Ultra-Fast Permutation-Based Multiple Comparison Adjustment for Weighted Experiments in SAS

David R. Judkins
Abt Associates, 4550 Montgomery Ave, Suite 800 North, Bethesda, MD 20814

Abstract
Permutation-based adjustments for multiple outcomes, treatments or outcomes work well for experiments with complex designs involving varying probabilities of selection and randomization, clustering and stratification. However, SAS/MultTest does not accept weights or clusters. By embedding SAS/SurveySelect and SAS/SurveyReg into SAS macros, it is possible to use the permutation approach on a much wider class of problems. However, this programming approach can be slow, particularly if the experimental sample size is large and the number of permutations is also large. Based on the ideas of Opdyke (2011) for fast permutation tests, I have developed a new SAS macro, FastAnalyzeTwo that is orders of magnitude faster than using a combination of SurveySelect and SurveyReg. It performs permutation-based adjustments for random-effects analysis of multi-centre trials where both centres and participants were selected with a probability design and there are dozens of centres and millions of participants. It will handle two outcomes. In this paper, I discuss the key tricks and share some simulation study results. I also discuss stabilization of variance estimates when there are few degrees of freedom and how to incorporate this into the fast adjustment.

Key Words: Multicentre trials, design-based inference, variance stabilization

1. Setting

The setting for this research was a multi-centre randomized trial involving a million subjects from ten centres. Both centres and subjects were randomly selected for the study according to a complex design that makes the weights non-ignorable even with knowledge of sampling strata. The study included a rich set of baseline covariates. This design was selected because of the objective to have very strong validity – both internal and external. We wanted to be able to predict with high confidence the hypothetical effect of expanding the program to all centres in the nation. Consistent with this objective, we decided to analyze the experiment with SAS/SurveyReg with unbiased sampling weights and treating the centres as PSUs. We also decided to control the family-wise error rate (FWER) for the two primary endpoints. Because the two primary endpoints have a strong negative correlation, we opted for permutation-based multiple-comparison adjustment (PBMCA) (Westfall and Young, 2003) to achieve FWER control.

Here we ran into problems. SAS/MultTest implements BPMCA, but it accepts neither weights nor clusters. This paper talks about three approaches to solving this problem. They are all SAS macros. After explaining the three approaches, performance statistics for the two faster macros are presented in Section 5. Section 6 closes with an application of the fastest macro to simulate alternates approaches to PBMCA after variance stabilization.

2. Brute Force Solution

The first approach we tried was to simply wrap a SAS macro around SAS/SurveySelect coupled with SAS/SurveyReg. The macro first selects a stratified simple random sample without replacement of the entire sample to serve as a permuted pseudo-treatment sample. Here, the centres are treated as strata. Example code is as follows:

```
PROC SURVEYSELECT DATA= DataIn METHOD=srs SAMPRATE=.5
        OUT=DataOut OUTALL NOPRINT SEED= &&seed_&iter;
        STRATA Centre;
RUN;
```

Covariates, outcomes, and weights stayed together. Only the treatment label is changed in the permutation. Note that original probabilities of assignment varied widely, but for the rerandomization, everyone was given a probability of assignment to treatment of 0.5. The dataset produced by this code has a special variable on it called "Selected." This is a binary 1/0 variable that then serves as permuted-treatment status. Special care is taken in the seed management. For each permutation, there is a separate seed that itself arises from a random number stream.

Each of the two outcomes is then analyzed on the output dataset using the code:

```
PROC SURVEYREG DATA= DataOut;
   CLUSTER Centre;
   CLASS Selected;
   MODEL Endpoint = &CovariateList Selected/ solution;
   WEIGHT StudyWeight;
RUN;
```

Using the output delivery system (ODS) (not shown), the absolute value of the t-statistic for "Selected" is captured for each of the two outcomes. Let these be designated as t_{1p} and t_{2p}, where p indexes the permutations. Also, let $t_{\max,p} = \max(t_{ip}, t_{2p})$ and t_1, t_2, and $t_{\max} = \max(t_1, t_2)$ be the actual-sample counterparts.

