P. J. Huber. Robust estimation of a location parameter. *Ann. Math. Statist.*, 35:73–101, 1964. ISSN 0003-4851. URL http://dx.doi.org.libproxy.lib.unc.edu/10.1214/aoms/1177703732. [p]

P. J. Huber and E. M. Ronchetti. *Robust Statistics*. Wiley Series in Probability and Statistics. John Wiley & Sons, 2nd edition, 2009. ISBN 978-0-470-12990-6. URL http://dx.doi.org.libproxy.lib.unc.edu/10.1002/9780470434697. [p]

J. T. Landis, H. An, and A. G. Bailey. *Dr4pl: Dose Response Data Analysis Using the 4 Parameter Logistic (4pl) Model*, 2018. URL https://bitbucket.org/dittmerlab/dr4pl. R package version 1.1.8. [p]

C. Lim. Robust ridge regression estimators for nonlinear models with applications to high throughput screening assay data. *Stat. Med.*, 34(7):1185–1198, 2015. ISSN 0277-6715. URL https://doi.org/10.1002/sim.6391. [p]

C. Lim, P. K. Sen, and S. D. Peddada. Robust nonlinear regression in applications. *J. Indian Soc. Agricultural Statist.*, 67(2):215–234, 291, 2013. ISSN 0019-6363. [p]

R. Mead. Plant density and crop yield. *Applied statistics*, pages 64–81, 1970. URL https://doi.org/10.2307/2346843. [p]

F. Mosteller and J. W. Tukey. Data analysis and regression: a second course in statistics. *Addison-Wesley Series in Behavioral Science: Quantitative Methods*, 1977. [p]

H. J. Motulsky and R. E. Brown. Detecting Outliers When Fitting Data with Nonlinear Regression - a New Method Based on Robust Nonlinear Regression and the False Discovery Rate. *BMC Bioinformatics*, 7:123, 2006. URL https://doi.org/10.1186/1471-2105-7-123. [p]

W. Oberhofer. The consistency of nonlinear regression minimizing the L_1-norm. *Ann. Statist.*, 10(1):316–319, 1982. ISSN 0090-5364. [p]

W. B. Pratt and P. Taylor. *Principles of Drug Action: The Basis of Pharmacology*. Churchill Livingstone, 3rd edition, 1990. URL https://doi.org/10.1021/jm00296a900. [p]

D. A. Ratkowsky. *Nonlinear Regression Modeling. A Unified Practical Approach*. Marcel Dekker, Inc., New York, 1983. [p]

A. H. Riazoshams, B. M. Habshah Jr, and A. Adam. On the outlier detection in nonlinear regression. *International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering*, 3 (12):1105–1111, 2009. [p]

C. Ritz, J. C. Streibig, and others. Bioassay analysis using R. *Journal of Statistical Software*, 12(5):1–22, 2005. URL https://doi.org/10.18637/jss.v012.i05. [p]

T. B. Robertson. On the normal rate of growth of an individual, and its biochemical significance. *Archiv für Entwicklungsmechanik der Organismen*, 25(4):581–614, 1908. URL https://doi.org/10.1007/bf02163864. [p]

G. A. F. Seber and C. J. Wild. *Nonlinear Regression*. Wiley Series in Probability and Mathematical Statistics: Probability and Mathematical Statistics. John Wiley & Sons, 1989. ISBN 0-471-61760-1. URL https://doi.org/10.1002/0471725315. [p]

Y. Wang, A. Jadhav, N. Southal, R. Huang, and D. T. Nguyen. A Grid Algorithm for High Throughput Fitting of Dose-Response Curve Data. *Curr Chem Genomics*, 4:57–66, 2010. URL https://doi.org/10.2174/1875397301004010057. [p]

Hyowon An
Department of Statistics and Operations Research
The University of North Carolina at Chapel Hill
USA
ahwbest@gmail.com

Justin T. Landis
Lineberger Comprehensive Cancer Center
The University of North Carolina at Chapel Hill
USA
jtlandis314@gmail.com

Aubrey G. Bailey
Lineberger Comprehensive Cancer Center
The University of North Carolina at Chapel Hill
USA
aubreybailey@gmail.com

J. S. Marron
Department of Statistics and Operations Research
The University of North Carolina at Chapel Hill
USA
marron@unc.edu

Dirk P. Dittmer
Lineberger Comprehensive Cancer Center
The University of North Carolina at Chapel Hill
USA
dirkdittmer@me.com

cvcrand: A Package for Covariate-constrained Randomization and the Clustered Permutation Test for Cluster Randomized Trials

by Hengshi Yu, Fan Li, John A. Gallis and Elizabeth L. Turner

Abstract The cluster randomized trial (CRT) is a randomized controlled trial in which randomization is conducted at the cluster level (e.g., school or hospital) and outcomes are measured for each individual within a cluster. Often, the number of clusters available to randomize is small (≤ 20), which increases the chance of baseline covariate imbalance between comparison arms. Such imbalance is particularly problematic when the covariates are predictive of the outcome because it can threaten the internal validity of the CRT. Pair-matching and stratification are two restricted randomization approaches that are frequently used to ensure balance at the design stage. An alternative, less commonly-used restricted randomization approach is covariate-constrained randomization. Covariate-constrained randomization quantifies baseline imbalance of cluster-level covariates using a balance metric and randomly selects a randomization scheme from those with acceptable balance by the balance metric. It is able to accommodate multiple covariates, both categorical and continuous. To facilitate implementation of covariate-constrained randomization for the design of two-arm parallel CRTs, we have developed the **cvcrand** R package. In addition, **cvcrand** also implements the clustered permutation test for analyzing continuous and binary outcomes collected from a CRT designed with covariate-constrained randomization. We used a real cluster randomized trial to illustrate the functions included in the package.

Introduction

Cluster randomized trials (CRTs) randomize clusters of individuals, such as schools, hospitals or clinics (Brown and Li, 2015). The CRT design is chosen when there are concerns of treatment contamination, when it is logistically easier to conduct the trial using cluster randomization and when intervention of interest is delivered at the group level (Turner et al., 2017a). CRTs have been used in many disciplines including social sciences, public policy, medicine and implementation science (Hayes and Moulton, 2009).

