Journal of Statistical Software

April 2017, Volume 77, Code Snippet 1. *doi: 10.18637/jss.v077.c01*

A SAS Macro for Covariate-Constrained Randomization of General Cluster-Randomized and Unstratified Designs

Erich J. Greene
Yale University

Abstract

Ivers *et al.* (2012) have recently stressed the importance to both statistical power and face validity of balancing allocations to study arms on relevant covariates. While several techniques exist (e.g., minimization, pair-matching, stratification), the covariate-constrained randomization (CCR) approach proposed by Moulton (2004) is favored when clusters can be recruited prior to randomization. **CCRA** V1.0, a macro published by Chaudhary and Moulton (2006), provides a SAS implementation of CCR for a particular subset of possible designs (those with two arms, small numbers of strata and clusters, an equal number of clusters within each stratum, and constraints that can be expressed as absolute mean differences between arms). This paper presents a more comprehensive macro, **CCR**, that is applicable across a wider variety of designs and provides statistics describing the range of possible allocations meeting the constraints in addition to performing the actual random assignment.

Keywords: constrained randomization, restricted randomization, balanced allocation, covariate balance, cluster-randomized trials, stratified group-randomized trials, SAS.

1. Introduction

In a recent methodological review of allocation techniques for cluster-randomized trials,[1] Ivers *et al.* (2012) stressed the importance to both statistical power and face validity of balancing allocations to study arms on relevant Covariates. While several techniques exist (e.g., minimization, pair-matching, stratification), their review favored the covariate-constrained

[1]Some authors, e.g., Ivers *et al.* (2012), prefer the term *cluster* while others, e.g., Moulton (2004), prefer the term *group*. While this paper uses *cluster*, either a cluster or a group is essentially a unit to be randomized, so the two terms can be used interchangeably in the context of this macro.

randomization (CCR) approach proposed by Moulton (2004) in situations where clusters can be recruited prior to randomization and the research team has sufficient statistical support.

To date, the only published program for performing CCR is **CCRA** V1.0 by Chaudhary and Moulton (2006). This macro can perform CCR under the following conditions: two study arms, small numbers of strata and clusters per stratum (their macro attempts to produce and test every possible allocation, which can quickly become a prohibitively large set), equal numbers of clusters per stratum (assumed by their algorithm for calculating differences at the overall level), and constraints that can be expressed as absolute differences between means of continuous variables. However, many practical clinical trials where CCR might be desirable may involve unequal-sized strata, differences in counts rather than means (e.g., balancing clusters on binary or categorical variables) or expressed as a proportion of the overall or stratum mean (rather than a fixed number), more than a few clusters and strata, and/or more than two arms. The macro presented here, **CCR**, allows for all these possibilities, as well as running more quickly and flexibly than **CCRA** V1.0 in large designs and providing statistics describing the range of allowed allocations (as well as performing the actual random assignment).

While the following discussion will generally be in terms of stratified cluster-randomized designs, **CCR** can be used to randomize an unstratified design by coding each cluster to a single, common stratum. It can also be used to randomize individuals rather than clusters; from the macro's perspective, a cluster is simply a unit to be randomized.

2. Algorithm

CCR's program flow largely follows that of **CCRA** V1.0, with additional capabilities, checks, and displays built in.

Step 0: Check macro arguments for validity and consistency.

Step 1: Generate ways of allocating clusters across study arms within each stratum, either splitting clusters as evenly as possible across arms (the default) or using fixed user-provided arm sizes. For small to moderate-sized clusters, complete sets can feasibly be generated and checked, but in designs where strata contain large numbers of clusters, random subsets (100,000 by default) of possible allocations are generated and checked for each stratum. (The idea of employing random sampling to make CCR more tractable has previously been proposed by Nietert, Jenkins, Nemeth, and Ornstein 2009.)

Step 2: Compute the sums and means of each covariate for each arm of every within-stratum allocation under consideration. (Though some of these calculations are not necessary for the stratum-level constraints – any one constraint is based on either the sum or the mean – they also feed into calculating the overall constraints, which **CCR** allows to be of a different type.)

Step 3: Eliminate allocations that fail to meet the stratum-level constraints and display the distribution of between-arm differences in each stratum for each covariate. If the constraints cannot be met in one or more strata, exit gracefully.

Step 4: Generate a list of overall allocations to check. In a brute force approach (such as **CCRA** V1.0's implementation), stratum-level allocations would be combined factorially

to create a complete listing, but since the number of possible overall allocations grows exponentially with the number of strata, the running time and memory requirements can quickly grow infeasible in multi-stratum studies. Instead, **CCR** generates a random subset (100,000 by default) of the possible overall allocations and then works with every allocation in that sample; however, if the number of possible overall allocations is smaller than the requested sample, **CCR** will fall back to brute force.

Step 5: Compute the sums and means of each covariate for each arm of every overall allocation under consideration.

Step 6: Eliminate allocations that fail to meet the overall constraints or unbalance the total number of clusters in each arm (unless unbalanced arm sizes were specified by the user) and report the proportion of satisfactory allocations. If no overall allocations meet the constraints, exit gracefully; otherwise display the distribution of overall differences for each covariate.

