

SAS® Randomization Techniques for Clinical Trials

Linda Quinn, Case Western Reserve University

ABSTRACT

There are several times during the course of a clinical trial that randomization is necessary. The SAS® system is capable of being used to meet those needs. Two macros are presented for two particular aspects of a clinical trial. The first application is selection of random samples (e.g. charts, Case Report Forms) to assess and ensure the quality of data entry, to survey for more information, or to choose for treatment. The second application is assignment of patients (or animals, specimens, etc.) to a treatment category. These macros only address simple random sampling, although it is possible to extend these macros to implement stratified sampling. While these macros are currently being used in a clinical trial setting, they are generic enough to be used and/or modified for use in a variety of disciplines, including economic research where initially developed.

AUDIENCE

This paper is intended for SAS users who need to use randomization schemes for selecting samples or determining treatment allocation. Specifically, this paper uses Base SAS data steps, random number functions, and some macro language.

While the concepts and code here are fairly simple, the application and expandability can also be of use to the intermediate and advanced users of SAS.

The applications are presented in a clinical trial format but are easily modified for most situations requiring randomization.

GENERAL TIPS

Within the macros described within are several SAS features which make these macros simpler and yet have many other applications outside of the ones presented.

- CALL SYMPUT can be used to get the number of observations of a dataset.
- (OBS = k) with a SET statement to restrict the data to the first k observations.
- The MOD(,k) function can be used to assign k values to observations.

APPLICATION: SAMPLING

There are many times sampling is required. Selecting Case Report Forms (CRFs) for quality audits, selecting charts for site audits, and selecting random samples of patients are several examples.

In each case there are several parameters to be specified. One parameter is whether the sample is taken with or without replacement. Another parameter is what size sample is needed. This is based on whether a percentage of the population is

desired or a fixed number. Both situations are considered here.

Finally, the sampling distribution needs to be considered. Most sampling needs are met using a uniform distribution. The presented

code uses a uniform distribution but may easily be modified to use one of the other distributions available in SAS. The other two parameters used in this macro are the dataset to be sampled and the random number seed.

THE MACRO: RANSAMP

1	%MACRO RANSAMP(DSN, REPLACE, PERC, NSIZE, SEED);
2	%GLOBAL TOTN;
3	DATA SAMPLE; SET &DSN;
4	ORDER=RANUNI(&SEED);
5	CALL SYMPUT('TOTN', N);
6	PROC SORT DATA=SAMPLE; BY ORDER;
7	%IF &NSIZE = 0 %THEN
8	%LET NSIZE=%EVAL(&PERC * &TOTN / 100);
9	%IF &REPLACE %THEN %Do;
10	DATA REPLACE (KEEP=VALUE);
11	Do I=1 To &NSIZE;
12	VALUE=MAX(ROUND(RANUNI(&SEED)*&TOTN,1),1);
13	OUTPUT;
14	END;
15	PROC SORT DATA=REPLACE; BY VALUE;
16	DATA SAMPLE; SET SAMPLE;
17	VALUE= _N_;
18	PROC SORT DATA=SAMPLE; BY VALUE;
19	DATA SAMPLE; MERGE SAMPLE REPLACE (IN=IN1);
20	BY VALUE;
21	IF IN1;
22	DROP ORDER VALUE;
23	%END;
24	%ELSE %Do;
25	DATA SAMPLE; SET SAMPLE (OBS=&NSIZE);
26	DROP ORDER;
27	%END;
28	%MEND RANSAMP;

%RANSAMP PARAMETERS

The five parameters of the macro call, in order, are:

- DSN: the dataset name to be used to sample from. This may be a two level SAS dataset name or can refer to a dataset in the WORK library.

- **REPLACE:** This is a zero/one flag variable where 0 = sample without replacement and 1 = sample with replacement. A logical test is done in the code; therefore, sampling without replacement will be done as a default unless a 1 is in this field.
- **PERC:** This is the percent of the population to be included in the sample. This will only be used if `NSIZE=0`. Otherwise, `NSIZE` will override this parameter. This should be specified as a percent and not as a decimal fraction. For example, if a 10% sample is desired, `PERC=10`.
- **NSIZE:** This is the fixed size of the sample. No checks are made to see if this number is larger than the population size (`TOTN`). If `NSIZE` is larger than `TOTN` and sampling is without replacement, the entire population will be returned in the sample. If `NSIZE` is larger than `TOTN` and sampling is with replacement, an `NSIZE` sample will be returned. If `NSIZE=0` then a percent of the population as specified by `PERC` will be returned.
- **SEED:** This is the random generator seed. Setting `SEED=0` uses the clock as the seed. If you wish to replicate samples, you may wish to fix `SEED` to a non-zero value.

