The procedure can generate enormous datasets and has only limited application to most data processing requirements.