In this paper, we focus on the two-arm parallel cluster randomized trial. Usually, there are a total of $\binom{n}{n_T} = \frac{n!}{n_T!(n-n_T)!}$ ways to allocate n_T clusters to the intervention arm, out of a total of n clusters. For example, in a CRT with 10 clusters, 5 of which are assigned to the treatment arm and 5 to the control arm, there are $\binom{10}{5} = 252$ unique allocations in the simple randomization space. Each allocation is called a randomization scheme and when simple randomization is used, one of the 252 allocations is randomly selected and implemented in the CRT. Because it is common that there are only a limited (usually fewer than 20) number of clusters available in a CRT (Fiero et al., 2015), there may be a non-negligible chance of imbalance between arms regarding the distribution of baseline covariates (Moulton, 2004). If the covariates are predictive of the outcome, such imbalance may threaten the internal validity, can lead to loss of power and usually requires statistical adjustment in the analysis stage (Ivers et al., 2012).

Several design strategies are available to avoid reliance only on statistical adjustment that accounts for baseline covariate imbalance in the analysis phase. Two most popular ones are pair-matching and stratification (Ivers et al., 2012), both of which are examples of restricted randomization. Matching pairs of clusters according to similarity in the baseline covariate profile (e.g., location), and performs randomization within each pair. Stratification is similar to matching but instead of only considering pairs of clusters, the procedure forms strata of 2 or more clusters where each stratum includes clusters with similar baseline covariate profiles. As with matching, the clusters within each stratum are then randomized into the two arms and, when there are an even number of clusters in each stratum, there is perfect balance of strata across treatment and control arms. There are several limitations of these procedures. The power of a pair-matched study might decrease due to a small number of pairs and a small correlation between the matching covariates and the outcome (Diehr et al., 1995). Loss to follow-up of a single cluster may require the exclusion of the matched pair in the analysis, and reduces study power (Ivers et al., 2012). In addition, the intracluster correlation coefficient is not easy to compute from the matched pairs (Donner and Klar, 2004; Klar and Donner, 1997; Campbell et al., 2012). In a CRT with a small number of clusters, stratified randomization is only possible with a small

number of stratification covariates. Otherwise, a single cluster might be in a stratum and will cause an imbalance between the arms (Ivers et al., 2012). Given that CRTs often identify and recruit all clusters at the start of the trial, minimization, a common restricted randomization for individually randomized trials, is rarely applicable to CRTs. To deal with these limitations especially with a small number of clusters and more than a few baseline covariates to balance, alternative restricted randomization methods are necessary.

Covariate-constrained randomization is an alternative restricted randomization procedure (Ivers et al., 2012; Raab and Butcher, 2001). Unlike matching and stratification, covariate-constrained randomization uses a measure called a balance metric to quantify the difference in mean covariate values between the two arms for a given randomization scheme across all baseline covariates that we wish to balance. The simple randomization space is then constrained by keeping the subset of randomization schemes with which covariates are considered sufficiently balanced by the balance metric. A final scheme is then selected from this constrained space, and tends to exhibit better baseline balance on average than a scheme randomly selected without constraints. Compared with pair-matching and stratification, covariate-constrained randomization may be preferred due to its capacity to accommodate multiple covariates, both categorical and continuous. Further, the ICC calculation remains unaffected under constrained randomization.

Although covariate-constrained randomization is a promising design strategy for CRTs, it is not commonly used in practice. One possible reason is that it requires more programming than simple randomization, pair-matching or stratification. Therefore, to facilitate its implementation in the design and analysis of cluster randomized trials, we have developed the `cvrall` and `cvrcov` functions in the **cvcrand** package. The `cvrall` function performs constrained randomization based on covariate balance measured by a scalar balance metric and can assign weights to reflect the relative importance of candidate covariates. The `cvrcov` function performs constrained randomization based on multivariate balance defined through each single covariate, similar to the routine provided in Greene (2017).

From an analysis perspective, when a CRT is designed using covariate-constrained randomization, this design feature should be reflected in the analysis of the individual-level outcome data collected during the trial (Li et al., 2016, 2017). To do so, a permutation-based approach can be used. The permutation test, discussed in Gail et al. (1996), should account for the variability within the constrained randomization space. In other words, the resulting clustered permutation test obtains the p-value for the treatment effect by referencing the observed test statistics to the permutation distribution within the constrained space. We provide the `cptest` function in the **cvcrand** package to facilitate the implementation of this permutation test.

Methods

To demonstrate the utility of the **cvcrand** package, we describe the concepts of covariate-constrained randomization and the clustered permutation test using an example of a real cluster randomized trial presented in Dickinson et al. (2015). This CRT aims to compare a collaborative centralized reminder approach with a practice-based reminder approach for increasing the immunization rate in children aged 19 to 35 months from 16 counties in Colorado. Each county represents a cluster of children and eight counties are randomized to each arm. The collaborative reminder approach depends on the joint efforts between health department leaders and physicians to develop a centralized notification, either using telephone or mail, for all parents whose pre-school children are not up-to-date on immunizations. Parents from the practice-based arm are invited to attend a web-based training for reminder using the Colorado Immunization Information System. Although counties are the randomization unit, the binary outcome, immunization status, is to be measured for participating children. A list of nine county-level covariates are collected (see Table 1 for the complete list, of which income is listed twice as it is coded as both a continuous variable and a derived categorical variable) prior to randomization, and balance on these covariates are desired during the randomization phase.

Covariate-constrained randomization

Covariate-constrained randomization, henceforth referred to simply as constrained randomization, is a promising balancing technique for cluster randomized trials (CRTs), especially for those with a limited number of clusters (Hayes and Moulton, 2009). Constrained randomization usually involves the following steps: (i) specifying the baseline covariates that one wishes to balance; (ii) enumerating all possible randomization schemes or randomly simulating a large number of randomization schemes within the simple randomization space (duplicates are removed if the schemes are randomly simulated); (iii) retaining a constrained randomization space with a subset of schemes where sufficient balance across baseline covariates is achieved according to some pre-specified balance metric; (iv) randomly selecting a scheme from the constrained randomization space for implementation.

Table 1: County-level variables in the motivating example.

