The Stata Journal (2017)
17, Number 3, pp. 630–651

Randomization inference with Stata: A guide and software

Simon Heß
Goethe Universität Frankfurt am Main
Faculty of Economics and Business Administration
Frankfurt, Germany
hess@econ.uni-frankfurt.de

Abstract. Randomization inference or permutation tests are only sporadically used in economics and other social sciences—this despite a steep increase in randomization in field and laboratory experiments that provide perfect experimental setups for applying randomization inference. In the context of causal inference, such tests can handle problems often faced by applied researchers, including issues arising in the context of small samples, stratified or clustered treatment assignments, or nonstandard randomization techniques. Standard statistical software packages have either no implementation of randomization tests or very basic implementations. Whenever researchers use randomization inference, they regularly code individual program routines, risking inconsistencies and coding mistakes. In this article, I show how randomization inference can best be conducted in Stata and introduce a new command, `ritest`, to simplify such analyses. I illustrate this approach's usefulness by replicating the results in Fujiwara and Wantchekon (2013, *American Economic Journal: Applied Economics* 5: 241–255) and running simulations. The applications cover clustered and stratified assignments, with varying cluster sizes, pairwise randomization, and the computation of nonapproximate p-values. The applications also touch upon joint hypothesis testing with randomization inference.

Keywords: st0489, ritest, randomization inference, permutation tests, treatment effects, causal inference

1 Introduction

The past decades saw a steep increase in using randomization as a tool to identify causal relationships in experimental studies. Randomized control trials (RCTs) enable researchers to credibly identify causal relationships. Such studies are conducted in many fields to test the effectiveness of large development interventions, to understand effects and side effects of medical treatments, and to shed light on human behavior in psychology and behavioral economics. Thus it is paramount to ensure that statistical hypothesis tests—usually materializing in p-values—are conducted correctly. As an alternative to classical inference, Fisherian randomization inference provides the means to assess whether an observed realization of a statistic, such as a treatment-effect (TE) estimate, is unlikely to be observed by chance and is hence statistically significant.

The most common methods to study data obtained with RCTs are well-established econometric methods. Most of these methods are geared toward making inference with large samples drawn from infinite populations, relying on asymptotic properties of estimators; these methods generally make strong model assumptions. As Young (2016), with reference to Leamer (2010), points out, data obtained from RCTs at times do not meet the requirements to rely on asymptotic properties. However, randomization in the experimental design enables researchers to use the powerful tool of randomization inference, as introduced by Fisher (1935), further developed by Rosenbaum (2002), and recently used in first-rate articles in economics and political science (including, Fujiwara and Wantchekon [2013], Ichino and Schündeln [2012], Cohen and Dupas [2010], Bloom et al. [2006]). In RCTs, the researcher usually knows exactly how the randomization was carried out, and randomization inference uses this knowledge of randomization to assess whether observed outcomes in a given sample are likely to have been observed by chance even if treatment had no effect. There is also a growing literature applying randomization inference to data obtained outside the realm of RCTs. For example, Cattaneo, Titiunik, and Vazquez-Bare (Forthcoming) and Cattaneo, Frandsen, and Titiunik (2015) use randomization inference in the context of regression discontinuity designs, and Ganong and Jäger (2015) use randomization inference in the context of regression kink designs.

For a more detailed primer in permutation tests, see the books by Rosenbaum (2010) or Imbens and Rubin (2015), which provide detailed introductions. The intuition behind randomization inference is fairly straightforward. Consider a data-generating process that involves a random draw from a known distribution, for example, treatment assignment. Under the null hypothesis that this random draw has no influence on other aspects of the data, the distribution of any statistics derived from the data is known too. These distributions can be numerically obtained through Monte Carlo methods by computing the desired statistics repeatedly for varying realizations of the random draw. To test the null hypothesis that there is no effect of the original random draw on the data, a researcher merely needs to assess whether the sample realization of the statistic is consistent with the numerically inferred distribution. This last step is most commonly done using the rank statistic.

In the usual large-sample approach, that is, when basing inference on asymptotic variance estimators, one assumes the estimation sample will be random draws from an infinite population that one wants to learn about. Of course, this is only one of several ways to conceptualize experiments. Under randomization inference, the experimental sample is taken as fixed. The only aspect assumed to be random is the assignment to either arm of the treatment—or whatever other variable is of interest.

The first step in conducting randomization inference is defining the test statistic of interest, such as the regression coefficient in a regression of the outcome on treatment and controls. The exact distribution of this test statistic under the sharp null hypothesis of no TE is obtained by computing the statistic for each possible alternative assignment of treatment. Because randomization inference is not built on or assumes random sampling from the population, it does not rely on large-sample theory to apply. It merely requires knowledge of the randomization process. Furthermore, while the statistic studied may be based on a model—such as average TE estimates from regressions with control variables—the validity of the carried-out inference does not depend on the validity of that model or the size of the sample. In that sense, randomization inference does not make assumptions about the sampling but is always conditional on the sample at hand.

Fisherian randomization inference for TE analysis in the social sciences is still limited to a few articles and is often sidelined into footnotes or appendix tables of robustness checks. Young (2016) revisited results of almost 2,000 regressions from more than 50 articles published in leading journals, repeating their analyses using randomization inference. Young not only computes p-values for each of these tests but also uses randomization inference to conduct joint tests for related coefficients within and across equations. He finds that more than 10% of significant tests do not remain significant at the same level. More than 30% of equations with multiple treatment measures and individually significant coefficients fail to pass joint tests of significance. Lastly, when accounting for multiple hypotheses testing, he finds that less than half the experimental articles can maintain the hypothesis that any TE exists at all.