WLOG, assume that $t_{\max} = t_1$. Then the permutation-adjusted p-values for the effect of the intervention on the two outcome are

$$\tilde{p}_1 = \frac{1}{G}\sum_p if\,(t_{\max} \geq t_{\max.p}) \text{ and } \tilde{p}_2 = \max\left\{\tilde{p}, \frac{1}{G}\sum_p if\,(t_2 \geq t_{2p})\right\}.$$

This code was fairly easy to write and is very flexible, but it was painfully slow to run even a few times on the million records in the study. Westfall in private communication recommended 20,000 permutations in order to control FWER. This might have taken weeks to run once. System updates and interruptions would be a serious worry in running this. Moreover, we wanted to be able to run simulations to satisfy ourselves that the procedure was really working. This would require running the 20,000 permutations at least 2000 times. So we realized that we needed a faster approach.

3. Acceleration with Data Reduction via Working Models

To improve the speed of the macro, my next step was to reduce both the width and depth of the input data. This reduced the input dataset from something like 30 covariates on a million records to zero covariates on 4000 records. This reduction involved two linked ideas. One idea was to residualize the covariates from the endpoints using a working model. The second idea was then to summarize the residualized endpoints on random subsets of the subjects. These random subsets might be thought of as subcentres that are group randomized. Each arm of each centre is divided into 200 random subcentres, which then translate to 4000 subcentres given that there are two arms in each of 10 centres.

SAS/Rank is run once (ranking on a random number) to generate the 4000 subcenrtres. The brute force macro can then be run on 4000 records instead of a million and with no covariates instead of 30. It is important to residualize first rather than summarizing both covariates and endpoints to the subcentre level and then performing the regression on the summarized data. If the latter were done, much of the power of the covariates would be lost because their coefficients would be attenuated.

The residualization is performed by separately regressing each endpoint on the covariates without treatment status in the model. Alternatively, the covariate-only model can be fit on the control sample by itself and then applied to the full sample to calculate residuals. It is not clear which procedure is best. Under the strong null hypothesis of no effects of treatment whatsoever, the best estimates of the covariate coefficients will be obtained by fitting the working model on the entire sample, but if treatment were, say to increase the variance of the endpoint without shifting the mean, it might be better to fit the working model only on the control sample. I did not use weights in the equation to generate the residuals. Unclear if this is necessary in a working model. However, I did summarize both the residuals and the weights to the subcentre level. The regressions at the subcentre level then were run as:

```
PROC SURVEYREG DATA= SummResiduals;
   CLUSTER Centre;
   CLASS Selected;
   MODEL Endpoint = Selected/ solution;
   WEIGHT SummWeight;
RUN;
```

The data reduction was very successful in accelerating the performance of the procedure. A job with 2,000 permutations can now run in 3.5 minutes instead of weeks. This allowed a small-scale simulation study to confirm that the procedure does at least approximately control FWER. However, the simulation took five days to run 2000 repetitions of 200 permutations, far fewer than recommended by Westfall.

4. Ultra-Fast Opdyke-Style Adaptation

In search of a way to further improve the speed of this algorithm, I came across the work of J.D. Opdyke (2011). He published an ultra-fast SAS macro for performing permutation tests. Although Opdyke's macro was designed for non-parametric testing, it

was obvious that some of the techniques he used to accelerate permutation tests could also be used to accelerate PBMCA of design-based parametric inference of multi-centre trials. Opdyke had three key insights:

1. Goodman and Hedetniemi (1977) published as ultra-fast algorithm for selecting a SRSWOR. This algorithm seems to have gone unnoticed by the writers of SAS/SurveySelect.
2. SAS handles looping through large arrays much faster than looping through the equivalent information when stored across records, so it is much better to work horizontally than vertically.
3. Further speed gains are realized if arrays are temporary (not associated with a set of variable names).

To exploit these insights for PBMCA of design-based parametric inference of multi-centre trials, I combined the data reduction ideas of my second macro (SlowAdjustTwo) with a manual coding of the key formulae in SAS/SurveyReg to operate in parallel on a small set of (nearly) sufficient statistics that operates on permutations generated in the same way as Opdyke's macro using the Goodman and Hedetniemi SRSWOR algorithm.

