The Stata Journal (*yyyy*) *vv*, Number *ii*, pp. 1–23

cvcrand and cptest: Efficient design and analysis of cluster randomized trials

John A. Gallis
Duke University Department of Biostatistics
Duke Global Health Institute
Durham, NC
john.gallis@duke.edu

Fan Li
Duke University Department of Biostatistics
Durham, NC
frank.li@duke.edu

Hengshi Yu
University of Michigan Department of Biostatistics
Ann Arbor, MI
hengshi@umich.edu

Elizabeth L. Turner
Duke University Department of Biostatistics
Duke Global Health Institute
Durham, NC
liz.turner@duke.edu

Abstract. Cluster randomized trials (CRTs), where clusters (e.g., schools or clinics) are randomized to comparison arms but measurements are taken on individuals, are commonly used to evaluate interventions in public health, education, and the social sciences. Since CRTs typically involve a small number of clusters (e.g., less than 20), simple randomization frequently leads to baseline imbalance of cluster characteristics across study arms, threatening the internal validity of the trial. In CRTs with a small number of clusters, classic approaches to balancing baseline characteristics—such as matching and stratification—have several drawbacks, especially when the number of baseline characteristics the researcher desires to balance is large (Ivers et al. 2012). An alternative design approach is covariate constrained randomization, whereby a randomization scheme is randomly selected from a subset of all possible randomization schemes based on the value of a balancing criterion (Raab and Butcher 2001). Subsequently, a clustered permutation test can be used in the analysis, which provides increased power under constrained randomization compared to simple randomization (Li et al. 2015). We describe covariate constrained randomization and the permutation test for the design and analysis of CRTs, and provide an example to demonstrate the use of our newly-created Stata programs `cvcrand` and `cptest` to implement constrained randomization and the permutation test.

Keywords: st0001, `cvcrand`, `cptest`, covariate constrained randomization, cluster randomized trials, permutation test

1 Introduction

The cluster randomized trial (CRT), which randomizes clusters (e.g., schools or clinics) of individuals to intervention arms, is a study design used in many fields of research. The cluster randomization design is typically chosen for logistical reasons, when there is a high probability of treatment contamination across study arms, when the intervention is group-based, or when individual randomization is not feasible (Turner et al. 2017a). For example, in the Thinking Healthy Program Peer-Delivered Plus (THPP+) study, the researchers recruited depressed women in their third trimester of pregnancy from 40 villages in Pakistan, and each village was randomized to either the intervention or enhanced usual care (Sikander et al. 2015; Turner et al. 2016). Since this was a public health intervention delivered by community health workers, the risk of contamination across study arms would be too high if individual women were randomized, especially if a woman receiving intervention and a woman receiving enhanced usual care live close to each other.

There is a variety of cluster randomization designs described in the literature (Turner et al. 2017a). Here we focus on the most common one, the two-arm parallel design (e.g., intervention arm and control arm). In this design, a set of clusters is identified at the beginning of the study and each one is randomly assigned to one of the two intervention arms. Although the clusters are the units of randomization, outcomes are typically measured at the individual level. At the analysis stage, outcomes from both arms are compared to determine whether the intervention is effective, by accounting for correlation due to the clustered design.

A frequent practical limitation of cluster randomization designs is that a small number of clusters are randomized, mostly because of availability or resource constraints. Fiero et al. (2016) found that of the 86 studies included in their review of CRTs, about 50% randomized 24 or fewer clusters. In CRTs related to cancer published between 2002 and 2006, Murray et al. (2008) found similar results, with about 50% randomizing 24 or fewer clusters. Additionally, in their review of 300 CRTs published between 2000 and 2008, Ivers et al. (2011) found that, of the 285 studies reporting number of clusters randomized, at least 50% randomized 21 or fewer clusters.

When a small number of clusters are randomized, cluster characteristics that are expected to be predictive of the outcome ("prognostic" covariates) could be unevenly distributed across arms under simple randomization. The chance of such baseline covariate imbalance increases as the number of available clusters decreases and as the number of predictive covariates increases. Implications of baseline imbalance include lack of internal validity of the trial, reduced statistical power, insufficient precision of effect estimates, and the possibility that additional statistical adjustment will be needed in the analysis phase, which may make the analysis more challenging (Ivers et al. 2012). All of these concerns could threaten the face validity of the trial.

To address these issues, several restricted randomization procedures, such as stratification, matching, and minimization, have been proposed to help achieve balance on important baseline covariates. When the total number of clusters is small, stratification and matching have several limitations. Specifically, stratified randomization is only feasible if the number of stratification variables is small. With more than a few stratification covariates, there is a risk of creating strata with only a single cluster, which may lead to unequal allocation of clusters as well as imbalance on the very variables stratification was intended to balance (Ivers et al. 2012). Ivers et al. (2012) recommend that the maximum number of strata should be limited to about $\frac{1}{4}$ to $\frac{1}{2}$ of the total number of clusters, and in CRTs with a small number of clusters this is only possible with no more than a few stratification variables. Matching, on the other hand, may suffer from severe power loss when one cluster is lost to follow-up because its match will be removed from the matched analysis (Ivers et al. 2012). Loss to follow-up can occur, for example, if a cluster which initially gave consent to participate in the trial withdraws consent before or during the follow-up phase. Further, matching may not be effective when the matching characteristics are poorly correlated with the outcome, and the subsequent matched analysis may lose power (Diehr et al. 1995). Matched clusters will also make it challenging to properly calculate the intracluster correlation coefficient, a measure of clustering that is recommended to be reported in all cluster randomized trials (Donner and Klar 2004; Klar and Donner 1997; Campbell et al. 2012). In addition, there is debate on how best to analyze matched trials (Diehr et al. 1995). Minimization could be used when clusters are recruited sequentially over time, but may have limited application when all clusters are recruited at the beginning of the trial, which is the setting of interest in the current paper. Therefore, given the limitations of stratification and matching for this setting, alternative strategies for restricted randomization are needed, especially when only a few clusters are randomized or a number of baseline covariates need to be balanced.