`ritest` is an easy yet versatile toolkit for randomization inference in Stata. As already stated by Bruhn and McKenzie (2009), conducting randomization inference currently requires significant programming skills from a researcher. When researchers use randomization inference, they often write their own code, which increases unnecessary additional risk of coding errors. When do-files for conducting randomization inference tests are published alongside research articles, they tend to be highly specific to the data they are applied to and can get very large.[1] I take this as a strong indication that `ritest` and this article, serving as a guideline, will be useful. I argue that more researchers would use randomization inference if standard statistical software endowed them with the necessary means, and I introduce the `ritest` command for this purpose. Notable in this context are the R-packages `RItools` (Bowers, Fredrickson, and Hansen 2016) and `ri` (Aronow and Samii 2012), which provide tools for randomization inference in R. However, the software available within Stata for this is limited. The built-in `permute` command only permutes individual observations, allowing the user to specify strata, but fails when more complicated treatment assignments—for example, clustered assignment—are to be modeled. `rdlocrand`, by Cattaneo, Titiunik, and Vazquez-Bare (2016), is a comprehensive user-written package that provides the means to conduct randomization inference specifically in the context of regression discontinuity designs under local randomization.

In this article, I focus on evaluating TE with binary treatment. The logic of randomization inference equally applies to other data-generating processes. While this is not explicitly covered, this article can still be useful for researchers working with continuous treatments and observational data. `ritest` is in no way restricted to handle only binary treatment indicators. In any situation where assumptions about the underlying distribution of independent variables of interest are justified, randomization inference is applicable.

[1] For example, Cohen and Dupas (2010) wrote 332 lines of code just to compute their p-values.

The rest of this article is structured as follows: Section 2 presents the theoretical background for randomization inference in Stata. Section 3 introduces `ritest` and its syntax. Finally, section 4 shows three alternative applications.

2 Theoretical building blocks

Fisherian randomization inference produces the distribution of a test statistic under a designated null hypothesis, allowing a researcher to assess whether the actually observed realization of the statistic is "extreme" and hence whether the null hypothesis has to be rejected. Such a test does not rely on assumptions regarding sample size, the accuracy of the model for the data-generating process, or the distribution of a model's noise term. Unlike asymptotic inference, which assumes each observation to be a draw from a distribution of outcomes (or noise terms), randomization inference takes the set of study subjects as fixed and regards only the treatment assignment as a random draw.

Because randomization inference is based on permuting or resampling the variable of interest, it is often referred to as permutation tests, [re]randomization tests, or resampling tests. In this article, I will make no distinction and use these terms interchangeably.

There is flexibility in choosing the test statistic used in randomization inference. In TE analyses, the coefficient estimate or the t statistic of the TE estimate are the most common choices. Unless stated differently, I will be assuming study of the estimand τ obtained by comparing means in the treatment and control group or from a regression

$$Y = \tau D + X'\beta + \varepsilon$$

where Y is a vector containing the observed outcome of interest for each observation, D is a vector indicating treatment status for each observation, and X is a matrix of pretreatment control variables. In some cases, researchers have shown that basing randomization inference on the t statistics rather than the coefficient estimates can be beneficial. MacKinnon and Webb (2016) discuss cases where the two approaches yield different results, and Young (2016) also compares both methods. Each particular statistic will have a different alternative hypothesis against which that particular statistic has the most power. The ideal choice will depend on the particularities of the data at hand, and I recommend using simulations to make an informed decision.

In general, any null hypothesis that specifies a TE for every observation can be tested with randomization inference. The typical randomization inference null hypothesis to test would be the sharp hypothesis of a zero TE,

$$y_i(D=1) = y_i(D=0) \quad \text{for all } i = 1, 2, \ldots, n$$

where $y_i(D=1)$ and $y_i(D=0)$ denote the outcomes that would be observed for individual i under treatment and under no treatment, respectively. Note that this differs from the null hypothesis of no average TE, which is usually used in asymptotic TE analysis.[2] In what follows, I refer to the sharp null hypothesis of no TE, unless explicitly stated otherwise.

To obtain the distribution of the test statistic under the null hypothesis, one must compute the test statistic for each possible permutation of the treatment vector. Because the set of possible permutations can get excessively large, it is common practice to use only several thousand random draws for the treatment vector.

2.1 Modes of treatment assignment

Enumerating all or a random subset of possible permutations of treatment can be trivial if treatment assignment corresponds to individual-level coin tosses. You can test this with Stata's `permute` command. However, lab and field experiments are using more sophisticated modes of randomization. Balancing methods, such as stratification or pairwise matching, are commonly used to ensure balance between treatment and control groups. Similarly, treatments are often assigned in clusters (schools, villages, etc.) to contain spillover or simply because there is no other way. More sophisticated experimental designs may feature multilevel assignment procedures or multidimensional treatments.

This poses difficulty for computing standard errors and hence for testing hypotheses. For treatment assignment in clusters, this difficulty stems from the fact that regression model errors will not be independent within clusters because outcome variables typically have nonzero intracluster correlation while treatment assignment is mechanically correlated within clusters (Cameron and Miller 2015). Typical examples include treatment assignment of school classes or villages. If a single school class benefits from unobservable teacher quality and this class falls into the treatment group, this unobserved variable will affect multiple treatment observations at once. Inference regarding the TE will need to account for the fact that model errors are correlated. Otherwise, standard errors will be misleadingly small.

Especially in smaller samples, researchers tend to stratify the randomization process to ensure balance of key baseline variables between treatment and control group. Bugni, Canay, and Shaikh (2016), like Bruhn and McKenzie (2009), show that when

[2] The sharp null hypothesis is stronger and implies the weaker hypotheses of no average TE. For a discussion of this issue, see Young (2016). Other sharp hypotheses can be equally tested in this framework, such as $y_i(D=1) = y_i(D=0) + \alpha_i$ for all $i = 1, 2, \ldots, n$, where α_i are specified as part of the hypothesis but may take on a different value for each observation i.

inference neglects the stratification, t tests tend to be overly conservative. This result extends to other covariate balancing methods such as pairwise randomization, in which observations are grouped into pairs, maximizing a measure of similarity within pairs. Subsequently, one observation within each pair is assigned treatment, while the other one is assigned to the control group.