Variable name	Variable description
`location`	location ("rural" or "urban")
`inciis`	percentage of children aged 19-35 months in the Colorado Immunization Information System (CIIS)
`numberofchildrenages1935months`	number of children aged 19-35 months
`uptodateonimmunizations`	percentage of up-to-date on immunizations
`africanamerican`	percentage of African American
`hispanic`	percentage of Hispanic ethnicity
`income`	average income ($)
`incomecat`	category of average income ("low", "medium", and "high")
`pediatricpracticetofamilymedicin`	pediatric practice-to-family medicine practice ratio
`communityhealthcenters`	number of community health centers

Stratification can be viewed as a special case of constrained randomization. For instance, we could consider stratifying on a single binary baseline covariate, geographic location (rural or urban), for the immunization trial introduced previously. Suppose that 6 counties are located in the rural area and that 10 counties are located in the urban area. Stratified randomization ensures that half of the clusters in each stratum, defined by distinct values of the geographic location variable, are assigned to treatment and the rest to control. If we measure balance by the absolute differences in the average covariate values between arms, it follows that the stratified randomization space coincides with a constrained randomization space with zero balance scores when each stratum contains an even number of clusters.

Constrained randomization generalizes stratification and extends naturally to situations where there are several, possibly continuous, baseline covariates. The generalization is featured by defining a balance metric accommodating multiple covariates. A balance metric gives a quantitative assessment about the balance between the two arms for each randomization scheme, and essentially any sensible balance metric can be used. We first develop the `cvrall` function that balances covariates by scalar balance scores as in Raab and Butcher (2001) and Li et al. (2016, 2017). Suppose we wish to balance K baseline covariates, either cluster attributes or individual characteristics aggregated at the cluster level (dummy variables are used for categorical covariates). We denote n as the total number of clusters, n_T, n_C as the number of treated and control clusters (i.e., $n = n_T + n_C$), x_{ik} as the kth covariate ($k = 1, \ldots, K$) of cluster i. The $l2$ balance metric, first introduced by Raab and Butcher (2001), can be written as

$$B_{(l2)} = \sum_{k=1}^{K} \omega_k \left(\bar{x}_{Tk} - \bar{x}_{Ck}\right)^2 \tag{1}$$

where $\bar{x}_{Tk} = \sum_{i=1}^{n_T} x_{ik}/n_T$ and $\bar{x}_{Ck} = \sum_{i=n_T+1}^{n} x_{ik}/n_C$ are the means of the kth cluster-level variable in the treatment arm and the control arm, respectively, and ω_k is a pre-determined weight for the kth variable. We choose ω_k to be the inverse of the variance of the kth variable across all clusters following Raab and Butcher (2001) and Li et al. (2016), namely

$$\omega_k = 1/s_k^2 = \frac{n-1}{\sum_{i=1}^{n} \left(x_{ik} - \bar{x}_k\right)^2}$$

where $\bar{x}_k = \sum_{i=1}^{n} x_{ik}/n$.

An alternative $l1$ balance metric was introduced by Li et al. (2017) as

$$B_{(l1)} = \sum_{k=1}^{K} \tilde{\omega}_k \left|\bar{x}_{Tk} - \bar{x}_{Ck}\right| \tag{2}$$

where the notations are consistent with the $l2$ metric except for the weight $\tilde{\omega}_k$, which is chosen to be the inverse of the standard deviation of the kth variable, s_k. It has been shown that the two balance metrics perform similarly in constrained randomization, that both metrics are invariant to affine transformation of baseline covariates (Li et al., 2016, 2017), and that the resulting balance scores are free of the unit used to measure the baseline covariates as long as the unit of measurement is consistent across clusters. Finally, after the randomization schemes are enumerated or simulated, we simultaneously compute the balance scores for all schemes according to either the $l1$ or $l2$ metric, using the matrix formula given in Li et al. (2017). We refer the reader to Web Appendix B of Li et al. (2017) for additional computational details.

To reflect the relative importance of different covariates, one may specify different weights in the

$l1$ and $l2$ balance metric. To do so, we can modify the $l2$ balance metric to be

$$B_{(l2)} = \sum_{k=1}^{K} d_k \omega_k \left(\bar{x}_{Tk} - \bar{x}_{Ck}\right)^2 \tag{3}$$

where d_k is the user-defined weight for the kth variable. By default, $d_k = 1$ for all variables and equation (3) reduces to equation (1). When researchers consider a certain variable to be more "important" (in terms of prognostic value) than the others, a large user-defined weight $d_k > 1$ could be assigned to that variable when assessing the balance scores. Similarly, we modify the $l1$ balance metric by allowing for user-defined weights as

$$B_{(l1)} = \sum_{k=1}^{K} d_k \widetilde{\omega}_k \left|\bar{x}_{Tk} - \bar{x}_{Ck}\right| \tag{4}$$

Another important element of constrained randomization is the cutoff value, which we denote by $q \in (0, 1]$. If we write F_B as the empirical cumulative distribution function of the balance scores calculated using a balance metric, we could define the cutoff value as the percentile such that the constrained space contains schemes with balance scores no larger than $F_B^{-1}(q)$. Intuitively, the cutoff value measures the proportion of schemes relative to the simple randomization space. When $q = 1$, there is no constraint and simple randomization is implemented. When $q < 1$, only a subset of schemes with sufficient balance will be retained and constrained randomization is implemented. In the immunization trial example, we have in total $\binom{16}{8} = 12,870$ possible randomization schemes to allocate 8 clusters each to intervention and control. If we set $q = 0.1$, the constrained randomization space contains around 1288 schemes, allowing for ties in the balance scores.

Ideally, the cutoff value q should be small and away from 1 so that only the "more balanced" randomization schemes are retained in the constrained space. In fact, the power of statistical inference on the intervention effect tends to increase as q decreases if prognostic covariates are balanced by constrained randomization. However, the cutoff value q should not be too small since this may risk deterministic allocation of clusters into arms (Moulton, 2004), and may prohibit permutation inference given a fixed type I error rate (Li et al., 2016). In addition, the relationship between q and power is not monotone since power may stabilize once $q < 0.1$, as seen in a number of simulations presented in Li et al. (2017). For this reason, we set the default cutoff value of $q = 0.1$ in `cvrall`, unless specified otherwise by the user. Finally, we note that in our `cvrall` function, one could also specify the exact number of schemes kept in the constrained randomization instead of the cutoff quantile value, through the `numschemes` argument.