The core steps in this algorithm are as follows:

1. Generate L random groups within each combination of centre and trial arm. Refer to these $2mL$ random groups as subcentres, where m is the number of centres in the trial.
2. Form residuals from working models for the two outcomes in terms of covariates only. These working models might be fit on the entire sample (omitting treatment status) or just on the control sample.
3. Summarize residuals and weights to the subcentre level. Designate these as $\bar{r}_{Oij}$ and w_{ij} for $i = 1,\ldots,m;\ j = 1,\ldots L;\ O = 1,2$ where i indexes centre, j indexes subcentre, and O indexes endpoint.

4. Calculate the estimated treatment effects, associated variance estimates, and associated test statistics as:

$$\hat{\beta}_O = \frac{\sum_i \bar{r}_{Oti}}{\sum_i w_{ti}} - \frac{\sum_i \bar{r}_{Oci}}{\sum_i w_{ci}},$$

$$\hat{v}_{\beta_O} = \frac{m}{m-1}\left[\frac{\sum_i w_{ti}^2\left(\bar{r}_{Oti} - \bar{r}_{Ot}\right)^2}{w_t^2} + \frac{\sum_i w_{ci}^2\left(\bar{r}_{Oci} - \bar{r}_{Oc}\right)^2}{w_c^2} + -2\frac{\sum_i w_{ti} w_{ci}\left(\bar{r}_{Oti} - \bar{r}_{Ot}\right)\left(\bar{r}_{Oci} - \bar{r}_{Oc}\right)}{w_t\, w_c}\right],$$

$$T_O = \frac{\hat{\beta}_O}{\sqrt{\hat{v}_{\beta_O}}},$$

where

$$\bar{r}_{Oti} = \sum_j w_{ij}\bar{r}_{Oij}\delta_{ij}\ ,\ \bar{r}_{Oci} = \sum_j w_{ij}\bar{r}_{Oij}\left(1-\delta_{ij}\right),$$

$$\bar{r}_{Ot} = \frac{\sum_i w_{ti}\bar{r}_{Oti}}{w_t},\ \bar{r}_{Oc} = \frac{\sum_i w_{ci}\bar{r}_{Oci}}{w_c}$$

$$w_{ti} = \sum_j w_{ij}\delta_{ij}\ ,\ w_{ci} = \sum_j w_{ij}\left(1-\delta_{ij}\right)\ ,\ w_t = \sum_i w_{ti}\ ,\ w_c = \sum_i w_{ci}\ ,$$

the *t/c* subscripts indicate treatment status (treatment or control), and δ_{ij} indicates the treatment status for the subcentre. For the derivation of the variance estimation formula, see the derivation of formula 137 of Chapter 7 of the second edition of Sukhatme and Sukhatme (1970) or the derivation of equation 2.1 of chapter 6 of Volume 2 of Hansen, Hurwitz, and Madow (1953). In the latter, note that the u-terms are weighted up to universe totals.

5. In a single data step:
   a. Read in the subcentre data by centre.
   b. Declare 3 temporary arrays to hold the summarized weights and the two summarized residuals for all the subcentres in the current centre.
   c. Declare 4 large arrays to hold permuted centre-levels weighted totals of the two endpoints for each pseudo-treatment status:

      $$\bar{r}_{Otip} = \sum_j w_{ij}\bar{r}_{Oij}\delta_{ijp},$$

      $$\bar{r}_{Ocip} = \sum_j w_{ij}\bar{r}_{Oij}\left(1-\delta_{ijp}\right) \text{ for } i = 1,\ldots,m;\ j = 1,\ldots L;\ p = 1,\ldots,G;O = 1,2$$

      where the *t/c* subscripts here indicate permuted treatment status, *p* indexes permutation, and δ_{ijp} is the permuted treatment status for the subcentre.
   d. Start a loop for the desired *G* permutations.
   e. Start a subloop for the *L* subcentres to be assigned to the pseudo-control sample.
   f. Draw one unselected subcentre to join the pseudo control sample for that permutation. Update $\bar{r}_{1tip}, \bar{r}_{1cip}, \bar{r}_{2tip}, \bar{r}_{2cip}$ accordingly.
   g. Swap places in the three data arrays for the newly selected pseudo-control subcentre and the lowest-indexed unselected subcentre. This is the key trick of the Goodman and Hedetniemi algorithm. This action keeps the selected and unselected sets of subcentres within compact subarrays. This makes it faster to draw the next pseudo-control. Moreover, once the desired *L* pseudo-control subcentres have been selected for a given permutation, the array is ready for the next

permutation without any housework. See Figure 1 for a visualization of the process.

h. Once the desired *L* pseudo-control subcentres have been selected for each of the *G* desired permutations, both loops are ended and the final versions of $\overline{r}_{1tip}, \overline{r}_{1cip}, \overline{r}_{2tip}, \overline{r}_{2cip}$ are written out to a centre-level record in array form.

i. This is repeated for each centre, resulting in a temporary dataset with *m* records and 4*G* variables.