Step 7: Count the number of times each pair of clusters appears together in the same arm. Display descriptive statistics for how frequently clusters are constrained to the same arm and list pairs of clusters that always, never, often (by default, in 75% or more of satisfactory allocations), or rarely (by default, in at most $\frac{1}{2 \cdot [\#\text{ of arms}]}$ of satisfactory allocations, i.e., half chance or below) appear in the same arm.

Step 8: Select one final satisfactory allocation, merge the arm assignments into the original data set, and display the absolute and relative differences of each covariate between each arm (calculated directly from the original data).

3. Using CCR

The syntax for a call to **CCR** is

```
%ccr(dataset, stratid, clustid,
     covariates,
     variable types,
     stratum constraints,
     overall constraints,
     option1 = val1, ..., optionN = valN);
```

where `dataset` contains the input data, `stratid` names the variable identifying strata, `clustid` names the variable identifying clusters, and `covariates` are the names of the covariate variables. The items in any list argument (i.e., `covariates`, `variable types`, `stratum constraints`, `overall constraints`, and certain optional parameter values) should be separated by spaces.

This ...	... specifies that ...
`s1`	arm totals must differ by no more than one (i.e., split as evenly as possible).
`s98.6`	arm totals must differ by no more than 98.6.
`sf.5`	arm totals must differ by no more than half the mean arm total.
`m15`	arm means must differ by no more than 15.
`mf.2`	arm means must differ by no more than 20% of the overall mean.
`any`	any arm means or totals are acceptable.

Table 1: Examples of constraint syntax.

3.1. Choosing variable types

CCR recognizes two broad categories of variables, named `d` (mnemonic for dichotomous, dummy, or discrete; intended for variables with few enough values for frequency tables to be useful) and `c` (mnemonic for continuous but also appropriate for discrete, many-valued interval measures; intended for variables where reporting descriptive statistics would be more useful than listing every possibility). The choice does not affect the randomization process but allows the user to choose how the distributions of differences are presented for each covariate. Sample output for `d` covariates is presented in Tables 5 and 7 in Section 4.1; for `c` covariates, see Tables 12 and 13 in Section 4.2.

3.2. Setting constraints

Each list of constraints in the **CCR** call must contain one constraint per covariate. Constraints can vary on two dimensions: mean/sum and absolute/fractional. A mean constraint compares covariates based on their averages in each arm, while a sum constraint compares covariates based on their totals. In an absolute constraint, the difference (in means or sums) must be less than the value specified, while in a fractional constraint, the difference must be less than a specified fraction of the overall mean. In any case, the constraint always acts on the absolute value of the difference.

A constraint's type is specified by its leading characters. If the first character is `m`, the constraint acts on means, while if the first character is `s`, the constraint acts on sums. If the second character is `f`, the constraint is treated as fractional, otherwise it is treated as absolute. The numeric constraint comes after the type.

See Table 1 for examples of constraints. Note that there is also a special constraint, `any`, which places no restrictions whatsoever.

Stratum-level and overall constraints can be chosen independently, so it is possible to constrain a covariate only at the stratum or overall level by setting the other level's constraint to `any`; in particular, an unstratified design can be randomized by assigning each cluster to a single stratum and setting every stratum constraint to `any`.[2] Difference distributions for a covariate are not displayed for a level where that covariate is unconstrained.

3.3. Optional parameters

While testing every stratum-level allocation and every combination of viable stratum-level

[2] As a convenience, if the `stratid` argument is left empty, **CCR** will create a special stratum id variable called `_sid` that is always equal to 1 and change the stratum constraints to `any`.

allocations as **CCRA** V1.0 does is feasible for small design spaces, larger designs require restricting attention to a random subset of possible allocations. The numbers of allocations to test are set with `ssample` (for sampling possible stratum-level allocations within strata) and `osample` (for sampling overall allocations); the defaults are 100,000, and settings of 0 force **CCR** to test every possible allocation. (Every possible allocation will also be tested if there are fewer than `ssample`/`osample` of them; for example, using the default settings in a two-arm study, there would be no sampling within a stratum with 18 clusters, since $\binom{18}{9} = 92{,}378 < 100{,}000$.)

By default, **CCR** assigns clusters across two arms, but this can be increased by setting `arms` to the desired number of arms.

For some designs, it may not be desirable to divide clusters into arms as evenly as possible. By setting `armsize` to a list of variables, the number of clusters from each stratum to be allocated to each arm can be fixed. The first variable should contain the number of clusters (which can vary by stratum) to be allocated to arm 1, the second variable should contain the number for arm 2, and so forth through arm $n - 1$. (The size of the last arm is fixed by the others and does not need to be specified.)

The complete set of optional parameters is described in Appendix A.

3.4. Caveats

Though the input dataset and variables can have any names, names beginning with underscores should be avoided, as **CCR** names all its data sets and variables with leading underscores. In particular, data sets in the `work` library whose names start with underscores are deleted when **CCR** starts to prevent data sets from previous runs from lingering; otherwise, if the macro fails to complete, results from a previous run could be mistaken for current output.

The stratum identifier, cluster identifier, and covariates must be numeric (though covariates can be dummy-coded and id codes need not be consecutive), and cluster id codes should be unique across the entire design (not repeated across strata).