Assuming enough clusters were used, clustered treatment assignment can be accounted for using cluster–robust standard errors (Cameron and Miller 2015). Similarly, the problems that stratification poses for standard-error estimation can be solved using strata fixed effects (Bugni, Canay, and Shaikh 2016; Cameron and Miller 2015; Bruhn and McKenzie 2009; Duflo, Glennerster, and Kremer 2008). In many of these cases, however, the "correct" specification is not straightforward. Besides, what constitutes a "large enough" sample is still a subject of ongoing debate and depends on various factors of the study design. For example, cluster–robust inference can fail even for larger samples if clusters vary in size (MacKinnon and Webb 2016). Similarly, if the randomization protocol involves redrawing treatment until a certain balance criterion was met, it is unclear what parametric specification to use (Bruhn and McKenzie 2009).

In contrast, to compute randomization inference p-values, one need only replicate the original assignment method and determine the distribution of the test statistic under the null hypothesis, F^τ, through resampling. If the independent variable of interest is a random treatment assignment of any kind, the original procedure using the same randomization device can simply be repeated. However, note that a resampling test that does not respect the original treatment assignment method will also produce inappropriate results (see Rosenberger and Lachin [2016, sec. 6.4]). For example, if the original process involved treatment assignment by villages, the resampling or permutation should not be based on individual-level permutations. In this regard, `ritest` is an important extension to existing software because it provides the option to specify virtually any form of randomization process.

When the independent variable of interest is not a mere random binary treatment assignment, randomization inference is still possible. Of course, in such cases the resampling or permutation still has to account for all relevant aspects of the data-generating process. For example, rainfall shocks can be regarded as random draws from an observable distribution, but simple permutation of rainfall measures in a country-year panel dataset destroys the spacial correlation and the autocorrelation across time. It is crucial to take all relevant features of the dependent variable of interest into account. Otherwise, permutation test results are rendered meaningless. Randomly permuting years, but not countries, would allow the user to keep the spacial correlation intact but destroy the autocorrelation. The derived distribution for a test statistic would hence be the distribution under the assumptions of i) no effect of rain and ii) rainfall not being autocorrelated across years, which may be justifiable in some contexts.

2.2 From distributions of estimates to p-values

In the last step, one must assess whether the sample realization of the chosen test statistic, $\widehat{\tau}$, is in line with its distribution under the null hypothesis, F^{τ}, which is obtained through resampling or permutation. This is often done using the rank or the rank of the absolute value,[3]

$$\mathrm{rk} = \sum_{m=1}^{M} \mathbb{1}(\mathring{\tau}_m \leq \widehat{\tau}) \quad \text{or} \quad \mathrm{rk}^{\mathrm{abs}} = \sum_{m=1}^{M} \mathbb{1}(|\mathring{\tau}_m| \geq |\widehat{\tau}|)$$

where $\mathring{\tau}_m$ are the M independent and identically distributed (i.i.d.) draws from F^{τ}.

For left- and right-sided tests, the p-value is straightforwardly computed as

$$p^{\mathrm{right}} = 1 - \frac{1}{M}\mathrm{rk} \quad \text{and} \quad p^{\mathrm{left}} = \frac{1}{M}\mathrm{rk}$$

In a two-sided test, the rank of the absolute test statistic is used to compute the p-value:

$$p^{\text{two-sided}} = \frac{1}{M}\mathrm{rk}^{\mathrm{abs}}$$

If the null hypothesis is true, $(\mathring{\tau}_m)_{m=1,\ldots,M}$ and $\widehat{\tau}$ are i.i.d. random draws from the same distribution, which implies that the resulting rank and hence the p-value are uniformly distributed.[4]

2.3 Multiple hypotheses testing

Lastly, randomization inference provides a way to improve on joint and multiple hypotheses testing methods. When a researcher conducts more than one statistical test, the expected rate of obtaining at least one false positive result is not bound by the nominal significance level of the tests and grows with the number of tests conducted. In classical statistics, Bonferoni corrections are most commonly used to gauge these so-called familywise error rates. Randomization tests can be very useful in this context, because when multiple tests are conducted at once, they also produce information about the relatedness of the distribution of test statistics.

Methods for this were developed by Romano and Wolf (2005) and developed further by Young (2016). It is possible to conduct the permutations computing two or more statistics simultaneously, for example, TE on multiple outcomes. The permutations

[3] Using the rank as a p-value is potentially misleading if the distribution is multimodal. For the most commonly used cases, the rank will suffice and is useful to assess whether the realized test statistic is extreme.

[4] Young (2016) provides a proof of the uniformity of the relative rank irrespective of distribution assumptions or sample sizes in the online appendix. His version is slightly different with regard to how ties are treated. In keeping with Stata's implementation of `permute`, `ritest` treats ties always in favor of the sample realization being less extreme, so that the p-values tend to be marginally more conservative than uniform. This effect is marginal in most applications and becomes visible in the simulations in section 4.2.

then reveal the joint distribution of the statistics under the null hypothesis. When one uses this information, the power of joint tests can be improved compared with traditional methods. Bonferroni-type corrections make conservative assumptions in lieu of the means to assess the relatedness of tests. Romano and Wolf (2005) suggest a stepwise procedure applied to the joint distribution of t statistics computed from rerandomization. Young (2016) builds upon their approach but suggests using the joint distribution of p-values computed from rerandomization to avoid issues arising from varying distributions across t statistics. The focus of this article is not on the joint tests, but `ritest` allows the user to store joint estimates to infer joint distributions and conduct these joint tests, as illustrated in the example in section 4.3.

3 The ritest command

The `ritest` source code is based on the code for `permute`. `ritest` provides an additional set of features to mimic a wide array of rerandomization methods. In section 3.1, `ritest`'s syntax is defined. The accompanying help file contains a more detailed description of all options. Section 3.2 explains available permutation and resampling methods, as well as the syntax to invoke them.

3.1 Syntax

The general structure of `ritest` is

`ritest` *resampvar* *exp_list* [, *options*]: *command*

resampvar is the name of a variable to be resampled, for example, a treatment indicator variable `treatment`. *exp_list* is a list of expressions to be collected in each step and to be compared with the realization in the main estimation sample, for example, a regression coefficient `_b[treatment]`. *options* specify the details of the resampling process, the computation of p-values, or the output. *command* is the program to be executed with each replication, for example, a regression such as `regress testscore treatment age`.