In addition to constraining the randomization space via a scalar summary score, we further developed the `cvrcov` function to implement constrained randomization with baseline balance defined directly through each covariate. This covariate-by-covariate constrained randomization places separate constraints on each covariate and ensures that the final allocation scheme satisfies marginal balance of each covariate. In particular, we follow the routine developed by Greene (2017) and constrain the arm mean difference (or arm total difference) to be no larger than a pre-specified value or a certain percentage of overall mean (or mean arm total). The covariate-by-covariate balance allows user-specified constraints on different covariates and is more flexible, but simulating the constrained randomization space usually requires more computations since the balance metric does not reduce to simple forms as the $l1$ or $l2$ scores.

To better understand the constrained randomization space, we also include a check on the randomization validity (Bailey and Rowley, 1987). Constraining the randomization may induce linkage or correlation between clusters so that certain pairs of clusters may always be allocated to the same arm (cluster coincidence) or never be allocated to the same arm (cluster separation), both of which lead to loss of randomization validity. To assess the degree of loss of validity, the `cvrall` and `cvrcov` functions provide summary statistics on cluster pairs that always or never appear together in the same arm, similar to the routine by Greene (2017). Such descriptive statistics may inform the appropriate selection of a constrained space.

Finally, enumerating all possible schemes in the entire simple randomization space may be computationally demanding, when there are quite a few clusters to randomize (e.g., more than 20). In that case, the `cvrall` and `cvrcov` functions in our package will randomly simulate a large number of randomization schemes and remove duplicates if any. By default, this large number is set to be 50,000, unless specified otherwise by the user through the `size` option. With this default setting, when the total number of schemes in the simple randomization space is no greater than 50,000, the enumeration method will be used. Otherwise, 50,000 schemes will be randomly simulated from the simple randomization space and duplicates will be removed to approximate the simple randomization space.

Clustered permutation test

After using constrained randomization in the design of a CRT, a permutation test can be used to test the intervention effect. We implement the clustered permutation test used in Gail et al. (1996) and Li et al. (2016) in the `cptest` function. Specifically, we denote the outcome of the jth individual ($j = 1, \ldots, m_i$) from the ith cluster ($i = 1, \ldots, n$) as Y_{ij}. During the analysis stage, researchers may wish to adjust for baseline covariates, which we denote by a vector $\mathbf{z}_{ij}$. The choice of adjustment variables may vary from study to study, and often depends on expert knowledge. Generally, it is a good practice to adjust for variables with high prognostic values that are already balanced by constrained randomization. For the permutation test, such a recommendation is not mandatory since the test size remains valid as long as the permutation distribution is obtained from the constrained randomization space (Li et al., 2016), even though adjusting for prognostic variables improves the test power (Li et al., 2016, 2017). However, if one prefers an unadjusted test, the following permutation inference still holds by setting $\mathbf{z}_{ij}$ as the null or empty vector.

The permutation test is implemented in a two-step procedure. In the first step, an outcome regression model is fitted for response Y_{ij} with covariates $\mathbf{z}_{ij}$. This is often done by fitting a linear regression model for continuous responses and a logistic regression model for binary responses, ignoring the clustering of responses. We then compute the predicted response for each individual by $\hat{Y}_{ij}$, which could be used to calculate the individual residual $r_{ij} = Y_{ij} - \hat{Y}_{ij}$. In the second step, cluster-specific residual averages are obtained as $\bar{r}_{i\cdot} = \sum_{j=1}^{m_i} r_{ij}/m_i$. The observed test statistic is then computed as

$$U = \frac{1}{n_T} \sum_{i=1}^{n} W_i \bar{r}_{i\cdot} - \frac{1}{n_C} \sum_{i=1}^{n} (1 - W_i)\, \bar{r}_{i\cdot} \tag{5}$$

where $W_i = 1$ if the ith cluster is assigned to the treatment arm and $W_i = 0$ otherwise, and $n_T = \sum_{i=1}^{n} W_i$, $n_C = \sum_{i=1}^{n} (1 - W_i)$ are the number of treated and control clusters.

Suppose there are S randomization schemes in the constrained randomization space. To obtain the permutation distribution of the test statistic, we permute the labels of the treatment indicator according to the constrained randomization space, and recompute a value for U_s ($s = 1, \ldots, S$) based on equation (5). The collection of these values $\{U_s : s = 1, \ldots, S\}$ forms the null distribution of the permutation test statistic. The p-value is then computed by

$$\text{p-value} = \frac{1}{S} \sum_{s=1}^{S} \mathbb{I}\left(|U_s| \geq |U|\right) \tag{6}$$

where $\mathbb{I}$ is the indicator function that equals 1 when $|U_s| \geq |U|$ and 0 otherwise.

Illustrative Examples

Constrained randomization by `cvrall`

We used the `cvrall` function to perform constrained randomization based on the CRT data published in Dickinson et al. (2015). To focus ideas, we selected five variables in Table 1 to balance in the design stage. These variables include `location` (categorical), `inciis` (continuous), `uptodateonimmunizations` (continuous) and `hispanic` (continuous). We further considered `incomecat` as a categorical variable to illustrate the use of `cvrall` in the presence of a factor variable. Of note, the `cvrall` function automatically converts the categorical variables into dummy variables when implementing the constrained randomization. For instance, here we categorized the county-level covariate `incomecat` into three levels based on sample tertiles: "low", "medium", and "high". Two dummy variables are then introduced to represent these three categories. The "high" level is by default considered as the reference level by alphanumerical order of the first letter. Similarly, when the permutation test is executed in the `cptest` function, each categorical covariates will be transformed into dummy variables before performing the analysis as well. It is also important to point out that there is more than one way to define dummy variables because any one of the levels of the categorical variable could be chosen as the reference level. In the `cvrall` function, if the variable is not specified as a factor with a specific reference level, we defined the reference level to be the first level by alphanumerical order. However, if one would like to specify other reference levels, it is possible to preprocess the data to manually create dummy variables before invoking the `cvrall` routines, or to specify the variables as factors with the specific reference levels.