With **CCR**, it is generally necessary to code for and explicitly constrain all levels of a multilevel variable (rather than code for every level but one as one might in multiple regression). For example, in attempting to assign 10 clusters across two arms as evenly as possible with respect to a three-level covariate, if three clusters are in level 1, three are in level 2, and four are in level 3, coding for and constraining only levels 1 and 2 would allow assigning two level 1 clusters and two level 2 clusters to the same arm, leaving one level 3 cluster in that arm and the other three in the opposite arm (splitting them 1-3 rather than 2-2 while maintaining an overall 5-5 split between arms).

CCR calculates the numbers of possible within-stratum and overall allocations and checks them exhaustively if there are fewer than the requested samples. However, these calculations will cause an overflow on current computers if there are 2^{1024} ($\approx 1.798 \times 10^{308}$) or more possible allocations.[3] Though a design space that large seems unlikely in practice, the checks can be disabled by setting `sizecheck` to 0 if the issue arises.

To reduce execution time, allocations are sampled with replacement. For large designs, the

[3]Currently, the maximum length of a numeric SAS variable is 8 bytes, or 64 bits (see SAS Institute Inc. 2014a, p. 273), and $2^{1024} - 1$ is the largest number that can be stored with standard 64-bit encoding (see IEEE Computer Society, Microprocessor Standards Committee and Engineers 2008, p. 8).

probability of duplicating allocations is vanishingly small, but if the number of possible allocations is not substantially greater than the square of the desired sample size, testing the entire space is generally the safest strategy.[4] However, if sampling is employed both within strata and overall, a too-small number of possible overall allocations can instead be addressed by increasing the within-stratum sample size.

Some amount of trial and error may be needed to balance running time and memory usage with obtaining enough acceptable allocations to get a sense of their rarity and distribution (i.e., how restrictive the constraints actually are). The number of possible allocations to be sampled from is printed to the log before the allocations are generated (unless size checks have been disabled), so if macro execution bogs down, this information may provide some guidance on how many generated and tested allocations are too many for the computer being used.

While sampling from the within-stratum and overall allocations greatly reduces the overall execution time, **CCR** does not employ a shortcut for assessing whether pairs of clusters are always, often, rarely, or never assigned to the same arm. The running time for this process scales as the square of the number of clusters and can be the majority of the total execution time for designs with many tens or hundreds of clusters.[5]

4. Example output

CCR has been used to perform the randomization in the STrategies to Reduce Injuries and Develop confidence in Elders (STRIDE) trial, for which the author is a member of the biostatistics working group. Patients are being recruited for this trial from 86 clinical practices across 10 participating healthcare systems, with each system providing between 5 and 12 participating practices. The intervention being tested is a practice change, so randomization was performed at the practice level, with practices acting as the clusters and healthcare systems acting as the strata. (See the STRIDE website, `http://stride-study.org/`, for more information about the trial.)

The practice-level data used in the randomization are provided in the file `v77c01.sas`. The actual randomization was performed balancing urban vs. rural, majority English-speaking vs. majority non-English-speaking, majority white vs. majority non-white, and small vs. medium vs. large practices; actual practice sizes are also presented to illustrate the use of continuous covariates in a hypothetical, follow-up example. The meaning of each variable is given in Table 2, and the distribution of the discrete covariates in each healthcare system is presented in Table 3.

The complete code to run these examples is provided in the file `v77c01.sas`, and the full, default printed output from these examples (using a seed parameter, `seed = 22571`, for reproducibility[6]) is provided in the file `code.pdf` in the supplementary material. Additional

[4] In sampling k of n possible allocations with replacement, the probability of any two sampled allocations being duplicates is $1/n$, and there are $\binom{k}{2}$ pairs of sampled allocations, so the expected number of duplications is $k(k-1)/2n$, which is negligible when $k^2 \ll n$. When sampling is employed, this expected number of duplications is printed to the log, and keeping it well below 1 is recommended.

[5] Using a dataset provided by a reviewer with 223 clusters in two strata, **CCR** ran in about 7 minutes on the author's computer (with an 8-core 2.40 GHz processor and 32 GB of RAM); over 6 minutes of that time was spent evaluating the restrictions on cluster coincidence.

[6] Note that since the trial is ongoing and many study personnel must remain blinded to condition assignments, this is not the seed used for actual study randomization.

Variable	Meaning
`practice`	Practice identifier.
`site`	Healthcare system (site) identifier.
`rur`	1 if practice is rural, otherwise 0.
`urb`	1 if practice is urban, otherwise 0.
`eng`	1 if practice is majority English-speaking, otherwise 0.
`neng`	1 if practice is majority non-English-speaking, otherwise 0.
`wht`	1 if practice is majority white, otherwise 0.
`nwht`	1 if practice is majority non-white, otherwise 0.
`size`	Number of eligible patients in practice.
`tert1`	1 if practice is in the lowest tertile by `size`, otherwise 0.
`tert2`	1 if practice is in the middle tertile by `size`, otherwise 0.
`tert3`	1 if practice is in the highest tertile by `size`, otherwise 0.