6. Using this fat but shallow dataset, permuted counterparts of the quantities in step 4 are calculated as:

$$\hat{\beta}_{Op} = \frac{\sum_i \overline{r}_{Otip}}{\sum_i w_{ti}} - \frac{\sum_i \overline{r}_{Ocip}}{\sum_i w_{ci}}$$

$$\hat{v}_{\beta_{Op}} = \frac{m}{m-1}\left[\frac{\sum_i w_{ti}^2\left(\overline{r}_{Otip} - \overline{r}_{Otp}\right)^2}{w_t^2} + \frac{\sum_i w_{ci}^2\left(\overline{r}_{Ocip} - \overline{r}_{Ocp}\right)^2}{w_c^2} - 2\frac{\sum_i w_{ti} w_{ci}\left(\overline{r}_{Otip} - \overline{r}_{Otp}\right)\left(\overline{r}_{Ocip} - \overline{r}_{Ocp}\right)}{w_t\, w_c}\right]$$

$$T_{Op} = \frac{\hat{\beta}_{Op}}{\sqrt{\hat{v}_{\beta_{Op}}}},$$

where

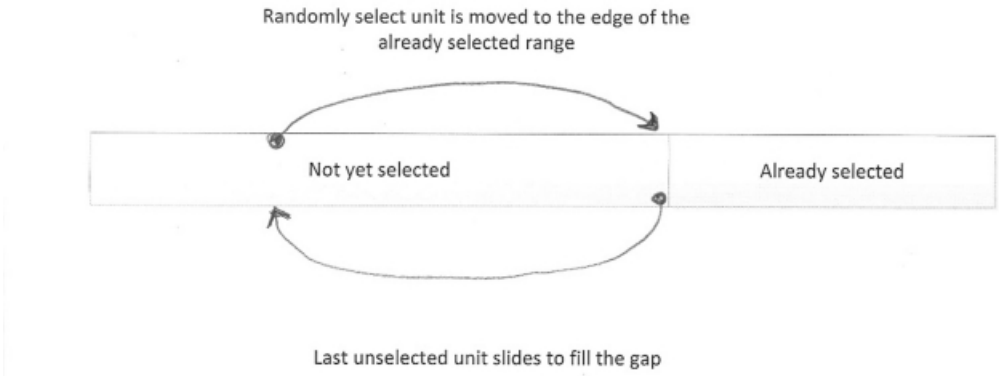
$$\overline{r}_{Otip} = \sum_j w_{ij}\overline{r}_{Oijp}\delta_{ijp}\ ,\ \overline{r}_{Ocip} = \sum_j w_{ij}\overline{r}_{Oijp}\left(1-\delta_{ijp}\right),$$

$$\overline{r}_{Otp} = \frac{\sum_i w_{ti}\overline{r}_{Otip}}{w_t}\ ,\ \overline{r}_{Ocp} = \frac{\sum_i w_{ci}\overline{r}_{Ocip}}{w_c}.$$

Note, in retrospect, perhaps it would be better to use the following permuted weight sums in the foregoing calculations:

$$w_{tip} = \sum_j w_{ij}\delta_{ijp}\ ,\ w_{cip} = \sum_j w_{ij}\left(1-\delta_{ijp}\right)\ ,\ w_{tp} = \sum_i w_{tip}\ ,\ w_{cp} = \sum_i w_{cip}\ .$$

However, this was not done, and simulation results were good without this refinement.



- Figure 1. Visualizing the SRSWOR algorithm of Goodman and Hedetniemi

5. Performance

To test the speed benefits of the Opdyke-style modifications to my macro, I ran several comparisons on a dataset of with close 968,713 records, the size of the experiment I was working on. Table 1 shows the comparison of SlowAnalyzeTwo and FastAnalyzeTwo for different numbers of permutations. Times for the brute force approach are not shown because I could not afford enough CPU time.

