Paper 251-27

A Randomization-test Wrapper for SAS® PROCs

David L. Cassell, CSC

ABSTRACT

SAS/STAT® procedures are often used in settings where the underlying model assumptions are not really met. Permutation tests can permit one to assess correct p-values in many of these cases, but too often the total number of permutations is unmanageable. Instead, a randomization test using a random subset of all permutations can be used. This paper demonstrates a flexible, extensible pair of 'wrapper' macros which allow the user to use almost any SAS® PROC and get a randomization p-value for the desired test at the back end. The wrapper is a front macro to create the data set of replicates with a randomized dependent variable, and a back macro to process the results of the SAS® PROC and compute the p-value according to a randomization test. These bracket the already-written SAS® procedure or process, which requires minimal alteration. We demonstrate the wrapper on several SAS® procedures, using standard output data sets as well as tables pulled out using ODS.

THE PROBLEM

While SAS/STAT® procedures provide a wide range of facilities for data analysis, only too often the data refuse to cooperate. We wish to perform an analysis of variance, but the errors are not normal, or do not have variances equal across the treatment categories. Standard data transforms are often applied. Welch's variance-weighted ANOVA is sometimes used for one-way ANOVA problems. Nonparametric procedures are often applied, such as the Kruskal-Wallis test available in PROC NPAR1WAY. But none of these are applicable all the time, and most become problematic when more complex analyses are desired.

Consider a dataset of biodiversity measures taken from a probability sample of sites. Unfortunately, some sites have no biota in the taxa of interest, yielding a large number of zeroes in the resulting dataset. The poorer habitats are more likely to have zeroes for diversity measures, so these zeroes have to be preserved in the data. The data are therefore clearly non-normal, and may even have serious issues with heterogeneity of variance.

An alternative is the standard permutation test. This test uses all possible distinct permutations of the dependent variable, holding the independent vairables fixed. Performing the original statistical analysis on all such permutations allows one to evaluate how the actual structure of the data compares to random re-arrangements of the data.

Unfortunately, a typical full permutation test is too time-consuming. Consider the above example with a small dataset. Assume there are 34 observations in three habitats, with 12 observations in the first two habitat types and 10 in the third habitat. The number of distinct permutations is thus 34!/(12!12!10!) which is unfortunately a whopping 355 trillion distinct permutations to assess. For a medium-sized dataset, the number of permutations becomes utterly unmanageable. An alternative to the full permutation test is what is often called a randomization test. It uses a Monte Carlo approach to select a random subset of the total number of permutations, so that the computations can be done in a reasonable amount of time.

THE APPROACH

The macro code presented here functions as a simple wrapper around the desired SAS® code. Suppose that the code we wish to run looks like:

```
proc glm data=outrand outstat=outstat1;
  class habitat;
  model FishS = habitat;
  means habitat / tukey hovtest ;
  run;
```

And suppose that we see a p-value of 0.00031 . Is this actually significant, given the strong non-normality of the data? One can run a randomization test to assess the legitimacy of this result.

In order to execute the randomization test, we add three lines to the above code:

```
%rand_gen(indata=yb98fish,outdata=outrand,
  depvar=FishS,numreps=10000,seed=12345678)

proc glm data=outrand noprint outstat=outstat1;
  by replicate;
  class habitat;
  model FishS = habitat;
  means habitat / tukey hovtest ;
  run;

%rand_anl(randdata=outstat1,
  where=_source_='Habitat' and _type_='SS3',
  testprob=prob,testlabel=Model F test)
```

And the resulting randomization test is printed in the SAS log. The entire randomization test is subsumed in the two macros `%rand_gen` and `%rand_anl` . Let's see how this works.

THE CODE

The `%rand_gen` macro works by first copying the original dataset &INDATA over the required number of times [which is also specifiable as a macro parameter]. Each record of each copy of the dataset has a random number rand_dep, so that the records can be randomly ordered within each replicate. Then finally the replicates are appended to the original dataset. While the original dataset is read, the values of the independent variable are inserted into a temporary array in the original order of the dataset. While the replicate dataset is read in, each record receives a new value of the independent variable from the array. This provides an ordered dependent variable associated with a randomized set of records, which is mathematically equivalent to randomly ordering the dependent variables while leaving the records untouched.

```
%macro rand_gen(
  indata=_last_,
  outdata=outrand,
  depvar=y,
  numreps=1000,
  seed=0);

  /* Get size of input dataset into macro
     variable &NUMRECS */

  proc sql noprint;
    select count(*) into :numrecs from
&INDATA;
    quit;

  /* Generate &NUMREPS random numbers for each
     record, so records can be randomly
     sorted within each replicate */

  data __temp_1;
    retain seed &SEED ;  drop seed;
    set &INDATA;
    do replicate = 1 to &NUMREPS;
      call ranuni(seed,rand_dep);
      output;
      end;
    run;

  proc sort data=__temp_1;
    by replicate rand_dep;
    run;

  /* Now append the new re-orderings to the
     original dataset.  Label the original
     as Replicate=0, so the %RAND_ANL macro
     will be able to pick out the correct
     p-value.  Then use the ordering of
     __counter within each replicate to
     write the original values of &DEPVAR ,
     thus creating a randomization of the
     dependent variable in every replicate. */

  data &OUTDATA ;
    array deplist{ &NUMRECS } _temporary_ ;
    set &INDATA(in=in_orig)
        __temp_1(drop=rand_dep);
    if in_orig then do;
      replicate=0;
      deplist{_n_} = &DEPVAR ;
      end;
    else &DEPVAR =
      deplist{ 1+ mod(_n_,&NUMRECS) };
    run;

%mend rand_gen;
```