Table 2: Variables in the example dataset.

site	practices	rur	urb	eng	neng	wht	nwht	tert1	tert2	tert3
1	8	7	1	8	0	8	0	1	5	2
2	8	0	8	2	6	2	6	2	5	1
3	11	0	11	11	0	10	1	5	4	2
4	9	0	9	9	0	0	9	5	1	3
5	12	0	12	12	0	12	0	1	1	10
6	8	0	8	8	0	8	0	1	2	5
7	8	1	7	8	0	8	0	7	1	0
8	5	0	5	5	0	5	0	0	1	4
9	8	0	8	8	0	8	0	2	5	1
10	9	0	9	9	0	9	0	4	4	1

Table 3: Distribution of practices and discrete covariates in the example dataset.

output can be printed by setting the `verbose` and `binomsig` parameters; see Appendix A for details.

4.1. Example 1: Actual STRIDE randomization

The STRIDE randomization was constrained to assign practices to two arms, intervention and control, such that the numbers of practices at each level of each discrete variable were as even as possible at both the within-healthcare-system and overall levels. However, we can see from the data that balancing majority English-speaking against majority non-English-speaking will provide no additional constraints beyond those from balancing majority white against majority non-white and intervention against control,[7] so the process can be simplified

[7]In nine of the ten healthcare systems, all of the practices are majority English-speaking, so for them, balance on language is guaranteed. In the tenth, site 2, the two majority non-English-speaking practices are also majority non-white and the six majority English-speaking practices are also majority white, so enforcing the race constraint will balance language as well.

by dropping this variable. The ultimate macro call to perform this randomization is:

```
%ccr(stride, site, practice,
     rur urb wht nwht tert1 tert2 tert3,
     d   d   d   d    d     d     d,
     s1  s1  s1  s1   s1    s1    s1,
     s1  s1  s1  s1   s1    s1    s1);
```

Summary variables

The summary variables **CCR** displays follow this nomenclature:

Arm summaries are named `_l#covar`, where `l` is `s` for a stratum-level summary (mean or sum) or `o` for an overall summary, `#` is an arm, and `covar` is the covariate being summarized.

Differences are named `_dlI_Jcovar`, where `I` and `J` are the arms being compared and `l` and `covar` are as above. Differences are computed as [arm I] − [arm J] (and will be negative if the value in arm `J` is larger), scaled by the sum or mean across all arms if the constraint is fractional. (For example, a fractional difference of means between arms 1 and 2 would be calculated as $\frac{\text{[arm 1 mean]}-\text{[arm 2 mean]}}{\text{[mean across all arms]}}$.)

Absolute differences are named `_adlI_Jcovar` and are the absolute values of the differences just described.

Stratum-level allocation summaries

The first table in the output presents the number and frequency of acceptable allocations in each stratum, as illustrated in Table 4. We see that although fewer than one tenth of one percent of possible allocations met the stratum-level constraints, there are still more than 10^{18} potential allocations. The constraints had markedly more impact on some strata than others: Nearly seven eighths of the possible allocations within site 3 were eliminated, while every possible allocation within site 7 met the constraints.

The remaining tables for the stratum-level allocations present the distribution of allocations with respect to covariates, as shown in Table 5. If constraints cannot be met in one or more strata, distributions for the remaining strata are still presented.

Table 5 shows how the mid-sized practices can be allocated within their healthcare systems. (Similar tables are produced for `urb`, `rur`, `wht`, `nwht`, `tert1`, and `tert3`.) Site 1 has five mid-sized practices that are required to be split as evenly as possible; half the satisfactory allocations (20 of the 40 noted in Table 4) place two practices in arm 1 and three in arm 2 while the other half do the reverse, and the difference is always ±1. Site 10 has four mid-sized practices that are always split with two in each arm and a difference of 0.

If no allocations satisfy the within-stratum constraints, the output ends here.

site	Possible allocations	Checked allocations	Fraction checked	Acceptable allocations	% acceptable of checked
1	70	70	1	40	57.143%
2	70	70	1	24	34.286%
3	924	924	1	120	12.987%
4	252	252	1	180	71.429%
5	924	924	1	504	54.545%
6	70	70	1	40	57.143%
7	70	70	1	70	100.000%
8	20	20	1	12	60.000%
9	70	70	1	40	57.143%
10	252	252	1	72	28.571%
	1.8225E21	1.8225E21	1	1.0113E18	0.055%

Table 4: Summary of acceptable within-stratum allocations.

site	_s1tert2	_s2tert2	_sd1_2tert2	Frequency
1	2	3	-1	20
1	3	2	1	20
2	2	3	-1	12
2	3	2	1	12
3	2	2	0	120
4	0	1	-1	90
4	1	0	1	90
5	0	1	-1	252
5	1	0	1	252
6	1	1	0	40
7	0	1	-1	35
7	1	0	1	35
8	0	1	-1	6
8	1	0	1	6
9	2	3	-1	20
9	3	2	1	20
10	2	2	0	72

Table 5: Stratum-level summary of a `d` covariate.

Overall allocation summaries

Table 6 shows how many overall allocations were tested and how many met the overall constraints. (The last column, giving the overall percentage of acceptable allocations, is the product of the acceptable-of-checked percentages from this table and Table 4.)

If any satisfactory allocations are found, their distributions are then presented as shown in

Stratum-balanced allocations	Checked allocations	Fraction checked	Acceptable allocations	% acceptable of checked	Overall % acceptable
1.0113E18	100000	9.888E-14	2639	2.639%	0.001%

Table 6: Frequency of acceptable overall allocations.