1.1 Covariate constrained randomization

An alternative form of restricted randomization that can be used to achieve baseline covariate balance in CRTs with all clusters enrolled before randomization is covariate constrained randomization (sometimes referred to simply as "constrained randomization"). Under simple randomization, a randomization scheme (i.e., a unique allocation of clusters to study arms) is randomly chosen from the space of all $\binom{k}{g}$ randomization schemes, where k is the total number of clusters and g is the number of clusters assigned to one study arm. For example, if we design a CRT with 12 clusters, 6 of which are assigned to intervention and 6 to control, there are $\binom{12}{6} = 924$ unique allocations of 12 clusters evenly assigned to two arms.

In a review of 300 randomly selected CRTs published between 2000 and 2008, approximately half randomized 21 or fewer clusters, but 44% of the 300 did not use any form of restricted randomization in the trial design, even though the probability of baseline covariate imbalance is not small (Ivers et al. 2011, 2012). Of the 56% that used some form of restricted randomization, most (57%) used stratification; very few used covariate constrained randomization. One reason for the infrequent use of covariate constrained randomization versus stratification or matching may be that practitioners find it challenging to implement. Therefore, to address this potential barrier we have created a user-friendly, easy-to-implement Stata package `cvcrand` to perform covariate constrained randomization for the design of CRTs and to implement an appropriate method in the analysis phase. Before introducing the program commands, we briefly review key features of the approach including the choice of balance metrics.

When applying covariate constrained randomization, a randomization scheme is randomly selected from a subset of all possible schemes based on the value of a pre-specified balance metric (Raab and Butcher 2001; Moulton 2004; Carter and Hood 2008; de Hoop et al. 2012; Li et al. 2015). A full description of covariate constrained randomization designs is provided in Li et al. (2017). In brief, to carry out a covariate constrained randomization design, a researcher will (i) specify important cluster-level covariates; (ii) either enumerate all or simulate a large number of potential randomization schemes; (iii) remove the duplicate randomization schemes if any; (iv) choose a constrained space containing a subset of schemes where sufficient balance across covariates is achieved according to some pre-specified balance metric; and (v) randomly sample one randomization scheme from this constrained space. This randomly sampled scheme will be used to assign clusters to study arms. Note that cluster-level data supplied for constrained randomization may also be aggregated from individual-level data. In practice, however, it is not always possible to obtain individual-level data at the design phase.

In principle, any sensible method for creating a balance metric may be selected. Here we describe two commonly used metrics. First, the $l2$ balance metric was proposed by Raab and Butcher (2001) and studied by Li et al. (2015) and Li et al. (2017). Following the notation of Li et al. (2017), for a given randomization scheme this balance metric is defined as

$$B_{(l2)} = \sum_{j=1}^{n} \omega_j (\bar{x}_{Tj} - \bar{x}_{Cj})^2, \tag{1}$$

where n is the total number of variables to balance, ω_j is a variable-specific weight, $\bar{x}_{Tj}$ is the average of the j^{th} variable in the intervention clusters, and $\bar{x}_{Cj}$ is the average of the j^{th} variable in the control clusters. Using the balance metric, balance scores can be computed for every potential randomization scheme. The $l2$ balance metric is defined for both continuous and binary variables (including $p-1$ dummy variables created from a p-level categorical variable). For binary variables, the mean is simply the proportion with level "1" of the variable. The weights are often chosen to be the inverse of the standard deviation of the j^{th} cluster-level covariate across the two intervention arms. Thus, any number of continuous or categorical variables can be included to compute the balance score using this balance metric. In practice, we recommend to only include variables which are hypothesized to be correlated with the outcome. If the number of variables to be constrained on is quite large relative to the number of clusters then when these variables are accounted for in the analysis stage, a subset may need to be selected to avoid overfitting. This subset should include the variables that are identified *a priori* to be the most predictive of the outcome, based on expert knowledge (Li et al. 2017).

Researchers may choose to assign larger weights to cluster characteristics considered "more important" than others. For example, suppose at the design stage that the researchers have variables they consider more important to balance than others. This can be accomplished by specifying a larger weight on the these variables. Such user-defined weights are distinct from the inverse standard deviation weights ω_j in equation (1). We can add a weight to equation (1) by

$$B_{(l2)} = \sum_{j=1}^{n} d_j \omega_j (\bar{x}_{Tj} - \bar{x}_{Cj})^2, \tag{2}$$

where d_j is the user-defined weight for variable j. If not specified, d_j defaults to 1 for each variable.

A second, alternative metric, is the $l1$ balance metric, in which the square in equation (1) is replaced with an absolute value:

$$B_{(l1)} = \sum_{j=1}^{n} \omega_j |\bar{x}_{Tj} - \bar{x}_{Cj}|. \tag{3}$$

User-defined weights can be added to this equation in a similar manner to equation (2). It can be seen from equations (1) and (3) that the smaller the value of the balance score, the more balanced the n selected baseline cluster-level covariates will be between the two intervention arms.

1.2 Clustered permutation tests

After performing covariate constrained randomization to balance cluster-level characteristics in the design of a CRT, an appropriate analysis technique should be selected to analyze the data collected during the implementation phase of the CRT. Using a simulation study, Li et al. (2015) provide evidence that even after balancing baseline cluster-level covariates in the design stage, analysis-based adjustment for prognostic covariates is necessary. Two prominent options include mixed model F-tests and clustered permutation tests (Li et al. 2015; Turner et al. 2017b). Li et al. (2015) showed that under covariate constrained randomization, adjusted clustered permutation tests provide increased power under constrained randomization compared to simple randomization. Clustered permutation tests are carried out in the constrained space, and have the desirable property that they preserve the nominal type I error rate even for very small CRTs, as long as the design is not overly constrained (Li et al. 2015). In a later paper, Li et al. (2017) show that only a subset of prognostic variables that are balanced for in the constrained randomization design must be adjusted for in the analysis, although in practice the researcher may wish to include all covariates on which the design was balanced. In addition, individual-level covariates may be included in the analysis to increase the precision of the test (Li et al. 2017).