In this trial, we would like to randomize 8 counties into the arm with a collaborative centralized reminder approach and 8 into the other arm with a practice-based approach. So we specified

`ntotal_cluster = 16` and `ntrt_cluster = 8` for the total number of clusters and the number of clusters in the treatment arm. Since the total number of possible schemes is $\binom{16}{8} = 12,870$, which is less than the default maximum number of simulated schemes (50,000), we enumerated all 12,870 schemes. The example syntax of the function is given as the following, where the `x=` argument references the data frame of the covariates that will be used in the calculation of balance scores and hence be balanced by constrained randomization.

```
Design_result <- cvrall(clustername = Dickinson_design$county,
                        balancemetric = "l2",
                        x = data.frame(Dickinson_design[ , c("location", "inciis",
                            "uptodateonimmunizations", "hispanic", "incomecat")]),
                        ntotal_cluster = 16,
                        ntrt_cluster = 8,
                        categorical = c("location", "incomecat"),
                        savedata = "dickinson_constrained.csv",
                        bhist = TRUE,
                        cutoff = 0.1,
                        seed = 12345,
                        check_validity = TRUE)
```

Here we used the balance scores calculated by the $l2$ metric as indicated by `balancemetric = "l2"`. The cateogrical variables were specified with `categorical = c("location","incomecat")`. Location has two levels: `"rural"` and code"urban"; the level `"rural"` is the reference level. As income category is a three-level categorical variable of `"low"`, `"med"`, and `"high"`, the level `"high"` is considered as the reference level and 2 dummy variables were created. Since we specified the cutoff value as `cutoff = 0.1`, the constrained randomization space only included the schemes with $l2$ balance scores less than the 10th percentile of the balance score distribution in the simple randomization space. Finally, we randomly sampled a scheme from the constrained space.

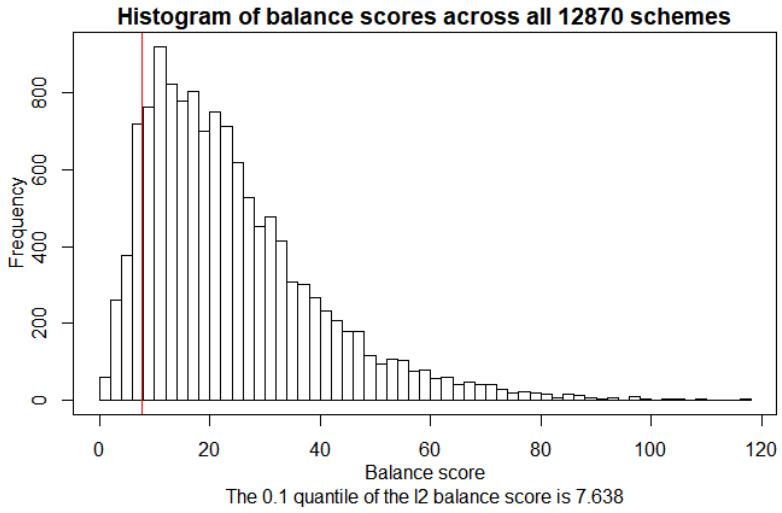

Figure 1: Histogram of balance scores across all 12870 schemes

We saved the constrained randomization space in a file named `dickinson_constrained.csv` in the current working directory. In this file, the first column is an indicator variable of whether the scheme is the final one selected by the program. The remaining columns records the constrained randomization matrix; each column of the matrix corresponds to a cluster, and each row of the matrix corresponds to an allocation scheme coded by 1's and 0's (1 if the cluster is assigned to the collaborative centralized reminder approach and 0 if assigned to the practice-based reminder approach). Furthermore, if simple randomization is used, namely `cutoff = 1`, the constrained randomization matrix has 12,870 rows and 16 columns. We provide the option to save the constrained randomization space to a local directory so that it could be used as an input for the permutation inference during the data analysis stage, which usually happens at a later calendar time.

To facilitate the understanding of the constrained randomization process, we could specify `bhist = TRUE` to generate a histogram displaying the distribution of all balance scores with a red line indicating the cutoff value (the 10th percentile). The sample histogram of balance scores is in Figure 1. The summary statistics of the balance scores are included in the `bscores` object, regardless of the `bhist =` option. As indicated below, the `bscores` object contains the cutoff value, the balance score corresponding to the selected scheme, and other quantiles of the balance score distribution.

```
> Design_result$bscores
1  score (selected scheme)   6.764
2              cutoff score   7.638
3                     Mean  24.000
4                       SD  15.775
5                      Min   1.161
6                       5%   5.826
7                      10%   7.638
8                      20%  10.849
9                      25%  12.221
10                     30%  13.840
11                     50%  20.578
12                     75%  31.621
13                     95%  55.486
14                     Max 116.656
```

In order to be transparent about the constrained randomization procedure, we also included additional summary messages in the following objects: `assignment_message`, `scheme_message`, `cutoff_message` and `choice_message`. These objects summarize the sample size and randomization ratio, the number of schemes used to calculate the balance score distribution, the balance metric and cutoff value, as well as the balance score of the selected scheme, respectively. For example, the sample size and randomization ratio are indicated in the following message:

```
> Design_result$assignment_message
[1] "You have indicated that you want to assign 8 clusters to treatment and 8 to control"
```

The final randomization scheme is included in the `allocation` object. In addition, we also provided a data frame containing the final randomization scheme in the `data_CR` element. The data frame includes the covariate values for each cluster in addition to the information on cluster allocation.

```
> Design_result$data_CR
   arm clustername location inciis uptodateonimmunizations hispanic incomecat
1    0           1    Rural     94                      37       44       Low
2    0           2    Rural     85                      39       23      High
3    0           3    Rural     85                      42       12       Low
4    1           4    Rural     93                      39       18      High
5    1           5    Rural     82                      31        6      High
6    0           6    Rural     80                      27       15       Med
7    1           7    Rural     94                      49       38       Low
8    0           8    Rural    100                      37       39       Low
9    1           9    Urban     93                      51       35       Med
10   1          10    Urban     89                      51       17       Med
11   0          11    Urban     83                      54        7      High
12   1          12    Urban     70                      29       13       Med
13   1          13    Urban     93                      50       13      High
14   0          14    Urban     85                      36       10       Med
15   1          15    Urban     82                      38       39       Low
16   0          16    Urban     84                      43       28       Med
```

To assess whether the selected constrained randomization scheme balances the baseline covariates, we provided a baseline table summarized under the selected randomization scheme. The baseline table indicates that the covariates are approximately balanced across the two arms, although more "urban" clusters are assigned to the collaborative centralized reminder approach. The baseline table is provided in the `baseline_table` element, and is illustrated below.

```
> Design_result$baseline_table
                                        arm = 0       arm = 1
n                                             8             8
location = Urban (%)                  3 (37.5)      5 (62.5)
inciis (mean (sd))                 87.00 (6.59)  87.00 (8.45)
uptodateonimmunizations (mean (sd))  39.38 (7.65)  42.25 (9.18)
hispanic (mean (sd))               22.25 (13.77) 22.38 (12.94)
incomecat (%)
   High                               2 (25.0)      3 (37.5)
   Low                                3 (37.5)      2 (25.0)
   Med                                3 (37.5)      3 (37.5)
```