_o1tert2	_o2tert2	_od1_2tert2	Frequency
14	15	-1	1331
15	14	1	1308

Table 7: Overall summary of a `d` covariate.

Table 7. Table 7 is quite similar to Table 5 but summarizes overall rather than within-stratum allocations. Of the 2,639 viable allocations found, 1,331 have 14 mid-sized practices in arm 1 and 15 in arm 2, while the other 1,308 have the reverse.

Restrictions on cluster coincidence

Covariate constraints may force a given pair of clusters to always or never be in the same arm; they may also yield a space of allocations where two clusters appear in the same arm quite often or quite rarely. Since this can indicate confounds between the arm assignments and one or more covariates (see below for a specific example), with possible implications for the design validity of the final randomization, **CCR** provides output to assess this possibility. (See Bailey and Rowley 1987, for a discussion of design validity in randomization schemes.)

The tables discussed in this section are based on the dataset `work._PAIRSTATS`, where each observation is a pair of clusters and the variables of interest are the number and proportion of satisfactory allocations in which the pair are in the same or different arms. (Appendix B contains descriptions of all the data sets created by **CCR**.)

Table 8 describes the range of how often practices (clusters) are together or apart across all possible pairs of practices. For a given pair of practices, `_samecount` is the number of allocations in which both practices are in the same arm, `_samefrac` is the proportion of allocations in which both practices are in the same arm, and `_diffcount` and `_diffrac` are analogous for allocations where the practices are in different arms. The table presents descriptive statistics for these variables across every pair of practices, and cases of full constraint, where `_samefrac` equals 0 or 1, are omitted to give a better sense of the range of partial constraints across the practices.

If the allocations were unconstrained, by chance we would expect the mean of `_samefrac` to be about 1/[# of arms]. Here, the distribution is quite close to that, with a given pair of practices appearing slightly more likely to be in different arms (in about 1,332 allocations, or 50.48%, on average) than in the same arm (in about 1,307 allocations, or 49.52%, on average).

Next, practices that are always constrained to the same or different arms are displayed. In this example, one pair of practices is always assigned to the same arm (see Table 9), and nine

Variable	Mean	Std Dev	Minimum	25th Pctl
_samecount	1306.8299	74.0494	854.0000	1295.0000
_samefrac	0.4952	0.0281	0.3236	0.4907
_diffcount	1332.1701	74.0494	947.0000	1302.0000
_difffrac	0.5048	0.0281	0.3588	0.4934

Variable	Median	75th Pctl	Maximum
_samecount	1317.0000	1337.0000	1692.0000
_samefrac	0.4991	0.5066	0.6412
_diffcount	1322.0000	1344.0000	1785.0000
_difffrac	0.5009	0.5093	0.6764

Table 8: Cluster coincidence descriptives.

cluster pair	% allocs in same arm
104 and 703	100.0%

Table 9: Pairs of clusters always allocated to the same arm.

cluster pair	% allocs in same arm
102 and 104	0.00%
102 and 703	0.00%
201 and 207	0.00%
204 and 206	0.00%
301 and 305	0.00%
503 and 504	0.00%
605 and 606	0.00%
802 and 1006	0.00%
907 and 908	0.00%

Table 10: Pairs of clusters always allocated to different arms.

pairs are always in different arms (see Table 10). If we were to examine the input data, we would find that in five of the ten strata (sites 1, 2, 3, 6, and 9), only two practices are in one level of a discrete covariate (for example, practices 102 and 104 are the only small site 1 practices, while practices 204 and 206 are the only majority-white site 2 practices), so those practices are forced into opposite arms to meet the constraint. Similarly, site 1 has one urban practice (104) and seven rural ones, site 7 has the reverse (703 is lone rural practice), and the other eight sites are all urban, so overall rural/urban balance forces practices 104 and 703 into the same arm.

Of potentially more concern, at site 5, ten of the 12 practices are large, while one (503) is small and one (504) is medium. In this stratum, evenly dividing both the large practices and the overall set of practices between arms forces the site's small and medium clusters into opposite arms, introducing a small confound between the size covariate (specifically the contrast between small and medium practices) and the study arms. Practice 802, the only medium practice at site 8, and practice 1006, the only large practice at site 10, are similarly forced into opposite arms by the overall balance constraints.

The potential impact of these confounds on the validity of the randomization should be considered, though the likely impact of confounds arising from isolated clusters lessens as the overall number of clusters increases. In STRIDE, it was decided that accepting these two slight confounds was preferable to the imbalances that would result from loosening or removing constraints. (In some studies, it might also be possible to address potential confounds by delaying randomization and adding additional clusters; this was not an option in STRIDE.)

Clusters can also appear together surprisingly often or rarely, with some arbitrariness in how we define "surprising." By default, **CCR** displays clusters appearing together less than half as often as one would expect by chance or more than 75% of the time, though these thresholds can be tuned with the `samearmlo` and `samearmhi` parameters.[8] Again, flagged pairs should be examined with an eye toward their impact on overall validity. No pairings in this randomization met either criterion (though some will in the next example).

If there are a small number of acceptable allocations and this section of output flags many pairs of clusters, it may be worth increasing the `osample` and/or `ssample` parameters to check whether the cluster pairs are generally constrained or the flags are artifacts of a small sample.