To perform a valid clustered permutation test using data collected in a CRT, the outcome data analyzed must be at the individual level, to avoid loss of information from aggregating up to the cluster level. In a clustered permutation test, the individual-level data are first analyzed using a regression that omits intervention arm as a variable. The regression method (e.g., linear or logistic) depends on the distribution of the outcome. From this regression, the residuals are obtained (e.g., on the logit scale for logistic regression), and then the cluster-level average residual is computed for each cluster. From these residuals, we calculate the observed test statistic by multiplying this vector of residuals by the vector of the selected scheme with -1 substituted for 0, then taking the absolute value. For example, suppose that the final randomization scheme for a trial with six clusters is given by

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix},$$

where 1 is assignment to intervention and 0 is assignment to control, and the average cluster-level residuals are

$$\begin{pmatrix} .84 \\ .54 \\ -.19 \\ -.22 \\ .43 \\ -.32 \end{pmatrix}.$$

Thus, the observed test statistic is

$$\begin{pmatrix} 1 & 1 & -1 & -1 & 1 & -1 \end{pmatrix} \begin{pmatrix} .84 \\ .54 \\ -.19 \\ -.22 \\ .43 \\ -.32 \end{pmatrix} = |2.54| = 2.54.$$

Next, we calculate the null "permutational distribution" by computing the value of the test statistic under all other possible randomization schemes in the randomization space. Under simple randomization, this space consists of all $\binom{k}{g}$ randomization schemes; under constrained randomization, the space includes only those randomization schemes where the balance score is below the cutoff (i.e., the constrained space from which the final randomization scheme was chosen). The observed test statistic is referenced against this permutational distribution to obtain a p-value for the intervention effect that accounts for both the clustered design of the CRT and the covariate constrained randomization used in selecting the final randomization scheme. This p-value is obtained by computing the percentage of times test statistics corresponding to other randomization schemes in the constrained space are greater than the test statistic corresponding to the randomization scheme used to assign clusters to intervention arms. For an adjusted permutation test, we simply control for the relevant cluster- and individual-level covariates in the regression model and use those residuals to obtain an adjusted test statistic. A sufficient condition under which the permutation test is valid is that an equal number of clusters are assigned to each arm (Gail et al. 1996). This means that if the number of clusters randomized to intervention is not the same as the number randomized to control, the test may be anti-conservative (i.e., the type I error may be larger than the nominal level). See Gail et al. (1996) for more technical details. Our program `cptest` implements the permutation test both in its unadjusted and adjusted form. The steps are illustrated in the example in section 4.

2 The cvcrand command

In this section, we introduce the `cvcrand` program, explaining the available options in detail and “going under the hood” to examine the inner workings of the program. The program implements covariate constrained randomization, and can handle a variety of situations. The program requires that the user provides a data set where each row of data corresponds to one cluster and the columns contain information on characteristics of the clusters. Those characteristics can be either cluster-level characteristics or individual-level data aggregated to the cluster level (e.g., percentage of cluster that is female). All continuous variables must be numeric, but categorical variables can be either of numeric or string type. Categorical variables should be supplied to the `categorical` option, as they will be converted to dummy variables for the program. The results of the program are sensitive to which level of a multi-categorical variable is removed. The user may wish to recode some categorical variables before running `cvcrand` in order to set which level of the categorical variable will be removed after transformation to dummy variables.

The `cvcrand` command requires that `ntotal_cluster` and `ntrt_cluster` be specified by the user. The total number of clusters (`ntotal_cluster`) specified must equal the number of rows in the data set. The number of clusters in the treatment (intervention) arm (`ntrt_cluster`) must be less than the total number of clusters. The program can be used whether an equal number or an unequal number of clusters are assigned to each study arm. In addition, the program can handle an odd total number of clusters. However, if the number assigned to each study arm is unequal, then when analyzing the final data, the clustered permutation test may be anti-conservative, as mentioned in section 1.2.

To avoid prohibitive computations associated with matrices with extremely high dimensions, the program will automatically simulate 50,000 randomization schemes if the simple randomization space contains more than 50,000 schemes. This can be overridden by the user by specifying the `nosim` option. The program allows the user to implement one of the two balance metrics mentioned in section 1.1, the $l1$ and $l2$, with $l2$ being the default.

The default cutoff of the balance score below which a randomization scheme is selected is 0.1, but this can be modified using the `cutoff` option. Simulations have shown that a cutoff of 10% works well in some scenarios (number of clusters k=16 and 26) (Li et al. 2017). Ideally, this number should be small, but not too small. In practice, if the number of clusters is even less than 10, the researcher may wish to choose a larger value of the cutoff, in order to avoid overly constraining the design, in which case there would be few randomization schemes in the constrained space.

All of the randomization schemes that make up the selected constrained space can be saved to a Stata data set by specifying the `savedata` option, and we strongly recommend that this option be selected. The data set obtained from this command is required in order to be able to implement the clustered permutation test in the analysis. In addition, the user may specify the `savebscores` option to save the column vector of balance scores to a Stata data set and produce a histogram of the balance scores.

Inside the program, the user-supplied data are passed to a `mata` program. As noted above, if the total number of randomization schemes is less than or equal to 50,000, the entire randomization space is enumerated; otherwise, the program simulates 50,000 randomization schemes, of which the unique schemes are kept. To create a vector of balance scores corresponding to equation (1), we first create a cluster-level design matrix, with each row indicating a cluster and each column corresponding to a variable in `varlist`. This cluster-level design matrix is standardized such that each element is centered by the column-specific (that is, variable-specific) mean and scaled by the column-specific standard deviation. In other words, each column has zero mean and unit variance. The matrix of unique randomization schemes (with a value of 0 corresponding to control and 1 corresponding to intervention) is then multiplied by the standardized design matrix to obtain a new matrix. The row sums of squared elements from this new matrix are proportional to the $l2$ balance metric and these computed balance scores will be used to rank the balance of each randomization scheme. A subset of randomization schemes is obtained by applying the pre-specified cutoff value to the set of balance scores, and a final randomization scheme is sampled from this subset. A similar algorithm is used to implement constrained randomization with the $l1$ balance metric corresponding to equation (3). A complete description of this algorithm is available in Li et al. (2017).