A typical example would be

```
ritest treatment _b[treatment], cluster(class) strata(school): ///
   regress testscore treatment age
```

options	Description
Main	
reps(#)	perform # random permutations; default is reps(100)
left | right	compute one-sided *p*-values; default is two-sided
Automatic resampling	
strata(*varlist*)	permute *resampvar* within strata
cluster(*varlist*)	keep *resampvar* constant within clusters
File-based resampling	
samplingsourcefile(*filename*)	take permutations of *resampvar* from Stata data file *filename*, containing variables named *resampvar*1, *resampvar*2, *resampvar*3, …
samplingmatchvar(*varlist*)	merge permutations in samplingsourcefile() with the data using these variables (1:1 or m:1)
Program-based resampling	
samplingprogram(*programname*)	generate permutations of *resampvar* by calling user-written program *programname*
samplingprogramoptions(*string*)	optionally pass *string* as options to *programname*
Reporting	
level(#)	set confidence level; default is level(95)
verbose	display full table legend
nodots	suppress replication dots
noisily	display any output from *command*
Advanced	
null(*outcome value*)	specify a null hypothesis of TE *value* or *varname* for outcome *outcome*
kdensityplot	plot the densities of each statistic in *exp_list*
kdensityoptions(*string*)	additional options to be passed on to kdensity
saveresampling(*filename*)	save all permutations of *resampvar* in a file called *filename* for later inspection
noanalytics	do not send anonymized usage statistics to Google Analytics
seed(#)	set random-number seed to #
eps(#)	numerical tolerance; seldom used
force	force ritest to accept weights in *command*

3.2 Description of resampling methods

Unlike `permute`, `ritest` allows one to specify more complex resampling structures in three different ways: i) resampling can be done automatically by permutation if the researcher specifies one or both of the `cluster(`*varlist*`)` and `strata(`*varlist*`)` options; ii) alternative assignments of the resampling variable can be read from an external file; or iii) the researcher can supply the name of a program that executes the resampling. Below I provide sample syntax for each of these cases along with a short description of their functionality. First, I give a brief overview of how the resampling variable and test statistics can be specified.

Specifying the resampling variable and test statistics

The first argument passed to `ritest` is the variable to be permuted. In the examples below, the variable is always a binary treatment indicator, but in principle, it can be any type of variable. The subsequent arguments passed to `ritest` are the expressions to be evaluated for each resampling iteration. If the command of interest is `regress` with a treatment dummy variable, specifying the expression `_b[treatment]` will correspond to using the coefficient estimate as a randomization statistic. Similarly, one can specify composite expressions, such as `_b[treatment]/_se[treatment]`, which corresponds to using the t statistic.

Automatic permutation

```
ritest resampvar exp_list, reps(#) strata(varlist) cluster(varlist): command
```

This call randomly permutes the values in *resampvar* # times, respecting `strata()` and `cluster()`, each time executing *command* and collecting the realized values for the expressions in *exp_list*. Not specifying strata assumes no strata were used, which is equivalent to all observations being in one single stratum. Not specifying clusters assumes no clusters were used, which is equivalent to each observation being a separate cluster.

This is the easiest but also the most restrictive way to define the resampling. All aspects of the original sampling are simply inferred from the specified strata and cluster variables as well as the observed distribution of the resampling variable. On the cluster level, the distribution of the resampling variable will remain unchanged. For example, if the original assignment involved 2 strata, of which the first stratum had 1 of 10 clusters treated and the second had 9 of 10 clusters treated, the distribution will stay the same in all resampling iterations. However, on the individual level, the distribution of treatment can change, for example, if clusters vary in size (for instance, because a large treated cluster and a small control cluster switch states).

Note that only realizations of the resampling variable that exist in the data are used for the permutation. For continuous variables, for example, this can be a strong restriction on the rerandomization process. If known, one should use the original distribution of the resampling variable with one of the two following methods to avoid this.

File-based resampling

```
ritest resampvar exp_list, reps(#) samplingsourcefile(filename) ///
    samplingmatchvar(varlist): command
```

This call merges the data # times using the file specified in `samplingsourcefile()` based on the ID variables specified in `samplingmatchvar()` (1:1 or, if IDs are not unique, m:1). *command* is executed each time, replacing *resampvar* iteratively with *resampvar*1, *resampvar*2, *resampvar*3, ... `samplingsourcefile()` must be created manually prior to executing `ritest`.

Program-based resampling

```
ritest resampvar exp_list, reps(#) samplingprogram(progname) ///
    samplingprogramoptions(string): command
```

This call redraws *resampvar* by executing `samplingprogram()` while simultaneously passing `samplingprogramoptions()` as options to it. This is the most versatile method and allows for many applications.[5] If the original randomization of an experiment was carried out with a do-file, the original code can be used to define program used in `samplingprogram()`. Beyond `samplingprogramoptions()`, two more arguments are passed to the program: the variable name of the resampling variable `resampvar(`*varname*`)` and an integer containing the count index of the current iteration `run(`*integer*`)`. In principle, the program may also alter characteristics of the data other than just the resampling variable. For example, it could resample more than one variable at once. The count index of the current iteration can be used to enumerate the full set of possible permutations as done below in section 4.1.

4 Applications and examples

In this section, I present three simple applications with different focuses. In section 4.1, I replicate results from Fujiwara and Wantchekon (2013), who already consistently use randomization inference in a dataset with few units of treatment assignment. The original analysis was done with preexisting software packages. I add to their work by showing how to obtain nonapproximate *p*-values and how to analyze the distributions of average TE estimators under the null hypothesis. I do this to illustrate that `ritest` produces the same results as existing Stata commands in cases where existing software is applicable but that it can be used to go beyond what `permute` can do. Section 4.2 uses simulated data to illustrate randomization inference in a sample with clustered

[5] In fact, the other methods are implemented by calling this method with prespecified programs.

treatment assignment and varying cluster sizes—a setting where traditional methods have proven unreliable (MacKinnon and Webb 2017). Section 4.3 also uses simulated data to briefly touch upon the usefulness of randomization inference in the context of joint tests of related hypotheses.