Final check

Once one acceptable allocation has been randomly selected, summary statistics are computed directly from the original data. Overall and in each stratum, the sums and means in each arm, and their differences and fractional differences, are displayed. Table 11 shows a section of the full table containing the checks on the total numbers of clusters and the sums for `rur`. The full table presents considerably more information than is needed to verify that the selected randomization satisfies the provided constraints, but since there is often some trial-and-error involved in selecting suitable constraints, the additional information may prove helpful in comparing the effects of different constraints or choosing constraints for a next iteration of testing.

[8] If the `binomsig` parameter is set, all pairs assigned to the same arm significantly more or less often than chance are also displayed. This option is turned off by default because with a large set of acceptable allocations, it produces long lists of small though statistically significant deviations from chance expectations. Probabilities are computed from the binomial distribution with $p = 1/[\#\text{ of arms}]$, and significance is tested with a two-tailed alpha of `binomsig`, Bonferroni-corrected for the number of pairs of clusters.

			rur			
			total			
	# clusters					fractional
	arm		arm		difference	difference
stratum	1	2	1	2	1-2	1-2
1	4	4	3	4	-1	-0.142857
2	4	4	0	0	0	0
3	5	6	0	0	0	0
4	5	4	0	0	0	0
5	6	6	0	0	0	0
6	4	4	0	0	0	0
7	4	4	1	0	1	1
8	3	2	0	0	0	0
9	4	4	0	0	0	0
10	4	5	0	0	0	0
	43	43	4	4	0	0

Table 11: Part of the final check table.

In addition to the displayed output, the dataset `work._FINAL_RANDOMIZATION_1` contains the merge of the input dataset and the arm assignments.

4.2. Example 2: Hypothetical randomization with a continuous covariate

Since the actual randomization of the STRIDE trial involved only discrete covariates, we turn to a hypothetical example to illustrate how **CCR** reports acceptable allocations involving continuous covariates.

Suppose that in the STRIDE trial, we had chosen to balance on actual practice sizes rather than tertiles, requiring that within each healthcare system, the mean practice size in each arm should be within 15% of the mean across the entire system, while overall, the difference between the means of each arm should be no more than 200. The macro call to do this would be:

```
%ccr(stride, site, practice,
     rur urb wht nwht size,
     d   d   d   d    c,
     s1  s1  s1  s1   mf.15,
     s1  s1  s1  s1   m200);
```

The full output for this example comprises the second half of `code.pdf` in the supplementary material. Since much of the output for this example is similar to that for the previous one, our discussion will only touch on the portions of the output that are qualitatively new.

Stratum-level allocation summaries

Table 12 describes the differences in `size`, the number of eligible patients at each practice,

Analysis Variable : _sad1_2size						
site	N Obs	Minimum	25th Pctl	Median	75th Pctl	Maximum
1	32	0.0062943	0.0322581	0.0555993	0.0923158	0.1201154
2	30	0.0242551	0.0577931	0.0961222	0.1272646	0.1482258
3	254	0.000042616	0.0295754	0.0618356	0.1022354	0.1485588
4	76	0.0028997	0.0362465	0.0823701	0.1123641	0.1422674
5	158	0.0020544	0.0388549	0.0752981	0.1117413	0.1463981
6	24	0.0094662	0.0407265	0.0675839	0.1014860	0.1459549
7	48	0.0091912	0.0386029	0.0606618	0.1047794	0.1305147
8	12	0.0386786	0.0453364	0.0737112	0.1385454	0.1452032
9	32	0.0099128	0.0567424	0.0875064	0.1172449	0.1384379
10	124	0.0035104	0.0404813	0.0687510	0.1164398	0.1493776

Table 12: Stratum-level summary of a `c` covariate.

Analysis Variable : _oad1_2size				
Minimum	25th Pctl	Median	75th Pctl	Maximum
0.0232558	10.4883721	22.4418605	37.3720930	103.4186047

Table 13: Overall summary of a `c` covariate.

between arms of the study. The presented distributions are based on the absolute values of the differences, and since the stratum constraint in this example is fractional, the differences in the table are expressed as fractions of the overall mean. For instance, the mean differences at site 8 range from 3.87% to 14.52% of the overall mean, with a median difference of 7.37%. Note that at sites 1, 7, and 9, the maximum differences do not approach the limit of 15%, so the size constraint does not particularly restrict allocations in those strata.

Overall allocation summaries

Since the overall constraint on `size` is absolute rather than fractional, Table 13 simply shows the distribution of the difference in overall means between arms. Some allocations are very close to balanced overall (the minimum difference is about .02), while the maximum difference of about 103.4 is well below the acceptable limit of 200.