2.1 Syntax

```
cvcrand varlist, ntotal_cluster(#) ntrt_cluster(#) [
   clustername(varname) categorical(varlist) balancemetric(string)
   cutoff(#) numschemes(#) nosim size(#) weights(numlist)
   stratify(varlist) seed(#) directory(string) savedata(string)
   savebscores(string) ]
```

2.2 Options

Required

`ntotal_cluster(`*#*`)` specifies the total number of clusters to be randomized. This value must be a positive integer and must be equal to the number of rows in the data set.

`ntrt_cluster(`*#*`)` specifies the number of clusters that the researcher desires to assign to the treatment (intervention) arm. It must be a positive integer less than the total number of clusters. Often, this is equal to half the number of total clusters.

Optional

`clustername(`*varname*`)` specifies the name of the variable that is the identification variable of the cluster. This is used when the program summarizes the variables after constrained randomization. If no cluster identification variable is specified, the default is to label the clusters by the order they appear in the data set (i.e., 1, 2, 3,...).

`categorical(`*varlist*`)` specifies categorical variables. Each categorical variable will be turned into $p-1$ dummy variables, where p is the number of levels of the categorical variable. Note that the results are sensitive to which level is excluded. Categorical variables may be recoded to specify which level to exclude, by setting this level to be the lowest number or earliest in the alphabet. If the weights option is used, then all categorical variables must be specified last in *varlist* in order for the program to work correctly.

`balancemetric(`*string*`)` specifies the balance metric to use. The default is the $l2$ balance metric. The $l1$ metric may be specified instead, if desired.

`cutoff(`*#*`)` specifies the percentile cutoff of the distribution of the balance score below which we randomly sample the final randomization scheme. The value will range between 0 and 1. The default is 0.1 (that is, 10%). A smaller balance score indicates better balance based on our balancing criterion. Therefore, we are "constraining" the randomization space and only sampling from the set of randomization schemes corresponding to the "best" values of balance score. The cutoff can be overridden by the `numschemes` option.

`numschemes(`*#*`)` specifies the number of randomization schemes to form the constrained space from which the final randomization scheme is selected. This overrides the cutoff option. If this option is specified, the program will randomly sample the final randomization scheme from the randomization schemes corresponding to the S smallest balance scores, as in `numschemes(S)`.

`nosim` overrides the program's default procedure of simulating when the number of randomization schemes is over 50,000, and will instead enumerate all randomization schemes, regardless of the size of the randomization space. Note: this can consume a lot of memory and may cause Stata to crash. For example, with 30 clusters and 15 assigned to treatment, the total randomization space is a $\binom{30}{15} = 155,117,520$ row by 30 column matrix.

`size(`#`)` specifies the number of randomization schemes to simulate if the size of the simple randomization space is greater than 50,000 unique schemes (as happens when, for example, there are 20 clusters and 10 assigned to intervention: $\binom{20}{10} = 184,756$). The default is to simulate 50,000 schemes. Simulation can be overridden by the `nosim` option.

`weights(`*numlist*`)` allows the specification of user-defined weights. These are distinct from the inverse standard deviation ω_j weights in equations (1) and (3). Instead, these user-defined weights correspond to d_j in equation (2). Note that these weights could be used to induce stratification on variables. For instance, if one variable is given a large weight, say 1000, and all other variables are given a weight of 1, the randomization scheme chosen will be stratified by the variable with the large weight, assuming a reasonably low cutoff value has been chosen. Stratification is directly implemented by the `stratify` option. The `weights` option cannot be specified at the same time as the `stratify` option. See section 4.3 for more details.

Weights must be replicated for categorical variables (e.g., a three-category variable must be given two weights, one for each dummy variable), and categorical variables should be specified last in *varlist* if `weights` is specified.

`stratify(`*varlist*`)` specifies variables the user wishes to stratify on. These variables must be categorical variables and placed last in the overall *varlist*. Variables specified in `stratify(`*varlist*`)` will be assigned an arbitrarily large weight of 1000, which will induce stratification on these variables (if possible). Stratification will not be possible, for example, if one of the levels of a categorical variable contains an odd number of clusters. See section 4.3 for more details. This option cannot used with the `weights` option.

`seed(`#`)` specifies the seed for simulation and random sampling, which is needed so that the randomization can be replicated, if desired. The default seed is 12345.

`directory(`*string*`)` is the directory in which the constrained randomization space and balance scores are saved. The default is to save them in the current working directory.

`savedata(`*string*`)` saves the constrained randomization space into a Stata data set specified by string. The data set will be saved into the current working directory, and will also contain an indicator variable specifying which row of the constrained space was chosen as the final randomization space. The constrained randomization space will be needed for the analysis once the CRT is completed.

`savebscores(`*string*`)` saves the vector of all balance scores (across the entire randomization space) as a Stata data set specified by *string*. When this option is specified, a histogram is also produced which displays the distribution of all balance scores with a red line on the graph indicating the selected cutoff. The histogram will not be automatically produced when either the *stratify* or *weights* option is specified.

3 The cptest command

In this section, we introduce the `cptest` program, which is used to implement a clustered permutation test. The program requires the user to provide a list of variables that will be passed to a regression procedure. Therefore, the first variable in *varlist* must be the outcome variable, and all variables following are the independent variables in the regression model. Outcome data must be at the individual level, not the cluster level. The independent variables passed to the regression procedures should not include the intervention assignment variable.

The user must indicate which variable identifies clusters using the option `clustername`. This cluster identification variable must be the same across all individuals in the same cluster. The user must also specify the name of the Stata data set containing the constrained randomization space (saved in Stata format by the `cvcrand` program) using the `cspacedatname` option, along with the directory where this data set is stored using the `directory` option.

In the program, Stata takes the *varlist* provided by the user and runs a regression model with type of model determined by the required `outcometype` option. Residuals are obtained and then averaged by cluster. The vector of residuals and the permutation matrix are passed to `mata`, at which point Stata carries out the procedure described in section 1.2 to produce the p-value for the clustered permutation test.