Finally, we considered the validity of the randomization and used the `check_validity =` argument to summarize the cluster coincidence (cluster pairs assigned to the same arm) and cluster separation (cluster pairs assigned to different arms) within the constrained space. If `check_validity = TRUE`, we could obtain the relevant descriptive statistics in the `cluster_coin_des` object. The four rows in this object summarize the count and fraction of clusters appearing together, as well as count and fraction of clusters appearing in the different arms across the constrained randomization space. Recall that under simple randomization, no linkage or correlation is introduced between clusters and so each cluster pair has a 50% chance to appear together in the same arm and a 50% chance to appear in different arms. With the 0.1 cutoff value, the cluster pairs has a 47% chance to appear in the same arm on average, which is not too distant from the reference value 50%. However, there is a cluster pair that will appear in the same arm for only about 29% of the times (and appear in different arms for 71% of times), indicating some loss of validity. On the other hand, the constrained randomization routine offered by Greene (2017) includes default proportion values, 25% and 75%, as thresholds for loss of validity. That is to say, a reasonable constrained space should ensure each cluster pair appears in the same arm (and in different arms) for at least 25% of times and at most 75% times. Our constrained randomization space satisfies this condition.

```
> Design_result$cluster_coin_des
             Mean Std Dev Minimum 25th Pctl  Median 75th Pctl Maximum
samecount 600.600  88.807 368.000   551.750 603.000   648.500 804.000
samefrac    0.467   0.069   0.286     0.429   0.469     0.504   0.625
diffcount 686.400  88.807 483.000   638.500 684.000   735.250 919.000
difffrac    0.533   0.069   0.375     0.496   0.531     0.571   0.714
```

Stratified constrained randomization by `cvrall`

Of note, the `cvrall` function could perform constrained randomization with a stratification factor to ensure exact balance on that stratification factor. We still considered the above trial example, but now we wish to perform constrained randomization within each strata defined by the binary `location` variable. In other words, two strata of eight counties each will be defined depending on `location`, and constrained randomization is then performed based on the additional four covariates within each stratum. Motivated by the weighted $l1$ and $l2$ metrics (3), (4), we could assign a large weight (e.g., 1000) to `location` and ensure exact balance on that variable, while keeping the weights for other variables as 1 (`weights = c(1000,1,1,1,1)`). Intuitively, a large weight assigned to a covariate sharply penalizes any imbalance of that covariate, therefore the resulting randomization space approximates the one obtained by stratifying on `location`. The example syntax is provided below.

```
# Stratification on location, with constrained randomization on other
# specified covariates.
Design_stratified_result <- cvrall(clustername = Dickinson_design$county,
                                   balancemetric = "l2",
                                   x = data.frame(
                                     Dickinson_design[,
                                       c("location", "inciis",
                                         "uptodateonimmunizations", "hispanic",
                                         "incomecat")]),
                                   ntotal_cluster = 16,
                                   ntrt_cluster = 8,
                                   categorical = c("location", "incomecat"),
                                   weights = c(1000, 1, 1, 1, 1),
                                   cutoff = 0.1,
                                   seed = 12345)
```

Depending on the choice of `cutoff` value, the above syntax may not lead to a randomization space exactly the same as the one obtained after stratifying on `location`. The `cvrall` function also allows one to directly stratify on the `location` variable using the `stratify` option, as shown next. We omitted the baseline covariate table obtained from stratified constrained randomization, but just comment that final scheme ensures exact balance on the `location` variable so that each arm has now 4 urban counties and 4 rural counties.

```
# An alternative and equivalent way to stratify on location
Design_stratified_result <- cvrall(clustername = Dickinson_design$county,
                                   balancemetric = "l2",
                                   x = data.frame(
                                     Dickinson_design[ ,
```

```
    c("location", "inciis",
      "uptodateonimmunizations", "hispanic",
      "incomecat")]),
ntotal_cluster = 16,
ntrt_cluster = 8,
categorical = c("location", "incomecat"),
stratify = "location",
cutoff = 0.1,
seed = 12345)
```

Constrained randomization by `cvrcov`

We additionally provided the `cvrcov` function to perform covariate-by-covariate constrained randomization, similar to the routine provided by Greene (2017). This approach is particularly attractive for its flexibility in directly balancing each covariate. We still considered our example trial where we randomized 8 counties into the each arm for illustration. We specified `ntotal_cluster = 16` and `ntrt_cluster = 8` for the total number of clusters and the number of clusters in the treatment arm. Since the total number of possible schemes is $\binom{16}{8} = 12,870$, which is less than the default maximum number of simulated schemes (`50,000`), we enumerated all 12870 schemes.

As the covariate-by-covariate constrained randomization acts on the numeric values of each variable, we transformed the values of the `location` to be numeric with `"Rural"` being `1` and `"Urban"` being `0`. For illustrative purposes, we also used the numeric average income values rather than its categories in this example. The `x =` argument points to the data frame containing the covariates that will be balanced by constrained randomization routine.

Table 2: Example syntax of balancing constraints.

Syntax	Explanation
`any`	no constraints, any arm means or arm totals are acceptable
`s5`	arm totals must differ in absolute value by no more than 5
`sf.5`	arm totals must differ in absolute value by no more than 0.5 times the mean arm total
`m10`	arm means must differ in absolute value by no more than 10
`mf0.2`	arm means must differ in absolute value by no more than 0.2 times the overall mean
`mf.5`	arm means must differ in absolute value by no more than 0.5 times the overall mean