4.1 Clientelism in Benin

Among the few articles that consistently use randomization inference is Fujiwara and Wantchekon's 2013 study of political clientelism in Benin. They studied village-level aggregates of survey data from 24 villages, half of which received a randomized treatment. The village-level treatment involved a change in how candidates campaigned there during a presidential election. Treatment involved holding meetings to discuss policy as opposed to the prevalent clientelist rallies held by candidates in other villages. The authors' findings indicate that these meetings reduce clientelism overall and decrease the chances of the candidate with a political stronghold in the community.

The original sample covers 24 villages, which is quite small. Furthermore, treatment assignment was stratified by district, such that the 24 villages belonged to 12 pairs. In each pair, one village randomly received treatment. This complicates matters for classical inference and would imply that it is necessary to control for pair-fixed effects (Duflo, Glennerster, and Kremer 2008) to get the standard errors correct. Thus 24 observations would be used to estimate the 12 intercepts and a TE. In no way does this setup allow the researchers to assume that asymptotic results would hold. Thus, inference based on asymptotics would be weak.

The fairly simple randomization structure (stratified but not clustered treatment assignment and binary treatment) allows the authors to conduct randomization inference using Stata's `permute` command. Their p-values are based on 1,000 draws from the distribution of their TE estimate under the sharp null hypothesis of no effect. My replication starts off with an exact replication of their results using `ritest` but goes further by computing precise p-values instead of approximated ones and by studying the shapes of the distributions of TE estimates.

Table 1 shows the results of the replication. The upper panel contains the pretreatment balance tests from their table 1, and the lower panel shows the TE estimates on their outcomes, equivalent to the first panel of their table 2. Columns 1–4 are equivalent to what they publish in their article based on `permute`. Column 5 contains a replication of column 4 using `ritest`. Differences between column 4 and 5 are solely due to the two being different realizations of the same random process.

Table 1. Replication (columns 1–4) and extension (columns 5–6) of panels A in tables 1 and 2 from Fujiwara and Wantchekon (2013)

					Randomization	Randomization
				Randomization	inference	inference
				inference	*p*-value	*p*-value
	Control		Standard	*p*-value	`ritest`	`ritest`
	mean	Treatment−control	error	`permute`	(approx.)	(precise)
	(1)	(2)	(3)	(4)	(5)	(6)
Female	0.50	0.01	0.01	0.169	0.173	0.164
Age (in years)	41.71	0.39	1.13	0.743	0.748	0.740
Fon ethnicity	0.50	−0.01	0.01	0.210	0.207	0.219
Yoruba ethnicity	0.14	0.02	0.01	0.032	0.033	0.031
French speaker	0.27	−0.01	0.03	0.768	0.789	0.766
Fon speaker	0.53	0.04	0.02	0.046	0.035	0.043
Christian	0.49	0.10	0.05	0.058	0.054	0.059
Muslim	0.22	−0.06	0.04	0.171	0.191	0.172
Primary schooling	0.24	0.02	0.03	0.507	0.502	0.492
Secondary schooling or higher	0.09	0.03	0.01	0.082	0.089	0.079
Single	0.03	0.02	0.01	0.052	0.064	0.054
Married (monogamous)	0.52	0.01	0.04	0.860	0.833	0.860
Married (polygamous)	0.35	−0.04	0.04	0.405	0.393	0.406
Has regular income	0.41	−0.03	0.03	0.424	0.430	0.424
Owns farm	0.75	−0.08	0.06	0.291	0.309	0.286
Electrical lighting at home	0.05	0.04	0.03	0.262	0.215	0.250
Member of Assoc./NGO	0.36	−0.00	0.04	0.929	0.945	0.936
Clientelism index	0.00	−0.23	0.08	0.024	0.012	0.020
Clientelism index, excl. vote buying	0.00	−0.22	0.10	0.049	0.043	0.045
Vote buying	0.22	−0.04	0.03	0.166	0.152	0.154

Given that the authors' sample consists of a mere 12 strata in which always either the first or the second observation is treated, there exist only $2^{12} = 4096$ possible permutations of treatment that are all equally likely to occur. These can easily be enumerated. Thus there is no real reason to use an approximation, aside from `permute` not allowing anything but i.i.d. reshuffling of treatment. Column 6 computes the precise p-value of the test by enumerating all 4,096 possible permutations. The results do not differ strongly from the approximations in columns 4 and 5. This result is unsurprising, given that they sample (with replacement) 1,000 of 4,096 treatment assignments. In general, computing nonapproximate p-values with `ritest` can be as easy as approximating them and should hence be preferred.

Analyzing the distribution of parameter estimates under the null hypothesis

Often, it is informative to study the distribution of an estimator under the null hypothesis. Density plots of the results from the replications can be created by passing the `kdensityplot` option to `ritest`. Figure 1 shows such plots for two of the balance tests in table 1. For illustration, a very balanced variable (age) and a fairly unbalanced variable (whether the respondent speaks Fon) are both used here. The vertical lines represent the parameter estimates in the actual sample.

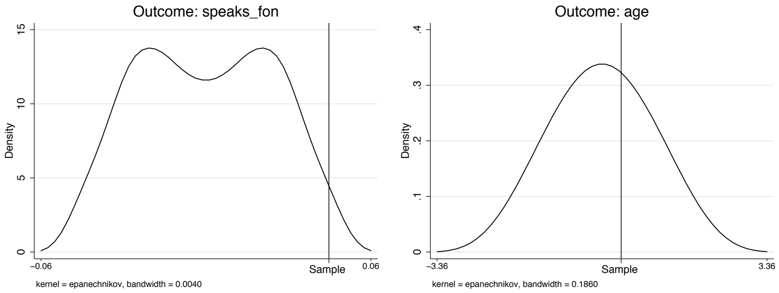

Figure 1. Densities of estimates under the null hypothesis obtained through resampling. The vertical line indicates the location of the estimate under the implemented treatment assignment.