It is possible to provide the program with only the outcome and no independent variables (an unadjusted permutation test) or only a subset of the variables constrained on in the design phase of the study. To achieve higher power, it is recommended to include all variables that are predictive of the outcome (Li et al. 2017). However, the permutation tests are robust to regression model misspecification and will maintain the nominal type I error even when some prognostic variables are left out (Gail et al. 1996). Note that this program could also be used to perform a clustered permutation test under simple randomization, by supplying the design matrix containing the simple randomization space.

3.1 Syntax

`cptest` *varlist*`, clustername(`*varname*`) directory(`*string*`) cspacedatname(`*string*`)`
`outcometype(`*#*`)` [`categorical(`*varlist*`)`]

3.2 Options

Required

`clustername(`*varname*`)` specifies the name of the variable that is the identification variable of the cluster.

`directory(`*string*`)` specifies the directory where the constrained randomization space (saved by program `cvcrand`) Stata data set is saved.

`cspacedatname(`*string*`)` gives the name of the data set containing the saved randomization space. This data set contains the permutation matrix, as well as a variable indicating which row of the permutation matrix was saved as the final scheme.

`outcometype(`*string*`)` specifies the type of regression model that should be run. Options are "continuous" for linear regression fit by Stata's `regress` command (suitable for continuous outcomes) and "binary" for logistic regression fit by Stata's `logit` command (suitable for binary outcomes).

Optional

`categorical(`*varlist*`)` specifies categorical variables. Categorical variables will be turned into $p-1$ dummy variables, where p is the number of levels of the categorical variable. The user must ensure that the same level of the categorical variable is excluded as was excluded when running `cvcrand`, by coding the variables the same way as in the design phase.

4 Example: Increasing up-to-date immunization rates

We now illustrate the use of `cvcrand` and `cptest` through an example. We use the data which are described and published in Dickinson et al. (2015). In this CRT, the researchers wished to compare two approaches (interventions) for increasing the "up-to-date" immunization rate in 19- to 35-month-old children. They planned to randomized 16 counties in Colorado in a 1:1 ratio to either a population-based reminder/recall approach or practice-based reminder/recall approach. These approaches are described in detail in Kempe et al. (2015).

4.1 Covariate constrained randomization

Prior to randomization, the researchers identified eight county-level variables potentially related to the outcome. For illustration, we will randomize by constraining on a subset of these variables. This subset contains the following five variables: % of children ages 19-35 months with $\geq$2 immunization records in the Colorado Immunization Information System (CIIS), estimated % of children already up-to-date on their immunizations, % Hispanic, location (urban/rural), and average income categorized into tertiles. Note that we could have left average income as a continuous variable, but chose to categorize it to illustrate the use of `cvcrand` on multi-category variables. Note also that we truncated the % in CIIS variable at 100%, as the value for one county published in Dickinson et al. (2015) exceeded 100%.

After loading this county-level data into Stata, we performed covariate constrained randomization with an equal number of counties in each intervention arm. The program `cvcrand` requires `mm_subsets()` from the `moremata` package, and the `table1` command from the `table1` package. The user will be prompted to download these if not already installed.

```
. cvcrand inciis uptodate hispanic location incomecat,
> categorical(location incomecat) ntotal_cluster(16) ntrt_cluster(8) clustername(county)
> seed(10125) cutoff(0.1) balancemetric(l2) savedata(dickinson_constrained)
> savebscores(dickinson_bscores)
  (output omitted)
```

We used the default balance metric ($l2$) and the default cutoff (0.1). We specified that two of the variables are categorical, and used the `savedata` option to save the constrained space as a Stata data set named "dickinson_constrained", which will be needed for later analysis. Selected output from the program is given below.

```
. cvcrand inciis uptodate hispanic location incomecat, ///
> categorical(location incomecat) ntotal_cluster(16) ntrt_cluster(8) clustername(county) ///
> seed(10125) cutoff(0.1) balancemetric(l2) savedata(dickinson_constrained) ///
> savebscores(dickinson_bscores)
  (output omitted)
Summary Stats   Balance Score

         Mean       24.00
    Std. Dev.       14.88
          Min        1.16
           p5        5.85
          p10        7.72
          p20       10.94
          p25       12.38
          p30       14.03
          p50       21.07
          p75       32.25
          p95       52.98
          Max       97.71

Cutoff value =      7.72

Value of selected balance score =      7.07

Row of constrained matrix =       903

        county   _allocation

  1.         1             0
  2.         2             1
  3.         3             0
  4.         4             1
  5.         5             0

  6.         6             0
  7.         7             0
  8.         8             1
  9.         9             0
 10.        10             1

 11.        11             1
 12.        12             1
 13.        13             0
 14.        14             0
 15.        15             1

 16.        16             1

  (output omitted)
```

First, the program provides summary statistics of the balance scores and the selected cutoff value. This is equal to p10 (10th percentile of the distribution) since we decided to use the default value. We are also given the value of the selected balance score, which is slightly below the 10th percentile of the balance score distribution. Any randomization scheme with a balance score of less than or equal to 7.72 could have been selected.

The program automatically saves a variable named "`_allocation`" (which contains the final selected randomization scheme) back onto the input data set, and in order to summarize balance across arms, it then summarizes or tabulates each variable by `_allocation`, using the `table1` command on each variable in `varlist` individually. However, we may run the user-written `table1` command on all variables in `varlist` together to easily summarize the results in one table

```
. table1, by(_allocation) ///
> vars(inci contn \ uptod contn \ hisp contn \ loc cat \ incomecat cat) ///
> format(%2.1f)
  --------------------------------------------------------------------------------
  Factor                    Level    _allocation = 0   _allocation = 1   p-value
  --------------------------------------------------------------------------------
  N                                  8                 8
  --------------------------------------------------------------------------------
  % in CIIS, mean (SD)               88.3 (5.8)        85.8 (8.8)           0.51
  --------------------------------------------------------------------------------
  % up-to-date, mean (SD)            40.4 (9.1)        41.3 (8.0)           0.84
  --------------------------------------------------------------------------------
  % Hispanic, mean (SD)              21.6 (14.8)       23.0 (11.7)          0.84
  --------------------------------------------------------------------------------
  Location                  Rural    5 (63%)           3 (38%)              0.32
                            Urban    3 (38%)           5 (63%)
  --------------------------------------------------------------------------------
  Average income            Low      3 (38%)           2 (25%)              0.82
                            Med      3 (38%)           3 (38%)
                            High     2 (25%)           3 (38%)
  --------------------------------------------------------------------------------
```

We see that all covariates are reasonably well-balanced between the intervention arms. Note, for example, that the binary variable location is not perfectly balanced between intervention arms, because 5 out of 8 rural counties are in one of the interventions compared to 3 out of 8 in the other intervention. If the researchers desired to stratify on this variable (to obtain perfect balance) while also constraining on other variables, this can be accomplished with user-defined weights. We will show an example of this functionality in section 4.3.