Table 1. Performance of SlowAnalyzeTwo and FastAnalyzeTwo in CPU Minutes

	Number Permutations		
Macro Version	2,000	10,000	20,000
SlowAnalyzeTwo	3.49	14.68	39.08
FastAnalyzeTwo	0.15	0.36	1.31
Ratio	23	41	30

One might ask, why go to the trouble of the Opdyke-style modifications for a savings of just 39 CPU minutes. The cost of the software development is many times the cost of 39 CPU minutes. However, if one wants to simulate the performance of the entire system, 2000 is pretty much the minimum useful simulation sample size, and 20,000 times 39 CPU minutes obviously is a huge difference. Few computer systems are stable enough to allow runs that last 8 weeks without interruption. Clearly, parallel processing is an option, but it is far more difficult to program. This system works in open SAS macro code, which will run on any modern processor under any standard operating system with minimal fuss.

6. Variance Stabilization

As an application of the speed of the Opdyke-style modifications, consider the following problem of variance stabilization. In a multi-centre experiment with 10 sites, there are only 9 degrees of freedom available for estimation of v_β. This does not mean that type I unconditional error rates are compromised. However, when treatment effects for the same outcome on different subgroups with similar sample sizes are calculated, it can lead to glaring discrepancies in estimated standard errors. In this context, one gets the strong intuitive feeling that one or both of the estimated standard errors must be "wrong," by which one means badly estimated. We refer to such a collection of estimated standard errors as a discordant ensemble.

To reduce the frequency of these discordant ensembles, consider estimating the standard error as the maximum of the estimated conditional and unconditional standard errors. The idea is similar in spirit to "variance generalization," a technique that has been in use at the Census Bureau for at least 40 years (Wolter, 1985). The idea of variance generalization is to use the ensemble to improve all the variance estimates in the ensemble by modeling the variance estimates in the ensemble as a function of expected sample size and then to replace each direct variance estimate by the model prediction of the variance estimate. This is a form of smoothing of variance estimates. Variance generalization is beneficial when the degrees of freedom are limited and the variance model is approximately correct. It is easiest to find well-fitting variance models for estimated proportions of people belonging to various classes. It works less well when dealing with continuous outcomes such as earnings and benefit amounts. Given that the survey at hand focused on such continuous estimates, we opted for the weaker form of variance smoothing of simply placing a reasonable lower bound on variance estimates. Unless there is strong negative intraclass correlation (a rarity), conditional variances should be lower than unconditional variances. Since there are many more degrees of freedom available for estimating conditional variances than unconditional variances, when this relationship is violated, it is safe to assume that the estimated unconditional variance estimate is the problem and that it is too small. In this case, substituting the conditional variance estimate for the unconditional variance estimate will tend to reduced discordance in any ensembles containing the very small unconditional variance estimates. It will also have the effect of making the inferential procedures slightly conservative, but it was decided that this was an acceptable price to pay for having fewer incidences of discordant ensembles of variance estimates.

The estimated unconditional variance is given above and reflects the variability if effectiveness of the intervention across centres. The conditional variance ignores this component of variance and is estimated as[1]