One may want to study the joint distribution of several parameter estimates under the null hypothesis. Young (2016) and Romano and Wolf (2005) propose variations of tests that correct for multiple hypothesis testing if several parameters are tested. To this end, the joint distributions of multiple estimates under the null hypothesis are needed. When the `saving(`*filename*`)` option is used, `ritest` stores a copy of the estimates. This facilitates the study of joint distributions. A simple example of this is given in figure 2. Two of the outcomes studied by Fujiwara and Wantchekon (2013) are the clientelism index (excluding vote buying) and the vote buying measure. Considering the p-values in column 6 of table 1, we see that the former is just marginally significant on a 5% level and that the latter has a p-value of 0.154. That is, both are somewhat marginal in their distributions under the null hypothesis but not extreme. In figure 2, their joint distributions under the null hypothesis are depicted. We can see that the estimate under the true treatment allocation is much more extreme than revealed when solely considering the marginal distribution for each estimand separately. The point cloud in figure 2 also indicates that the two tests are fairly independent. A case where this is different will be considered in section 4.3.

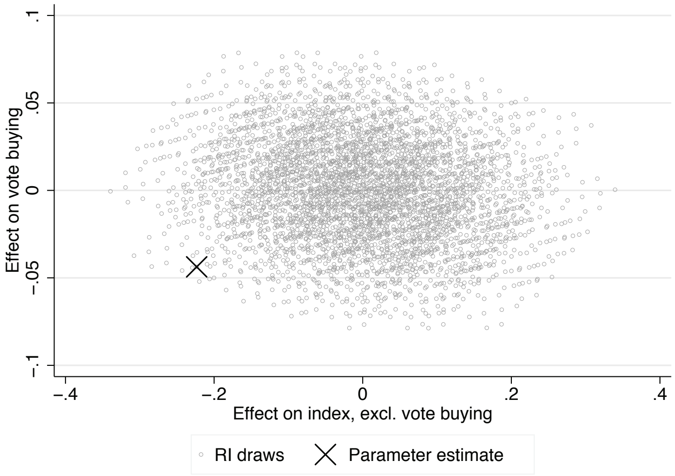

Figure 2. Joint distribution of two TE estimators under the null hypothesis

Implementation of the preceding analysis in Stata

This section describes how to implement the preceding analysis in Stata with `ritest`. To compute the results in table 1, the loop in the code fragment below iterates over all variables and computes the mean in the control group (line 4 in the code below), the TE estimate (line 5), the randomization inference p-values as in the article (line 6), and the approximate as well as the accurate p-values using `ritest` (lines 7 and 8).

```
1    use survey_data_AEJ.dta
2    generate observation_id = _n
3    foreach i in female age ethn_fon ethn_yoruba speaks_french speaks_fon    ///
       christian islam primary_schooling secondary_schooling single monog     ///
       polyg reg_income has_farm electri member index index_nocash cash {
4         summarize `i´ if treat==0
5         areg `i´ treat, robust absorb(depcom)
6         permute treat _b[treat], strata(depcom) reps(1000) seed(0) nodots:  ///
            regress `i´ treat dcom2-dcom12
7         ritest treat _b[treat], strata(depcom) reps(1000) seed(0) nodots:   ///
            regress `i´ treat dcom2-dcom12
8         ritest treat _b[treat], samplingprogram(enumerate_permutations)     ///
            samplingprogramoptions("stratavar(depcom) obsid(observation_id)") ///
            r(4096) kdensityplot nodots: regress `i´ treat dcom2-dcom12
     }
```

Line 8 calls a program that has to be defined before its use. To compute nonapproximate p-values, the user needs a program that returns a distinct treatment assignment vector for each integer passed inside the argument `run()`. Treatment assignment was stratified by 12 districts, where each district contained 2 villages. Hence, each stratum can take on two states: “high”, if the first of the two villages is being treated, and “low”, if the second village is treated. The program converts the passed integer into a 12-digit binary variable, then uses each digit to determine treatment allocation in the corresponding district. If the ith digit takes on the value zero, stratum i is in state low, and the second village is assigned to treatment. Conversely, if the ith digit takes on the value one, the first village is assigned to treatment because i is in state high. For example, `index` = 2057 yields assignment 100,000,001,001, which implies that in districts 1, 9, and 12, the first village receives treatment, while in all other districts the second village receives treatment. The program code is as follows:

```
program enumerate_permutations
   syntax, resampvar(varname) stratavar(varname) obsid(varname) run(integer) *
   local index=`run´-1 //which assignment is picked
   sort `stratavar´ `obsid´, stable
   mata: st_strscalar("assignment",(inbase(2,`index´)))
   local av = string(real(assignment),"%0`: di _N/2´.0f")
   // av is now a string of 0s and 1s where a 1 [0] at position X indicates   ///
      that the Xth strata has the first [second] observation treated
   levelsof `stratavar´, local(strata)
   local counter = 0
   foreach statum of local strata {
       by `stratavar´: replace `resampvar´=real(substr("`av´",`++counter´,1)) ///
            if `stratavar´==`statum´ & _n==1
       by `stratavar´: replace `resampvar´=1-real(substr("`av´",`counter´,1)) ///
            if `stratavar´==`statum´ & _n==2
   }
end
```

4.2 Simulations with varying cluster sizes

MacKinnon and Webb (2017) illustrate how classical cluster–robust inference can fail, even if there are many clusters and the “rule of 42 [clusters]” (Angrist and Pischke 2009) is satisfied. This can happen if i) clusters vary in size or if ii) the numbers of treated clusters and untreated clusters are very different. The authors study this problem and provide evidence that the Wild bootstrap can allow for valid inference under such circumstances, at least if the treated-untreated imbalance does not become too severe. Randomization inference is equally suited in this situation because it does not rely on any assumptions about the cluster sizes or the ratio of treated to untreated clusters. In this simulation, I mimic a dataset of students in classes of varying sizes.

If the unit of measurement is the student but the unit of assignment is the class, treatment will ex-post be correlated with class size.[6] In natural clusters (classes, countries, villages), a correlation between cluster sizes and other—potentially unobservable—

[6] This is in spite of the correlation being zero in expectation before assignment. Of course, the direction of the correlation is equally likely to turn out to be positive or negative. To see this, I recommend simulating a dataset as described in this subsection.

characteristics is very likely to occur. Thus varying cluster sizes result in correlations between treatment and unobservables, even if treatment assignment is i.i.d. on the cluster level.