At this point the output dataset &OUTRAND is ready for use. The variable REPLICATE is used as a by-variable for the analysis, and the NOPRINT option is added to suppress the many copies of the analysis, but otherwise no changes are made in the original SAS® code. This means that the time required to run a randomization test will not be significantly greater than the time needed without the macro wrapper.

The output dataset OUTSTAT1 then becomes the input for the `%rand_anl` macro:

```
%macro rand_anl(
  randdata=outrand,
  where=,
  testprob=probf,
  testlabel=F test,);

  data _null_;
    retain pvalue numsig numtot 0;
    set &RANDDATA end=endofile;
    %if "&WHERE" ne ""
      %then where &WHERE %str(;) ;
    if Replicate=0 then pvalue = &TESTPROB ;
    else do;
      numtot+1;
      numsig + ( &TESTPROB < pvalue );
      end;
    if endofile then do;
      ratio = numsig/numtot;
      put "Randomization test for &TESTLABEL "
      %if "&WHERE" ne "" %then "where &WHERE";
        " has significance level of "
        ratio 6.4 ;
      end;
    run;

%mend rand_anl;
```

The macro `%rand_anl` has to be able to select only the relevant records in the dataset OUTSTAT1 above. The format of the OUTSTAT table includes for each value of the by-variable, a record with the error sum of squares, a record with the Type I sum of squares for the independent variable HABITAT, and a record with the Type III sum of squares for the independent variable HABITAT. Since this is only a one-way ANOVA, the Type I and Type III sums of squares will be the same. But for a more involved analysis, one would need to be able to select precisely the desired test of significance. The &WHERE parameter allows one to pull out only the relevant records, here selecting the records where we have _source_='Habitat' and _type_='SS3'. The null data step then calculates the proportion of records for which a randomized replicate has an even more significant p-value than the original dataset did. This is, in essence, the p-value for the randomization test.

The null data step could have been built to compare the values of the original F-test and all the replicated F-tests. In this case, such a comparison would yield exactly the same result. However, some tests may be one-sided, like a t-test with a one-sided alternative hypothesis, and may be comparing to see if the test statistic for the replicates is lower than the the t-statistic for the original data. The p-value would still be compared correctly, but the test statistic might not. Thus it makes sense to compare the p-values instead of the test statistics.

Finally the macro provides the results of the randomization test in the SAS log file:

```
Randomization test for Model F test where
_source_='Habitat' and _type_='SS3' has
significance level of 0.0006
```

So, in this case, the randomization test supports the result found in the analysis of variance, and yields a p-value which can withstand some peer review despite the clear non-normality of the data.

As an aside, the original PROC GLM on the small [n=34] data set took 0.33 seconds to run, while the entire macro wrapper took 2.11 seconds to run, and the interior PROC GLM took 0.98 seconds to run all 1001 levels of the by-variable. While this will not extend linearly with the size of the analysis data set, the wrapper is still much faster than a thousand separate iterations of the PROC could be.

EXTENDING THE EXAMPLE

Now suppose that one wished instead to examine whether the variances were equal across the habitats. PROC GLM provides a number of tests of heterogeneity-of-variance. Let's use Bartlett's as an example, even though this is known to be less robust than some other alternatives. If we perform the original analysis and find a p-value of, say, 0.0325, is this really significant given the non-normality of the data? We run the randomization test on the HOVFTest table available from ODS:

```
%rand_gen(indata=yb98fish,outdata=outrand,
depvar=FishS,numreps=1000,seed=12345678)

ods listing close;
ods output HOVFTest=hov;

proc glm data=outrand /* noprint */
   outstat=outstat1;
  by replicate;
  class habitat;
  model FishS = habitat;
  means habitat / tukey hovtest ;
  run;

ods output close;
ods listing;

%rand_anl(randdata=outstat1,
where=_source_='Habitat' and _type_='SS3',
testprob=prob,testlabel=Model F test)

%rand_anl(randdata=hov,
where=Source='Habitat',
testprob=probf,testlabel=Bartletts test)
```

Note that the NOPRINT option is not used here. In order to access ODS tables, this option cannot be in force. Instead, one can use the ODS statement

```
        ods listing close;
```

to end the generation of lines to the Output window until the

```
        ods listing;
```

statement turns output back on.