$$\hat{v}_{w\beta_O} = \frac{1}{w_t^2}\left[\sum_i \frac{n_{ti}\left(1-w_{ti}\bar{r}_{Oti}/n_{ti}\right)}{n_{ti}-1}\sum_j\left(w_{ij}\bar{r}_{Oij}\delta_{ij}-w_{ti}\bar{r}_{Oti}/n_{ti}\right)^2 + \right.$$
$$+\sum_i \frac{n_{ti}\left(w_{ti}\bar{r}_{Oti}/n_{ti}\right)}{n_{ti}-1}\sum_j\left(w_{ij}\bar{r}_{Oij}\delta_{ij}-w_{ij}\delta_{ij}-w_{ti}\bar{r}_{Oti}/n_{ti}+w_{ti}/n_{ti}\right)^2 +$$
$$\left.+\sum_i \frac{n_{ti}\left(w_{ti}\bar{r}_{Oti}/n_{ti}\right)\left(w_{ti}\bar{r}_{Oti}/n_{ti}-1\right)}{n_{ti}-1}\sum_j\left(w_{ij}\delta_{ij}-w_{ti}/n_{ti}\right)^2\right] +$$
$$\frac{1}{w_c^2}\left[\sum_i \frac{n_{ci}\left(1-w_{ci}\bar{r}_{Oci}/n_{ci}\right)}{n_{ci}-1}\sum_j\left(w_{ij}\bar{r}_{Oij}\left(1-\delta_{ij}\right)-w_{ci}\bar{r}_{Oci}/n_{ci}\right)^2 + \right.$$
$$+\sum_i \frac{n_{ci}\left(w_{ci}\bar{r}_{Oci}/n_{ci}\right)}{n_{ci}-1}\sum_j\left(w_{ij}\bar{r}_{Oij}\left(1-\delta_{ij}\right)-w_{ij}\left(1-\delta_{ij}\right)-w_{ci}\bar{r}_{Oci}/n_{ci}+w_{ci}/n_{ci}\right)^2 +$$
$$\left.+\sum_i \frac{n_{ci}\left(w_{ci}\bar{r}_{Oci}/n_{ci}\right)\left(w_{ci}\bar{r}_{Oci}/n_{ci}-1\right)}{n_{ci}-1}\sum_j\left(w_{ij}\left(1-\delta_{ij}\right)-w_{ci}/n_{ci}\right)^2\right]$$

The stabilized variance estimator is then
$\hat{v}_{stab\beta_O} = \max\left(\hat{v}_{\beta_O}, \hat{v}_{w\beta_O}\right)$.

The challenge for using this stabilized variance estimate in the context of PBMCA was whether to use the stabilization on the permutations as well as on the actual estimates.

We considered five rules, of which the three most interesting were:

1. $\hat{v}_{stab\beta_{Op}} = \hat{v}_{\beta_{Op}}$
2. $\hat{v}_{stab\beta_{Op}} = \max\left(\hat{v}_{\beta_{Op}}, \hat{v}_{w\beta_O}\right)$
3. $\hat{v}_{stab\beta_{Op}} = \begin{cases} \hat{v}_{\beta_{Op}} & \text{if } \hat{v}_{\beta_O} \geq \hat{v}_{w\beta_O} \\ \max\left(\hat{v}_{\beta_{Op}}, \hat{v}_{w\beta_O}\right) & \text{if } \hat{v}_{\beta_O} < \hat{v}_{w\beta_O} \end{cases}$

(We also considered recalculating the conditional variance estimates on the permuted samples, but these showed very little variation across replicates and so this idea was dropped to improve speed.) Having the fast version of PBMCA allowed us to determine that the first and third rules both work well under realistic conditions but that the second

[1] This is a very convoluted version of the standard equation for conditional variance. I programmed it this way in the hope that permutations would be faster to calculate because the weight and the outcome would not need to be multiplied together on every permutation. This formula is friendlier for calculations on the go.

rules performs very poorly. Prior to making the simulations runs, I would have guessed that the second rule would work best because it follows the general rule of bootstrap, permutation and other resampling procedures of replicating original sample procedures on all resamples. This turns out to be a case where the general rule does not apply. Stabilizing the variance estimates for the permuted pseudo-effects turns to be a bad idea unless the original sample estimate was too small and needed to be brought up the level of the conditional variance estimate.

References

Opdyke, J.D. (2011). Permutation tests (and sampling without replacement) orders of magnitude faster using SAS, *InterStat*, January 2011.

Goodman, S. and Hedetniemi (1977). *Introduction to the Design and Analysis of Algorithms*. New York: McGraw-Hill.

Hansen, M.H., Hurwitz, W.N., and Madow, W.G. (1953). *Sample Survey Methods and Theory, Volume II*. New York: John Wiley and Sons, Inc.

SAS Institute (2014). SAS OnLineDoc® 9.4. Cary, NC: SAS Institute.

Wolter, K.M. (1985). *Introduction to Variance Estimation*. New York: Springer-Verlag.

Sukhamte, P.V. and Sukhatme, B.V. (1970). *Sampling Theory of Surveys with Applications*. Ames, Iowa: Iowa State University Press.