%RANSAMP: EXAMPLES

`%RANSAMP(TEST,0,0,5,0);` This call will select 5 observations without replacement at random from the dataset `TEST` using the clock as the random generator seed.

`%RANSAMP(TEST,1,0,5,0);` This call will select 5 observations with replacement at random from the dataset `TEST` using the clock as the random generator seed. Recall that sampling with replacement allows the possibility that all five observations will be the same.

`%RANSAMP(TEST,0,10,0,0);` This call will select ten percent of the observations without replacement from the dataset `TEST` using the clock as the random generator seed. Note that if the original dataset, `TEST`, has 100 observations, the sample will have 10 observations and if `TEST` has 198 observations, the sample will have 19 observations. There is no rounding of the percentage. This, of course, could be customized if desired.

%RANSAMP CODE EXPLANATION

The macro creates a new dataset, `SAMPLE`, which is in the work library. This dataset is a subset of your original dataset (`DSN`) and will contain the same number of variables.

A quick walk through of the code reveals that lines 2 through 5 extract the total number of observations in `DSN`. This is stored in the global macro variable `TOTN`. Also, the data set `SAMPLE` is initially created which is equivalent to `DSN` and has one additional variable, `ORDER`. `ORDER` is a random variate. Lines 7 through 12 calculate the sample size desired if using the percentage option. Lines 9 through 23 show how to sample with replacement. Lines 24 through 27 samples without replacement.

The logic goes as follows: Create a duplicate dataset, `SAMPLE`, from the original dataset `DSN`. Add one additional variable, `ORDER`, which is a random variate. Sort `SAMPLE` by the random variate. If sampling without replacement, take the first `NSIZE` observations as your sample. If sampling with replacement, create a dataset with the `NSIZE` observations to take from the original dataset (`REPLACE`). Merge `REPLACE` with `SAMPLE` and keep only those observations that match with the ones in `REPLACE`.

%RANSAMP ENHANCEMENTS

While not shown here, this code can be customized to resort the `SAMPLE` dataset into the original `DSN` order, print dataset `DSN` with a flag showing whether the observation is in the sample or not (note: This requires even more care if sampling is done with replacement), or print out just dataset `SAMPLE`, or change the random generator from `RANUNI`. When one of the parameters is always fixed the code can be abbreviated to reflect one less parameter. For example, a majority of the code can be eliminated if sampling is confined to without replacement.

APPLICATION: TREATMENT ALLOCATION

Assigning patients, animals, or material specimens to treatment groups usually requires some type of randomization, particularly if statistical comparisons are to be made between groups. To simplify matters, the allocation in this macro is such that all groups are of equal sample size. This

is the predominant randomization scheme for clinical trials.

The code is presented in macro format although it could easily be presented as stand-alone code with several macro declarations or if an interactive program was desired, the user could be prompted to supply the values.

`TRTMT` is used to produce a randomization scheme for `NTREAT` possible treatments in blocks of size `BLKSIZE`. The total number of blocks is `BLOCKS`. Therefore, the total number of patients or specimens randomized is `BLOCKS` times `BLKSIZE`. In general, to avoid a string or series of treatments, make `BLKSIZE` the longest string or series desired. While this example only allows for one blocking level, the logic can be extrapolated for multiple nesting of blocks.

For example, if patients will be randomized to one of two groups (treatment or placebo) and there are ten investigational sites participating in the study, it is desired that the patients should be randomized at each site to preserve balance and reduce site to site (between site) bias. Patients will be randomized in blocks of size 10 with one block going to each investigational site. There will be 100 patients in the entire trial, so the size of each block is $100/10 = 10$. Using simple random sampling can leave the possibility of the first 5 patients receiving one treatment and the last five patients in the block receiving the other treatment. To disallow this and make the largest string smaller than 5, reduce `BLKSIZE` appropriately ($BLKSIZE = NTREAT * \text{largest string acceptable}$).