The `cvrcov` function works the same way as the `cvrall` function, except for that the former requires additional syntax to specify the balancing constraints for each covariate. The syntax used to balance each covariate is the same those used in Greene (2017). Specifically, if the first letter is specified as `m`, the balancing constraint acts on means, whereas if the first letter is `s`, the balancing constraint acts on sums or totals. If the second letter is `f`, the balancing constraint will be compared to a fractional of a population quantity (overall mean or mean arm total), otherwise the constraint will be compared to an actual value. A numeric constraint will follow the specified letters and indicates the tightness of the constraint. Additional examples are provided in Table 2.

```
Dickinson_design_numeric <- Dickinson_design
Dickinson_design_numeric$location = (Dickinson_design$location == "Rural") * 1

Design_cov_result <- cvrcov(clustername = Dickinson_design_numeric$county,
                            x = data.frame(Dickinson_design_numeric[ , c("location", "inciis",
                                                                         "uptodateonimmunizations",
                                                                         "hispanic", "income")]),
                            ntotal_cluster = 16,
                            ntrt_cluster = 8,
                            constraints = c("s5", "mf.5", "any", "mf0.2", "mf0.2"),
                            categorical = c("location"),
                            savedata = "dickinson_cov_constrained.csv",
                            seed = 12345,
                            check_validity = TRUE)
```

We specified `constraints = c("s5","mf.5","any","mf0.2","mf0.2")` for the five covariates respectively. As indicated above, `s5` indicates that the allocation scheme should ensure that the arm totals differ in absolute value by no more than 5. Synaxt `mf.5` indicates that the allocation scheme should ensure that the arm means differ by no more than 0.5 times the overall mean for `inciis`, among others. We saved the resulting constrained randomization space as `dickinson_cov_constrained.csv`.

Similar to `cvrall`, the `cvrcov` routine included additional summary messages in the following objects: `assignment_message` and `scheme_message`. These two objects summarize the sample size and randomization ratio, the number of schemes enumerated or simulated before applying the constraints. In addition, a data frame containing the selected final allocation scheme is saved in the `data_CR` element as follows.

```
> Design_cov_result$data_CR
   arm id location inciis uptodateonimmunizations hispanic income
1    0  1        1     94                      37       44  35988
2    1  2        1     85                      39       23  67565
3    0  3        1     85                      42       12  35879
4    0  4        1     93                      39       18  63617
5    1  5        1     82                      31        6  59118
6    0  6        1     80                      27       15  57179
7    1  7        1     94                      49       38  29738
8    1  8        1    100                      37       39  37350
9    1  9        0     93                      51       35  52923
10   0 10        0     89                      51       17  58302
11   0 11        0     83                      54        7  93819
12   0 12        0     70                      29       13  54839
13   1 13        0     93                      50       13  63857
14   1 14        0     85                      36       10  53502
15   0 15        0     82                      38       39  39570
16   1 16        0     84                      43       28  52457
```

To evaluate whether the selected constrained randomization scheme balances the baseline covariates, we provided a baseline table summarized under the that selected randomization scheme. The baseline table indicates that the covariates are well balanced across the two arms, with an equal number of “urban" clusters assigned to each reminder approach.

```
> Design_cov_result$baseline_table
                                              arm = 0             arm = 1
n                                                   8                   8
location = 1 (%)                              4 (50.0)            4 (50.0)
inciis (mean (sd))                          84.50 (7.76)        89.50 (6.35)
uptodateonimmunizations (mean (sd))         39.62 (9.44)        42.00 (7.43)
hispanic (mean (sd))                       20.62 (13.38)       24.00 (13.09)
income (mean (sd))                  54899.12 (19130.82) 52063.75 (12800.82)
```

The `cvrcov` function permits the check of randomization validity (Bailey and Rowley, 1987), and summarizes the cluster coincidence and separation statistics in the `cluster_coin_des` object. The result indicates that all cluster pairs appear together in the same arm at least 37% and at most 55% of the times across the constrained randomization space. Using the 25% and 75% threshold, the summary statistics indicate that the constrained randomization does not severely depart from validity. Finally, the `cvrcov` function summarizes the information of the constrained space in the `overall_allocations` and `overall_summary` objects, which are suppressed here due to limited space. In short, the summary information informs that the there are in total 12,870 allocations and 5,776 ($\approx$ 45%) satisfied the balancing constraints.

```
> Design_cov_result$cluster_coin_des
              Mean Std Dev  Minimum 25th Pctl   Median 75th Pctl  Maximum
samecount 2695.467 197.148 2138.000  2567.000 2720.000  2824.500 3182.000
samefrac     0.467   0.034    0.370     0.444    0.471     0.489    0.551
diffcount 3080.533 197.148 2594.000  2951.500 3056.000  3209.000 3638.000
difffrac     0.533   0.034    0.449     0.511    0.529     0.556    0.630
```

Clustered permutation test by `cptest`

Since the immunization study is an ongoing trial, we used simulated outcome data to demonstrate the clustered permutation test with the above example where constrained randomization was performed using `cvrall` based on the 5 covariates (the selected scheme had a balance score of 6.764). The same syntax applies to the constrained randomization results obtained from `cvrcov` and so is not considered further here. Suppose that the researchers were able to assess 300 children in each county, and the trial is randomized according to the selected final scheme. For illustration, we chose the covariates to be adjusted in the test $\mathbf{z}_{ij}$ as the list of covariates x_i balanced by design. This step is in line with the

recommendation of Li et al. (2017) that adjusting for prognostic factors in the analysis improves the test power.

To generate the correlated binary outcome of whether the children is eventually up-to-date on immunizations (`1`) or not (`0`), we used a generalized linear mixed model (GLMM) with a logistic link to induce correlation by including a random intercept at the county level. The intraclass correlation coefficient (ICC) is usually used to quantify the degree of association between individual outcomes in a cluster (county). We used the latent response definition of binary ICC defined by variance components in the GLMM (Eldridge et al., 2009). The ICC was set to be 0.01, which is a reasonable value for population health studies (Hannan et al., 1994). The outcome variable depends on the county-level covariates used in performing the constrained randomization, as previously mentioned, and we simulated a treatment effect so that the collaborative reminder approach increases up-to-date immunization rates compared to the practice-based reminder approach (odds ratio equals to $e^{0.5} \approx 1.649$). The binary outcome for each individual child is generated from a Bernoulli model with event probability specified by the GLMM.