Restrictions on cluster coincidence

As one might expect, the absolute constraints arising from the size tertiles in the actual STRIDE randomization are replaced by an assortment of absolute and partial constraints. Tables 14 and 15 show practices that are often (in at least 75% of acceptable randomizations) or rarely (in at most 25% of acceptable randomizations) randomized to the same arm.

cluster pair	% allocs in same arm	# allocs in same arm	# allocs in different arms
404 and 409	76.64%	9635	2936
601 and 602	75.13%	9444	3127
902 and 908	80.90%	10170	2401

Table 14: Pairs of clusters randomized to the same arm at least 75% of the time.

cluster pair	% allocs in same arm	# allocs in same arm	# allocs in different arms
301 and 305	9.79%	1231	11340
402 and 404	7.84%	986	11585
402 and 409	15.51%	1950	10621
602 and 603	24.87%	3127	9444
802 and 804	16.51%	2075	10496
803 and 805	17.18%	2160	10411
902 and 903	24.86%	3125	9446
902 and 904	18.39%	2312	10259
1002 and 1006	19.27%	2423	10148

Table 15: Pairs of clusters randomized to the same arm at most 25% of the time.

5. Discussion

CCR allows investigators to perform covariate-constrained randomization across a wide variety of possible designs, including those with more than two arms, with unequal numbers of clusters within strata, with constraints on counts or totals rather than means of covariates, and with allocation spaces too large to search exhaustively. However, as Ivers *et al.* (2012) note, covariate-constrained randomization requires more statistical know-how than some other randomization techniques, and **CCR** does not alleviate this situation; rather, its purpose is to allow a skilled practitioner to apply the technique to a greater assortment of studies. As discussed in Sections 3 and 4, there are a number of judgment calls that a user must make in the course of using the macro and evaluating its output (and perhaps some amount of trial and error in tuning the tightness of the covariate constraints to produce viable allocations while minimizing the potential for confounding the study arms with the covariates), but before running **CCR** at all, a prospective user must decide that covariate-constrained randomization is the best approach and decide how loose each constraint can be before it no longer serves the needs of the study. These decisions must be made in the context of a particular study, may not be simple, and are beyond the scope of this paper.

CCR is released under the terms of the GNU Lesser General Public License (included with

the distribution). The author's most recent version can be found at https://github.com/ejgreene/ccr-sas.

Acknowledgments

This publication was made possible by CTSA Grant Number UL1 RR024139 from the National Center for Research Resources (NCRR) and the National Center for Advancing Translational Science (NCATS), components of the National Institutes of Health (NIH), and NIH Roadmap for Medical Research; and Grant Number U01 AG048270 from the Patient-Centered Outcomes Research Institute (PCORI) and the National Institute on Aging (NIA) at NIH. Its contents are solely the responsibility of the author and do not necessarily represent the official view of NIA, NIH, or PCORI.

The author wishes to thank Denise Esserman, Peter Peduzzi, and three anonymous reviewers for helpful feedback on earlier drafts of this paper.

Example output for this paper was generated using SAS/STAT software, Version 13.2 for Windows (SAS Institute Inc. 2014b). SAS, SAS/STAT, and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.

References

Bailey RA, Rowley CA (1987). "Valid Randomization." *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **410**(1838), 105–124. doi:10.1098/rspa.1987.0030.

Chaudhary MA, Moulton LH (2006). "A SAS Macro for Constrained Randomization of Group-Randomized Designs." *Computer Methods and Programs in Biomedicine*, **83**(3), 205–210. doi:10.1016/j.cmpb.2006.04.011.

IEEE Computer Society, Microprocessor Standards Committee, Engineers IE (2008). *IEEE Standard for Floating-Point Arithmetic*. Institute of Electrical and Electronics Engineers, New York.

Ivers NM, Halperin IJ, Barnsley J, Grimshaw JM, Shah BR, Tu K, Upshur R, Zwarenstein M (2012). "Allocation Techniques for Balance at Baseline in Cluster Randomized Trials: A Methodological Review." *Trials*, **13**(1), 120. doi:10.1186/1745-6215-13-120.

Moulton LH (2004). "Covariate-Based Constrained Randomization of Group-Randomized Trials." *Clinical Trials*, **1**(3), 297–305. doi:10.1191/1740774504cn024oa.

Nietert PJ, Jenkins RG, Nemeth LS, Ornstein SM (2009). "An Application of a Modified Constrained Randomization Process to a Practice-Based Cluster Randomized Trial to Improve Colorectal Cancer Screening." *Contemporary Clinical Trials*, **30**(2), 129–132. doi:10.1016/j.cct.2008.10.002.

SAS Institute Inc (2014a). *SAS 9.4 Statements: Reference*. 3rd edition. SAS Institute Inc., Cary.

SAS Institute Inc (2014b). *SAS/STAT Software, Version 13.2*. Cary. URL http://www.sas.com/.

A. List of optional macro parameters

Note that the SAS macro language treats 0 as false and other numbers as true when evaluating logical expressions.