4.2 Clustered permutation test analysis

At the end of the study, the researchers will have ascertained the outcome in the 16 counties. As discussed in section 1.2, the outcome data must be at the individual-level. In the case of the Dickinson et al. (2015) data, the researchers will have up-to-date immunization information on a subset of children from the 16 counties. For the purposes of this example, we have created a simulated data set to illustrate how to use `cptest`. Suppose that `_allocation = 1` denotes the practice-based intervention, and that this intervention succeeds in improving up-to-date immunization rates much more than the community-based intervention. Suppose also that the researchers were able to assess 300 children in each cluster. We simulated correlated outcome data at the individual level using a generalized linear mixed model (GLMM) to induce correlation by including a random effect. The intracluster correlation (ICC) was set to be 0.01, using the latent response definition provided in Eldridge et al. (2009). This is a reasonable value of the ICC for population health studies (Hannan et al. 1994). We simulated one data set, with the outcome data dependent on the county-level covariates used in the constrained randomization design, and a positive intervention effect so that the practice-based intervention increases up-to-date immunization rates more than the community-based intervention. Summarizing the data, we find that about 86% of the children in the practice-based intervention (`_allocation` = 1) are up-to-date on immunization at the end of the study, while 79% of the children in the community-based intervention (`_allocation` = 0) are up-to-date.

```
. tab _allocation, summarize(outcome)

            |         Summary of outcome
_allocation |       Mean   Std. Dev.       Freq.
------------+------------------------------------
          0 |   .78916667   .40798529       2,400
          1 |   .85958333   .34749121       2,400
------------+------------------------------------
      Total |     .824375   .38054044       4,800
```

After loading these data into Stata, we run an adjusted clustered permutation test by including the outcome variable (`outcome`) followed by all the variables we used in the constrained randomization in *varlist*. We specify where the constrained randomization space data set is located, and use `outcometype(Binary)` to indicate that we would like to perform logistic regression. In the analysis, one should ensure that for a p-category variable, the same level of the variable is excluded when converting into the $p-1$ dummy variables as was removed when `cvcrand` was run.

```
. cptest outcome inciis uptodate hispanic location incomecat, ///
>  clustername(county) directory(P:\Program\Stata Journal) ///
>  cspacedatname(dickinson_constrained) ///
>  outcometype(Binary) categorical(location incomecat)

Logistic regression was performed
Final chosen scheme used by the cptest program:

     +---+
   1 | 0 |
   2 | 1 |
   3 | 0 |
   4 | 1 |
   5 | 0 |
   6 | 0 |
   7 | 0 |
   8 | 1 |
   9 | 0 |
  10 | 1 |
  11 | 1 |
  12 | 1 |
  13 | 0 |
  14 | 0 |
  15 | 1 |
  16 | 1 |
     +---+
Clustered permutation test p-value =    0.0047
Note: test may be anti-conservative if number of intervention clusters
> does not equal number of control clusters
```

The program outputs a column vector displaying the randomization scheme implemented in the study. The user can compare this to his or her data set to check that this matches the order of clusters in the data set. Following the procedure laid out in section 1.2, we obtain an adjusted clustered permutation test p-value, which in this case shows evidence of a difference in effect between the interventions. For illustration, we assume that only county-level covariates are available in the analysis. In practice, if individual-level data are available, we may also include those individual-level variables in the permutation test to improve power (Li et al. 2017).

More details about the programs and the above example can be found in our recent Stata conference presentation slides (Gallis et al. 2017).

4.3 Stratified covariate constrained randomization

As discussed in section 1.1, user-defined weights can be used to allow researchers to provide more weight to variables of their choice. For example, larger weights can be given to variables considered more important to balance than other covariates (see equation (2)). These user-defined weights can also be used to induce stratification on categorical variables by setting very large weights for such variables. This is implemented with the `stratify` option. For example, suppose we specify a weight of 1000 to the `location` variable. Then in all cases where `location` is not perfectly balanced between arms, at least 1000 will be added to the balance score. (If location is perfectly balanced, then $\bar{x}_{Tj} - \bar{x}_{Cj}$ reduces to 0 for j = location.) Thus, for a reasonably low cutoff value, the randomization schemes where `location` is unbalanced will have no chance of being included in the constrained space, and hence no chance to be selected as the final randomization scheme.

This can be illustrated by the code below. Because of the way `cvcrand` processes macros, all categorical variables should be placed at the end of the overall *varlist* if the `stratify` or `weights` option is specified. In addition, variables specified in the `stratify` option should be placed at the very end of the overall *varlist* and at the end of the *varlist* in the categorical option. Below, we place the `location` variable in the `stratify` option. This assigns the `location` variable a weight of 1000, while all other variables receive the default weight of 1.

```
. cvcrand inciis uptodate hispanic incomecat location, ///
> categorical(incomecat location) ntotal_cluster(16) ntrt_cluster(8) clustername(county) ///
> seed(10125) cutoff(0.1) balancemetric(l2) savedata(dickinson_constrained_strat) ///
> stratify(location)
You have specified the stratify option.  Please be sure the stratification variables
are placed at the end of the overall variable list, and last in the variable list of the
categorical option.
```

(*output omitted*)

```
Summary Stats | Balance Score
--------------+------------------
         Mean |  4.00e+06
    Std. Dev. |  5.49e+06
          Min |      1.16
           p5 |      6.22
          p10 |      9.22
          p20 |     16.45
          p25 |     21.00
          p30 |     28.37
          p50 |  3.75e+06
          p75 |  3.75e+06
          p95 |  1.50e+07
          Max |  6.00e+07

Cutoff value =      9.22

Value of selected balance score =      4.76

Row of constrained matrix =      903
```