In the following simulation, 42 classes are sorted into a treatment and a control group. There exists an unobservable class characteristic $\varepsilon^c \sim N(0,1)$. Class size X^c is related to this characteristic in the following manner:

$$X^c = \begin{cases} 20 & \text{if } \varepsilon^c \leq 1.5 \\ 100 & \text{if } \varepsilon^c > 1.5 \end{cases}$$

Treatment is assigned according to two similar regimes. In one version of the simulation, every second class receives treatment, whereas the other classes are assigned to the control group. In the other version, only 10 classes receive treatment. Two outcome variables are used, one with and one without a TE (as indicated by the superscript), to assess size as well as power. The two outcomes are distributed as

$$\begin{aligned} y_i^{\text{no TE}} &\sim N(\varepsilon_i^c, 0) \\ y_i^{\text{TE}} &\sim N(\varepsilon_i^c + \tau_i^c, 0) \end{aligned}$$

where τ_i^c is an indicator of the treatment status of class c.

Figure 3 shows the distributions of p-values from tests of the null hypothesis of no TE in various versions. The upper panel shows the distributions of p-values if the outcome without the TE is used. The gray bars correspond to the situation where treatment and control groups contain the same number of clusters, and the black outlined bars show the distribution when the treatment group contains fewer clusters than the control group. The first two histograms show that randomization inference produces well-calibrated tests, irrespective of whether the coefficient or the t statistic is used. Standard cluster–robust inference produces highly oversized tests. This effect increases when treated clusters become few in number. The wild bootstrap suffers less from varying cluster sizes, especially the restricted version (compare MacKinnon and Webb [2017]). Yet, if treated clusters are few, the wild bootstrap also over-rejects.

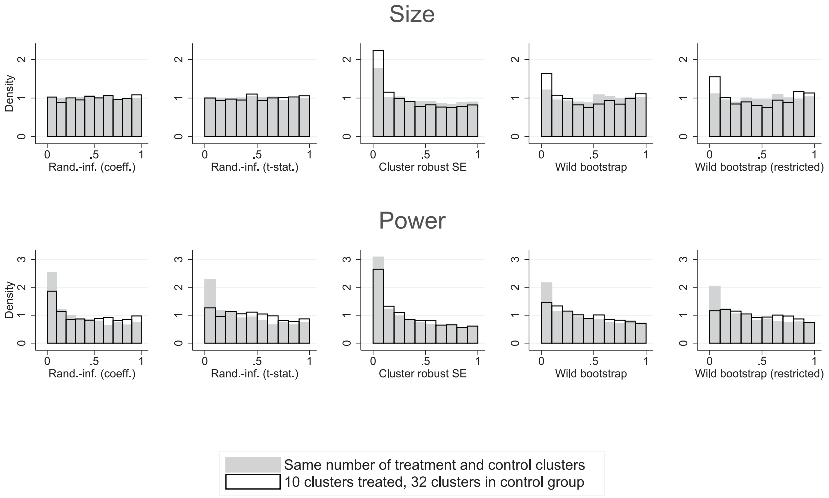

Figure 3. Distribution of p-values based on different testing methods

To illustrate statistical power, the lower half of figure 3 shows the same tests but uses the outcome in which a constant TE was simulated. For this particular data-generating process, randomization inference—in both variants—has more power than either the restricted or the unrestricted wild bootstrap, despite the bootstraps being oversized.

4.3 Simulation with two strongly related tests

Assume TEs on two related outcomes are studied. It might not be possible to detect a significant TE if the two tests are considered separately. When studied jointly, however, it becomes apparent that the combination of test results is very unlikely to occur under the null hypothesis. Consider the simple case of two normal outcomes and a constant average TE. Both outcomes are related, meaning they are affected by the same unobservable noise term.

The data are generated as follows:

```
set obs 24                              // sample size
generate x  = 2*rnormal()              // unobservable
generate t  = round(runiform())        // treatment
generate y1 =   x + t  + rnormal()     // first outcome
generate y2 = - x + t  + rnormal()     // second outcome
```

When one estimates the TE for either outcome, detecting treatment is difficult because power is low. Figure 4 shows the distribution of estimates under the null hypothesis in contrast with the actual estimate in 1 sample of 24 observations. While the location of the actual estimates is not extreme in either of the marginal distributions, it is clearly an outlier in the joint distribution.

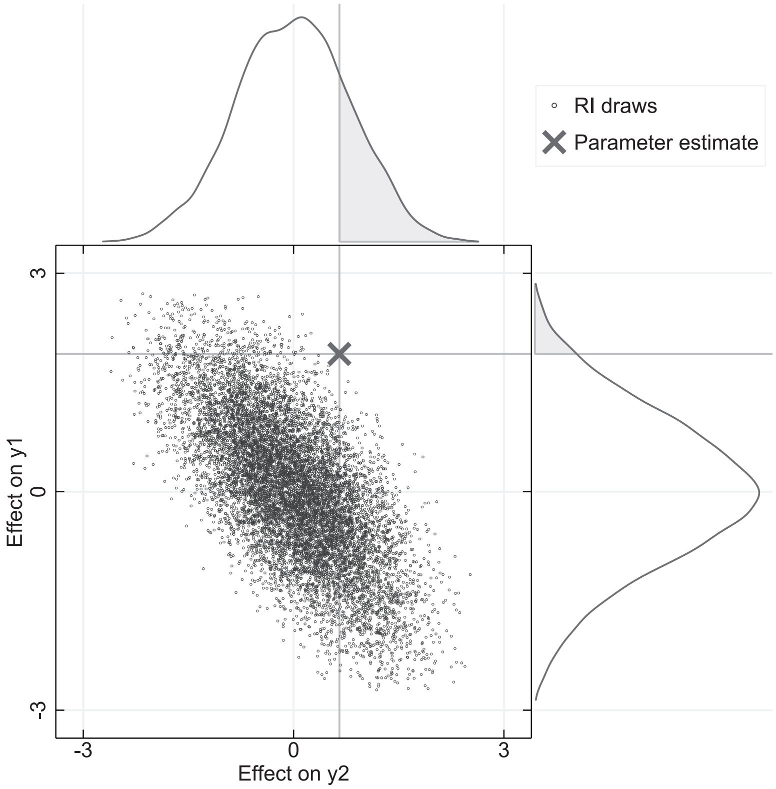

Figure 4. Joint distribution of two TE estimators under the null hypothesis versus the actual estimate