Parameter	Default value	Meaning
`arms`	2	Number of study arms.
`ssample`	100,000	– If nonzero, the maximum number of within-stratum allocations to test in each stratum. – If zero, all within-stratum allocations are tested.
`osample`	100,000	– If nonzero, the maximum number of overall allocations to test. – If zero, all overall allocations are tested.
`sizecheck`	1 (true)	– If true (not zero), the numbers of possible within-stratum and overall allocations are calculated, and all allocations are tested if the number possible is smaller than `ssample` (for within-stratum allocations) or `osample` (for overall allocations). – If false (zero), the numbers of possible allocations are not calculated, and sampling occurs per the values of `ssample` and `osample`.
`select`	1	How many final randomizations to generate.
`seed`	*undefined*	– If defined, the seed to use when sampling allocations and choosing the final randomization; if multiple randomizations are to be generated, the seed is incremented by 1 for each call to **proc surveyselect**. – If undefined, a seed is generated from the system clock (the **rand** function's and **proc surveyselect**'s default behavior).
`armsize`	*undefined*	– If defined, the variable(s) specifying how many clusters in a stratum to allocate to each arm; the first variable will specify the number of clusters to allocate to arm 1, the second arm 2, and so forth; [# of arms] − 1 variables are needed. – If undefined, clusters are split as evenly as possible across arms at both the within-stratum and overall levels.
`samearmhi`	75	Clusters assigned to the same arm as least this often (taken as a percentage) are displayed.
`samearmlo`	*undefined* (use half chance)	– If defined, clusters assigned to the same arm at most this often (taken as a percentage) are displayed. – If undefined, the display threshold is set to half chance ($\frac{100}{2 \cdot [\#\text{ of arms}]}$).

Parameter	Default value	Meaning
`binomsig`	0 (false)	– If true (not zero), pairs of clusters assigned to the same arm significantly more or less often than chance (with a two-tailed alpha of `binomsig`, Bonferroni-corrected for the number of cluster pairs) are displayed (in addition to the default checks). – If false (zero), only the default checks (always together, never together, together above `samearmhi`, and together below `samearmlo`) are displayed.
`verbose`	0	– If 0, only the summaries and checks described in Sections 2 and 4 are displayed. – If 1, the allocations that satisfy the constraints and the final randomization are also listed. – If 2, all possible/sampled within-stratum allocations are also listed. – If 3, all possible/sampled overall allocations are also listed (as in **CCRA** V1.0).
`debug`	0 (false)	If true (not zero), assorted macro variables are printed to the log as they are defined.

B. List of intermediate and output data sets

These data sets (listed in order of their creation) are available in the `work` library after execution.

`_D` is the original data set, with extraneous variables removed.

`_SCLUSTLIST` lists the clusters within each stratum.

`_SALLOCLIST` contains the number of possible and checked allocations within each stratum.

`_DS`*s* contains the stratum-level allocations to be checked for stratum *s*.

`_PARTIAL_S`*s*`_A`*n* contains stratum-level allocations for the first *n* arms of stratum *s* if all possible allocations were generated (or all arms, if $n = [\#\text{ of arms}] - 1$).

`_ASSIGN` contains all the stratum-level allocations to be checked.

`_ARMSIZES` contains the number of clusters in each arm of each stratum-level allocation.

`_ALLGROUPS` contains the stratum-level allocations with covariates merged in.

`_STATS_`*n* contains the sums and means of each covariate in arm *n* in each stratum-level allocation.

`_STATS_ALL` contains the overall sums and means of each covariate in each stratum-level allocation.

_STATS contains the full stratum-level statistics (covariate in-arm and overall sums and means and between-arm differences) for allocations meeting the stratum-level constraints.

_STRATUM_SURVIVORS contains the number of satisfactory stratum-level allocations for each stratum.

_CANDIDATES contains the space of overall allocations (combinations of stratum-level allocations) that will be checked.

_RSAMPLE merges the stratum-level allocation stats with the overall allocation list.

_RSAMP4STATS also contains the number of clusters in each arm from each stratum.

_DSTATS contains overall sums and means of each covariate in each overall allocation.

_OK contains the full overall statistics (covariate in-arm and overall sums and means and between-arm differences) for allocations meeting the overall constraints.

_VETTED also contains the stratum-level statistics.

_OSAMPLEPROPS is a condensed version of _OK, containing only the sums, means, and differences needed to report on the overall distribution of covariates across the acceptable overall allocations (with one observation per allocation rather than one per allocation/stratum combination).

_NVALALLOC contains the number of overall allocations checked and the number and percentage that are satisfactory.

_GROUPSARMn contains every cluster ever allocated to arm n (along with which stratum it's in).

_ALLOCSARMn contains the same information, but transposed with one observation per stratum.

_ARMn contains every pair of clusters appearing together in arm n (with one observation per appearance).

_OUTGROUPn contains every pair of clusters appearing together in arm n and the number of allocations they appear together in.

_ALLOUTGROUP is the concatenation of all _OUTGROUPn datasets.

_ALLPAIRS contains all possible pairs of clusters.

_SEENPAIRS contains all possible pairs of clusters and the number of allocations (possibly 0) in which they appear together in any arm.

_PAIRSTATS contains the number and proportion of allocations in which each pair of clusters appear in the same or in different arms and the probability (based on a binomial distribution) that difference between 1/[# of arms] and the actual proportion is due to chance.

FINALSAMPLEn contains the arm assignments for the nth selected allocation.

`_FINAL_RANDOMIZATION_n` contains the original data plus the arm assignments from the nth selected allocation.

Affiliation:

Erich J. Greene
Yale Center for Analytical Sciences
Yale School of Public Health
Yale University
300 George St., Suite 555
New Haven, CT 06511, United States of America
E-mail: erich.greene@yale.edu
URL: http://ycas.yale.edu/working/people/erich_greene.profile

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
April 2017, Volume 77, Code Snippet 1 *Submitted:* 2015-06-23
doi:10.18637/jss.v077.c01 *Accepted:* 2016-01-11