(*output omitted*)

```
. table1, by(_allocation) ///
> vars(inci contn \ number contn \ uptod contn \ hisp contn \ loc cat \ incomecat cat)
>  ///
> format(%2.1f)
  ------------------------------------------------------------------------------
   Factor                      Level   _allocation = 0   _allocation = 1   p-value
  ------------------------------------------------------------------------------
   N                                   8                 8
  ------------------------------------------------------------------------------
   % in CIIS, mean (SD)                85.4 (5.2)        88.6 (9.0)           0.39
  ------------------------------------------------------------------------------
   % up-to-date, mean (SD)             40.4 (9.2)        41.3 (7.9)           0.84
  ------------------------------------------------------------------------------
   % Hispanic, mean (SD)               23.3 (15.1)       21.4 (11.3)          0.78
  ------------------------------------------------------------------------------
   Average income              Low     3 (38%)           2 (25%)              0.82
                               Med     3 (38%)           3 (38%)
                               High    2 (25%)           3 (38%)
  ------------------------------------------------------------------------------
   Location                    Rural   4 (50%)           4 (50%)              1.00
                               Urban   4 (50%)           4 (50%)
  ------------------------------------------------------------------------------
```

Equivalently, this stratification can be accomplished with the following code, using the `weights` option.

```
. cvcrand inciis uptodate hispanic incomecat location, ///
> categorical(incomecat location) ntotal_cluster(16) ntrt_cluster(8) clustername(county) ///
> seed(10125) cutoff(0.1) balancemetric(l2) savedata(dickinson_constrained_strat) ///
> weights(1 1 1 1 1000)
```

Now any randomization scheme with imbalance on `location` is appreciably larger than randomization schemes where `location` is balanced, ensuring that a randomization scheme stratified on `location` is chosen, assuming the user has specified a cutoff below the 50th percentile (since randomization schemes at the 50th percentile and above correspond to huge balance scores related to imbalance on `location`). We see from the `table1` output that location is now perfectly balanced across arms, as there are 4/8 rural counties in each arm.

An alternative strategy to achieve stratification is to perform covariate constrained randomization within each strata defined by location, and could be carried out by applying the code in Section 4.1 to subsets of the full data.

5 Discussion

We have introduced the `cvcrand` program to aid in the design of CRTs and the `cptest` program to aid in the subsequent analysis of CRTs designed using covariate constrained randomization. These programs are simple to use and do not require advanced programming skills, making them accessible to many researchers. Still, it is important for researchers to carefully consider features of the design—such as important baseline characteristics related to the outcome—and analysis, and *a priori* decide which covariates to include and which options to specify (e.g., what metric and cutoff value to use). Importantly, the `cvcrand` command should only be run once on any given data set, rather than rerunning until a desirable randomization scheme is selected, as this would technically alter the type I error.

There are some limitations to the programs. The `cvcrand` program will not work for a trial with more than two intervention arms. In addition, it only handles randomization for a parallel-arm cluster randomized trial design. These limitations reflect the current state of the research on constrained randomization. As the literature develops more, we plan to add more features and options to the program. Additionally, the program does not currently allow for modeling count outcomes using a Poisson regression. Most cluster trials have binary or continuous outcomes (Fiero et al. 2016), but we plan to extend the program to handle Poisson regression in the future.

Another consideration is that the permutation test only provides a test of significance, without reporting an intervention effect estimate or confidence interval. In practice, researchers may wish to use mixed-effects regression models or generalized estimating equations (GEE) to obtain the intervention effect estimate and its confidence interval. Such procedures have been shown to perform satisfactorily if the prognostic covariates used to perform covariate constrained randomization in the design phase are appropriately adjusted for in the analysis phase (Li et al. 2015, 2017). Nevertheless, the permutation test is attractive since it maintains the nominal test size and so could be used to evaluate the statistical significance of the intervention effect, even when a model-based approach is used.

It is recommended to use some form of restricted randomization when designing and implementing cluster randomized trials. Constrained randomization is ideally suited for this task when the number of clusters to be studied is small, especially when there are a relatively large number of baseline characteristics to balance. Our new programs `cvcrand` and `cptest` will facilitate constrained randomization and clustered permutation test analysis in CRTs, with the goal to provide more efficient design and analysis of CRTs, particularly those with a small number of clusters.

6 Acknowledgements

The authors would like to thank Alyssa Platt, Joe Egger, and Ryan Simmons of the Duke Global Health Institute Research Design and Analysis Core for testing and providing feedback on the programs. This research was funded in part by National Institutes of Health grant R01 HD075875 (PI: Dr. Joanna Maselko). In addition, the program `cvcrand` has already been applied to the design of the study *Evaluation of an Early Childhood Development Intervention for HIV-Exposed Children in Cameroon* (PI: Dr. Joy Noel Baumgartner). We also thank an anonymous reviewer, whose comments helped improve the final version of this paper.

7 References

Campbell, M. K., G. Piaggio, D. R. Elbourne, and D. G. Altman. 2012. Consort 2010 statement: extension to cluster randomised trials. *BMJ* 345. http://www.bmj.com/content/345/bmj.e5661.

Carter, B. R., and K. Hood. 2008. Balance algorithm for cluster randomized trials. *BMC Medical Research Methodology* 8: 65.

Dickinson, L. M., B. Beaty, C. Fox, W. Pace, W. P. Dickinson, C. Emsermann, and A. Kempe. 2015. Pragmatic cluster randomized trials using covariate constrained randomization: A method for practice-based research networks (PBRNs). *The Journal of the American Board of Family Medicine* 28(5): 663–672.

Diehr, P., D. C. Martin, T. Koepsell, and A. Cheadle. 1995. Breaking the matches in a paired t-test for community interventions when the number of pairs is small. *Statistics in Medicine* 14(13): 1491–1504.

Donner, A., and N. Klar. 2004. Pitfalls of and controversies in cluster randomized trials. *American Journal of Public Health* 26(1): 2–19.