The same `%rand_anl` line as before is present, but now there is a new line added. The wrapper macros do not have a limit on the number of subsequent tests to be run on the generated tables. And this time, the resulting randomization test yields:

```
Randomization test for Bartlett's test where
Source='Habitat' has significance level of
0.0840
```

So now we have reason to doubt whether the errors are really heteroskedastic.

A FURTHER EXTENSION

If one had a more complex model to analyze, then this wrapper could be used to examine different randomization tests on multiple effects. As an example, let's repeat the previous example, but with two main effects and an interaction term. This can be done using the classic OUTSTAT= option which has been available for a long time. We will also pull out a test for the overall ANOVA, using ODS. Note that we do not attempt to perform Bartlett's test here, since that is not an option for anything except one-way ANOVAs.

```
%rand_gen(indata=yb98fish,outdata=outrand,
  depvar=FishS,numreps=1000,seed=3743586)

ods output OverallANOVA=overall;
/* Have to remove NOPRINT to make this work */

proc glm data=outrand /* noprint */
    outstat=outstat1;
  by replicate;
  class habitat month;
  model FishS = habitat month habitat*month;
  run;

ods output close;

%rand_anl(randdata=outstat1,
  where=_source_='Habitat' and _type_='SS3',
  testprob=prob,testlabel=main effect test)

%rand_anl(randdata=outstat1,
  where=_source_='month' and _type_='SS3',
  testprob=prob,testlabel=main effect test)

%rand_anl(randdata=outstat1,
  where=_source_='Habitat*month'
        and_type_='SS3',
  testprob=prob,
  testlabel=interaction effect test)

/* The dataset OVERALL has variables Source,
  ProbF, etc.  We want the lines where Source
  is equal to 'Model' to get the p-values out.
  The p-values are missing elsewhere.     */

%rand_anl(randdata=overall,
  where=Source='Model',
  testprob=ProbF,
  testlabel=Overall ANOVA test)
```

And in the log will appear the randomization test output for each of the four separate tests. [Note that the real log will have other text interspersed with the messages from the `%rand_anl` macro.]

```
Randomization test for main effect test
where _source_='Habitat' and _type_='SS3'
has significance level of 0.0220

Randomization test for main effect test
where _source_='month' and _type_='SS3' has
significance level of 0.8140

Randomization test for interaction effect
test where _source_='Habitat*month' and
_type_='SS3' has significance level of
0.8780

Randomization test for Overall ANOVA test
where _source_='Model' has significance
level of 0.0080
```

Furthermore, this example can be extended to show how flexible the macro wrapper is. Consider the case where one wishes to obtain a different analysis, using another SAS® PROC, say for a test which would normally not be available in PROC GLM. If one wanted to perform a randomization test on differences between means of specific values of the habitat variable, one would need a PROC which could output multiple comparisons and associated p-values.

PROC MIXED can do this. The ODS table Diffs provides variables which name the two levels of the variable, a p-value, and an adjusted p-value. For the variable Habitat, the variable names in the Diffs table will be Habitat and _Habitat . The comparisons are done in sorted order, so the three levels of Habitat [N, U, and Z] yield three comparisons:

 Habitat='N' and _Habitat='U'
 Habitat='N' and _Habitat='Z'
 Habitat='U' and _Habitat='Z'

For a single comparison of habitats U and Z, we do not need to use an adjusted p-value to take into account the multiple comparisons being performed. So we will use the value Probt in this example. This is just as well, since the limitation of this approach to randomization testing is that all the tests are in essence single-comparison evaluations which do not take the number of comparisons into account.

```
%rand_gen(indata=yb98fish,outdata=outrand,
  depvar=FishS,numreps=1000,seed=3743586)

ods output Diffs=diffs;
ods listing close;
  /* turn off output for PROC MIXED */

proc mixed data=outrand;
  by replicate;
  class habitat;
  model FishS = habitat;
  lsmeans habitat / pdiff ;
  run;

ods output close;
ods listing;

%rand_anl(randdata=diffs,
  where=(Habitat='U' & _Habitat='Z'),
  testprob=Probt,testlabel=Habitats U vs. Z)
```

This causes the following text to appear in the log:

```
Randomization test for Habitats U vs. Z
where  (Habitat='U' & _Habitat='Z') has
significance level of 0.0210
```

CONCLUSIONS

The macros %rand_gen and %rand_anl provide a simple, flexible way to provide randomization tests in place of the standard SAS® analyses, yielding tests which are more appropriate when the usual model assumptions will not be met. The wrapper is fast and efficient, and encapsulates the underlying analyses so that the programmer needs to make few changes in the original code.

ACKNOWLEDGEMENTS

SAS® and SAS/STAT® are registered trademarks of SAS Institute, Inc. in the USA and other countries.

CONTACT INFORMATION

The author may be contacted by mail at

David L. Cassell
CSC, c/o U.S. EPA
200 SW 35th St.
Corvallis, OR 97339

or by e-mail at

Cassell.David@epa.gov