THE MACRO: TRTMT

1	%MACRO TRTMT(DSN, BLOCKS, BLKSIZE, NTREAT, SEED);
2	DATA &DSN;
3	DO BLOCK = 1 To &BLOCKS;
4	DO NUMBER = 1 To &BLKSIZE;
5	RANDOM = RANUNI(&SEED);
6	OUTPUT;
7	END;
8	END;
9	PROC SORT; BY BLOCK RANDOM;
10	DATA &DSN; SET &DSN;
11	TREAT = MOD(N_,&NTREAT) + 1;
12	KEEP BLOCK NUMBER TREAT;
13	PROC SORT; BY BLOCK NUMBER;
14	PROC PRINT;
15	%MEND TRTMT;

%TRTMT PARAMETERS

The five parameters of the macro call, in order, are:

- **DSN:** The dataset name to contain the resulting treatment assignment. This may be a two level SAS name.
- **BLOCKS:** The number of blocks to be created.
- **BLKSIZE:** The size of each block. Note that the total assignments made will be BLOCKS*BLKSIZE.
- **NTREAT:** The number of treatments to be assigned. Remember that the macro does equal assignment. However, if BLKSIZE is not evenly divisible by NTREAT, the macro will still execute but the treatment assignment will be unbalanced.
- **SEED:** The random number generator seed. SEED=0 will use the clock. To reproduce results a non-zero value should be used.

%TRTMT: EXAMPLES

%TRTMT(SAMPLE,10,10,2,0); This call will create 10 blocks of size 10 and assign each to one of two treatments using the random generator clock. The resulting dataset is SAMPLE.

%TRTMT(SAMPLE,1,21,3,0); This call will create 1 block of size 21 and assign each to one of three treatments using the random generator clock. The resulting dataset is SAMPLE. This would be useful if randomizing only one group of patients.

%TRTMT(SAMPLE,2,21,4,0); This call will create 2 blocks of size 21 and assign each to one of four treatments using the random generator clock. Notice that 21 is not divisible evenly by four. Therefore, there the treatment allocation will be unbalanced and unpredictable. The resulting dataset is SAMPLE.

%TRTMT CODE EXPLANATION

The macro creates a dataset, DSN which contains the treatment allocation scheme.

Lines 2 through 8 create a the dataset, DSN, with BLOCKS by BLKSIZE observations. A random variable is also created. DSN has three variables (BLOCK, NUMBER, and RANDOM).

Line 9 sorts the dataset within BLOCK by the random variate. Lines 10 through 12 assigns the treatment. The MOD function is used for treatment assignment. This gives equal allocation when BLKSIZE is divisible by NTREAT. Otherwise, the allocation scheme will be invalid.

Line 13 sorts DSN in the order of assignment, NUMBER within BLOCK and Line 14 prints the resulting dataset.

%TRTMT ENHANCEMENTS

While this macro is simple, it has been easy to modify for extensions not described here, including unbalanced designs, stratified assignment, and non uniform sampling distributions.

PROC PLAN

PROC PLAN constructs designs and randomizes plans for nested and crossed experiments. PROC PLAN could be used in lieu of macro TRTMT. While the results will not be identical, they will be similar. Initially, it was simpler to write the macro than to pick up PROC PLAN and create the same design. Once accustomed to it, it was easier to use PROC PLAN than to modify the macro for stratified allocation or other design changes.

CONCLUSIONS

Random number generators, macros calls, and PROC PLAN have made the sampling tasks associated with clinical trials much easier than using a random numbers table. In addition, sampling tasks associated with design of experiments can be simplified with these tools as well. Macros can be designed to be flexible for what the end user needs and still allow for future expansion with minimal time investment.

CONTACT INFORMATION

Linda M. Quinn
Research Associate

Division of General Internal Medicine
and Program in Health Care Research
Case Western Reserve University and
Veterans Affairs Medical Center
10701 East Boulevard
Cleveland, OH 44106

Phone: 216-225-4184 (H)
216-791-3800 x4814 (W)
FAX: 216-231-3476
e-mail: lmq2@po.cwru.edu

SAS is a registered trademark of SAS Institute, Inc., in the USA and other countries. ® indicates USA registration.