Eldridge, S. M., O. C. Ukoumunne, and J. B. Carlin. 2009. The intra-cluster correlation coefficient in cluster randomized trials: a review of definitions. *International Statistical Review* 77(3): 378–394.

Fiero, M. H., S. Huang, E. Oren, and M. L. Bell. 2016. Statistical analysis and handling of missing data in cluster randomized trials: a systematic review. *Trials* 17(1): 72.

Gail, M. H., S. D. Mark, R. Carroll, and S. B. Green. 1996. On Design Considerations and Randomization-based Inference for Community Intervention Trials. *Statistics in medicine* 15: 1069–1092.

Gallis, J., F. Li, H. Yu, and E. L. Turner. 2017. cvcrand and cptest: Efficient Design and Analysis of Cluster Randomized Trials. In *Stata Conference*. https://www.stata.com/meeting/baltimore17/slides/Baltimore17_Gallis.pdf.

Hannan, P. J., D. M. Murray, D. R. Jacobs Jr, and P. G. McGovern. 1994. Parameters to aid in the design and analysis of community trials: intraclass correlations from the Minnesota Heart Health Program. *Epidemiology* 88–95.

de Hoop, E., S. Teerenstra, B. G. van Gaal, M. Moerbeek, and G. F. Borm. 2012. The "best balance" allocation led to optimal balance in cluster-controlled trials. *Journal of Clinical Epidemiology* 65(2): 132–7.

Ivers, N., M. Taljaard, S. Dixon, C. Bennett, A. McRae, J. Taleban, Z. Skea, J. Brehaut, R. Boruch, and M. Eccles. 2011. Impact of CONSORT extension for cluster randomised trials on quality of reporting and study methodology: review of random sample of 300 trials, 2000-8. *BMJ* 343: d5886.

Ivers, N. M., I. J. Halperin, J. Barnsley, J. M. Grimshaw, B. R. Shah, K. Tu, R. Upshur, and M. Zwarenstein. 2012. Allocation techniques for balance at baseline in cluster randomized trials: a methodological review. *Trials* 13: 120.

Kempe, A., A. W. Saville, L. M. Dickinson, B. Beaty, S. Eisert, D. Gurfinkel, S. Brewer, H. Shull, D. Herrero, and R. Herlihy. 2015. Collaborative centralized reminder/recall notification to increase immunization rates among young children: a comparative effectiveness trial. *JAMA Pediatr* 169(4): 365–73.

Klar, N., and A. Donner. 1997. The merits of matching in community intervention trials: a cautionary tale. *Statistics in Medicine* 16(15): 1753–64.

Li, F., Y. Lokhnygina, D. M. Murray, P. J. Heagerty, and E. R. DeLong. 2015. An evaluation of constrained randomization for the design and analysis of group-randomized trials. *Statistics in Medicine* 35(10): 1565–79.

Li, F., E. L. Turner, P. J. Heagerty, D. M. Murray, W. M. Vollmer, and E. R. DeLong. 2017. An evaluation of constrained randomization for the design and analysis of group-randomized trials with binary outcomes. *Statistics in Medicine* 36(24): 3791–3806.

Moulton, L. H. 2004. Covariate-based constrained randomization of group-randomized trials. *Clinical Trials* 1(3): 297–305.

Murray, D. M., S. L. Pals, J. L. Blitstein, C. M. Alfano, and J. Lehman. 2008. Design and analysis of group-randomized trials in cancer: a review of current practices. *Journal of the National Cancer Institute* 100(7): 483–491.

Raab, G. M., and I. Butcher. 2001. Balance in cluster randomized trials. *Statistics in medicine* 20(3): 351–365.

Sikander, S., A. Lazarus, O. Bangash, D. C. Fuhr, B. Weobong, R. N. Krishna, I. Ahmad, H. A. Weiss, L. Price, and A. Rahman. 2015. The effectiveness and cost-effectiveness of the peer-delivered Thinking Healthy Programme for perinatal depression in Pakistan and India: the SHARE study protocol for randomised controlled trials. *Trials* 16(1): 1–14.

Turner, E. L., F. Li, J. A. Gallis, M. Prague, and D. Murray. 2017a. Review of Recent Methodological Developments in Group-Randomized Trials: Part 1 - Design. *American journal of public health* 107(6): 907–15.

Turner, E. L., M. Prague, J. A. Gallis, F. Li, and D. Murray. 2017b. Review of Recent Methodological Developments in Group-Randomized Trials: Part 2 - Analysis. *American Journal of Public Health* 107(7): 1078–1086.

Turner, E. L., S. Sikander, O. Bangash, A. Zaidi, L. Bates, J. Gallis, N. Ganga, K. ODonnell, A. Rahman, and J. Maselko. 2016. The effectiveness of the peer delivered Thinking Healthy Plus (THPP+) Programme for maternal depression and child socio-emotional development in Pakistan: study protocol for a three-year cluster randomized controlled trial. *Trials* 17(1): 442.

About the authors

John Gallis, ScM, currently works as a biostatistician at Duke University in the Department of Biostatistics and Bioinformatics and at the Duke Global Health Institute. His research interests include the design and analysis of cluster randomized trials, and the analysis of data with other forms of clustering, including longitudinal and spatial data.

Fan Li, M.Sc., is a PhD student in the Department of Biostatistics and Bioinformatics at Duke University School of Medicine. His primary research interests include statistical methodology in the design and analysis of cluster randomized trials, longitudinal and spatial data analysis, and Bayesian methods.

Hengshi Yu, MSc, is a PhD student in the Department of Biostatistics at the University of Michigan. His research interests focus on multivariate statistics including the analysis of correlated data from cluster randomized trials.

Elizabeth Turner, PhD, is an assistant professor in the Department of Biostatistics and Bioinformatics and in the Duke Global Health Institute (DGHI) at Duke University. Her primary research interest is the design and analysis of cluster randomized trials (CRTs), with a special focus on translating methods to be accessible to the practitioner. She heads the DGHI Research Design and Analysis Core and has led the design and analysis of a range of CRTs in global mental health, malaria and cardiovascular disease in multiple settings around the world including in Kenya, Tanzania, Nepal, China and Pakistan.