We performed the clustered permutation test using the `cptest` function for the binary outcome of the status of up-to-date on immunizations. As indicated in Li et al. (2016), valid permutation test under constrained randomization should only shuffle the treatment label within the constrained space, and so it is important to save and input the constrained randomization space in the design stage (the file named `dickinson_constrained.csv`). The permutation test is performed by first regressing the outcome on the five covariates, `inciis`, `uptodateonimmunizations`, `hispanic`, `location`, and `incomecat`. As the last two covariates are categorical, the `cptest()` function creates dummy variables and set reference levels according to alphanumerical order, matching the steps in `cvrall`. Of note, had different reference levels been selected for the constrained randomization design procedure, the corresponding dummy coding should be reflected in the analysis phase when the clustered permutation test is used. We specified `outcometype` to be "binary" so that logistic regression is performed to compute the residuals. An example syntax of the function is given as follows.

```
Analysis_result <- cptest(outcome = Dickinson_outcome$outcome,
                          clustername = Dickinson_outcome$county,
                          z = data.frame(Dickinson_outcome[, c("location", "inciis",
                              "uptodateonimmunizations", "hispanic", "incomecat")]),
                          cspacedatname = "dickinson_constrained.csv",
                          outcometype = "binary",
                          categorical = c("location","incomecat"))
```

The covariates to be adjusted for in the permutation test is indicated in the `z =` option, which matches the covariate matrix used in `cvrall` for constrained randomization. If one wishes to an unadjusted permutation test, one could leave out the `z =` option as it is an optional argument. The output of `Analysis_result` includes the final scheme selected by design (`FinalScheme` object), the p-value of the test (`p-value` object) and a sentence to describe the p-value (`pvalue_statement` object). We omitted the code output here for brevity, but comment that, in this example, the p-value equals to `0.042`, indicating that there is a significant difference in the effect of the interventions on the outcome of up-to-date on immunizations, if testing is performed at the 5% significance level. Again, if the constrained randomization is performed by `cvrcov`, we could use the `cptest` function in a similar way once we provided the constrained permutation matrix obtained from `cvrcov` in the `cspacedatname =` argument.

Summary

The **cvcrand** package contains three main functions for the design and analysis of cluster randomized trials. Given that it is common for such trials to enroll a small number of clusters and that this gives rise to chance imbalance in covariates that are predictive of the outcome, the `cvrall` and `cvrcov` functions can be used to implement covariate-constrained randomization in the design phase to ensure better balance. The `cvrall` function uses a balance metric to quantify balance across multiple cluster-level covariates, whereas the `cvrcov` allows for covariate-by-covariate balance and could potentially be more flexible. For analysis of the individual-level outcome data collected in the CRT, the `cptest` function could help perform the clustered permutation test, which accommodate both continuous and binary outcomes and should be treated as a flexible alternative to model-based analysis.

There are several limitations of the **cvcrand** package. First, the `cvrall` and `cvrcov` only deal with two-arm parallel cluster randomized trials and may not be directly applied to balance covariates in other designs such as the stepped wedge designs (Hussey and Hughes, 2007; Li et al., 2018). Second, although the `cptest` function performs a valid analysis for individual-level outcome data when there is an equal number of clusters per arm, the test may be anti-conservative when there is an

unequal number of clusters per arm (Gail et al., 1996). Furthermore, our `cptest` routine does not provide a confidence interval for the intervention effect estimate, and additional programming is required to obtain a permutation confidence interval. Essentially, the permutation test will be inverted to numerically search for the interval limits, as is done in Gail et al. (1996) for an unadjusted test under simple randomization. For the adjusted test under constrained randomization, the following steps could be carried out: (i) hypothesize an treatment effect δ on the link function scale; (ii) obtain the residuals $r_{ij} = Y_{ij} - \hat{Y}_{ij}$, where $\hat{Y}_{ij}$ is estimated from regressing Y_{ij} on $\mathbf{z}_{ij}$ and the hypothesized treatment effect; (iii) perform the permutation test under the constrained randomization space and obtain a p-value; (iv) repeat steps (i)-(iii) for different values of δ and the confidence interval is the collection of δ such that the p-value is at least 0.05. We noticed that few studies were present in the CRT literature to evaluate the performance of permutation intervals that adjust for covariates under constrained randomization, and this is an avenue for future research. On the other hand, it is also important to notice that point and interval estimates could be easily obtained from model-based approaches, with caveats discussed in Li et al. (2016). In the class of model-based approaches, the most commonly-used approaches are the generalized linear mixed model (GLMM) model approach, which estimates the cluster-specific conditional effect, and the generalized estimating equations (GEE) approach, which estimates the population-averaged or marginal effect (Turner et al., 2017b). In each case, it has been demonstrated that the model-based analyses should account for the prognostic covariates used in the design (Li et al., 2016, 2017). Finally, although the `cptest` function can handle both continuous and binary outcomes, we have not yet extended the function to accommodate count outcomes. In summary, these limitations reflect the current research on constrained randomization. We plan to update the **cvcrand** package as the theory and knowledge of these procedures develop in the future.

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).

Bibliography

R. Bailey and C. Rowley. Valid randomization. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 410(1838):105–124, 1987. URL https://doi.org/10.1098/rspa.1987.0030. [p]

A. W. Brown and P. Li. Best (but oft-forgotten) practices: Designing, analyzing, and reporting cluster randomized controlled trials. *The American Journal of Clinical Nutrition*, 102(2):241–248, 2015. URL https://doi.org/10.3945/ajcn.114.105072. [p]

M. K. Campbell, G. Piaggio, D. R. Elbourne, and D. G. Altman. Consort 2010 statement: Extension to cluster randomised trials. *Bmj*, 345:e5661, 2012. URL https://doi.org/10.1136/bmj.e5661. [p]

L. M. Dickinson, B. Beaty, C. Fox, W. Pace, W. P. Dickinson, C. Emsermann, and A. Kempe. 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, 2015. URL https://doi.org/10.3122/jabfm.2015.05.150001. [p]

P. Diehr, D. C. Martin, T. Koepsell, and A. Cheadle. 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, 1995. URL https://doi.org/10.1002/sim.4780141309. [p]

A. Donner and N. Klar. Pitfalls of and controversies in cluster randomization trials. *American Journal of Public Health*, 94(3):416–422, 2004. URL https://doi.org/10.2105/AJPH.94.3.416. [p]

S. M. Eldridge, O. C. Ukoumunne, and J. B. Carlin. The intra-cluster correlation coefficient in cluster randomized trials: A review of definitions. *International Statistical Review*, 77(3):378–394, 2009. URL https://doi.org/10.1111/j.1751-5823.2009.00092.x. [p]

M. Fiero, S. Huang, and M. L. Bell. Statistical analysis and handling of missing data in cluster randomised trials: Protocol for a systematic review. *BMJ open*, 5(5):e007378, 2015. URL https://doi.org/10.1186/s13063-016-1201-z. [p]