In this example, randomization inference provides a researcher with the correlation structure of the two tests. This information can be used to derive further tests, either by approximating and parameterizing the joint distribution under the null hypothesis and assessing whether the realization falls into the tails of it or by picking an alternative outcome measure that incorporates information from both dimensions. This could even be done without or before observing the actual realization where data mining is a concern. These can then be used to derive joint tests or to account for multiple hypothesis testing as proposed by Romano and Wolf (2005) or Young (2016).

Implementation of the preceding analysis in Stata

Because `ritest` evaluates the return value of a single command, regressions for both TEs have to be estimated by a single e-class program call. To do so, I define a simple program that executes two commands and returns a particular estimate from both:

```
program two_estimates, eclass
    syntax, command1(string) command2(string) estimate(string)
    `command1´
    local first=`estimate´
    `command2´
    estadd scalar first=`first´
    estadd scalar second=`estimate´
end
```

Endowed with this wrapper-command, ritest can now be instructed to save both estimates from each permutation. This file can be read to plot and study the joint distribution of estimates as in figure 4:

```
tempfile tmp
ritest t e(first) e(second), saving(`tmp´) r(10000) seed(0): two_estimates, ///
    command1(regress y1 t) command2(regress y2 t) estimate(_b[t])
use `tmp´, clear
histogram _pm_1
histogram _pm_2
scatter _pm_1 _pm_2
```

5 Conclusion and summary

In this article, I illustrated the usefulness of randomization inference for TE analysis. Using examples with real and simulated data, I presented and discussed various cases in which randomization inference facilitates consistent analysis or provides additional insights. The examples covered a range of data-generating processes commonly encountered in experimental setups.

All the analyses are carried out using the ritest command, which is available in the supplementary materials to this article. ritest provides a battery of options to compute p-values suitable for different tests and different randomization procedures.

6 References

Angrist, J. D., and J.-S. Pischke. 2009. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton, NJ: Princeton University Press.

Aronow, P. M., and C. Samii. 2012. *ri: R package for performing randomization-based inference for experiments*. R package version 0.9. https://cran.r-project.org/package=ri.

Bloom, E., E. King, I. Bhushan, M. Kremer, D. Clingingsmith, B. Loevinsohn, R. Hong, and J. B. Schwartz. 2006. Contracting for health: Evidence from Cambodia. Report, The Brookings Institution. https://www.brookings.edu/wp-content/uploads/2016/06/20060720cambodia.pdf.

Bowers, J., M. Fredrickson, and B. Hansen. 2016. *RItools: Randomization inference tools*. R package version 0.1-15. https://cran.r-project.org/package=RItools.

Bruhn, M., and D. McKenzie. 2009. In pursuit of balance: Randomization in practice in development field experiments. *American Economic Journal: Applied Economics* 1: 200–232.

Bugni, F. A., I. A. Canay, and A. M. Shaikh. 2016. Inference under covariate-adaptive randomization. Working paper, Northwestern University.

Cameron, A. C., and D. L. Miller. 2015. A practitioner's guide to cluster–robust inference. *Journal of Human Resources* 50: 317–372.

Cattaneo, M. D., B. R. Frandsen, and R. Titiunik. 2015. Randomization inference in the regression discontinuity design: An application to party advantages in the U.S. Senate. *Journal of Causal Inference* 3: 1–24.

Cattaneo, M. D., R. Titiunik, and G. Vazquez-Bare. 2016. Inference in regression discontinuity designs under local randomization. *Stata Journal* 16: 331–367.

———. Forthcoming. Comparing inference approaches for RD designs: A reexamination of the effect of head start on child mortality. *Journal of Policy Analysis and Management*.

Cohen, J., and P. Dupas. 2010. Free distribution or cost-sharing? Evidence from a randomized Malaria prevention experiment. *Quarterly Journal of Economics* 125: 1–45.

Duflo, E., R. Glennerster, and M. Kremer. 2008. Using randomization in development economics research: A toolkit. In *Handbook of Development Economics*, ed. T. P. Schultz and J. A. Strauss, vol. 4, 3895–3962. Amsterdam: Elsevier.

Fisher, R. A. 1935. *The Design of Experiments*. Edinburgh: Oliver & Boyd.

Fujiwara, T., and L. Wantchekon. 2013. Can informed public deliberation overcome clientelism? Experimental evidence from Benin. *American Economic Journal: Applied Economics* 5: 241–255.

Ganong, P., and S. Jäger. 2015. A permutation test for the regression kink design. Working paper, Harvard University.

Ichino, N., and M. Schündeln. 2012. Deterring or displacing electoral irregularities? Spillover effects of observers in a randomized field experiment in Ghana. *Journal of Politics* 74: 292–307.

Imbens, G. W., and D. B. Rubin. 2015. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. New York: Cambridge University Press.

Leamer, E. E. 2010. Tantalus on the road to asymptopia. *Journal of Economic Perspectives* 24: 31–46.

MacKinnon, J. G., and M. D. Webb. 2016. Randomization inference for difference-in-differences with few treated clusters. Queen's University, Department of Economics, Working Paper No. 1355. https://ideas.repec.org/p/qed/wpaper/1355.html.

———. 2017. Wild bootstrap inference for wildly different cluster sizes. *Journal of Applied Econometrics* 32: 233–254.

Romano, J. P., and M. Wolf. 2005. Exact and approximate stepdown methods for multiple hypothesis testing. *Journal of the American Statistical Association* 100: 94–108.

Rosenbaum, P. R. 2002. *Observational Studies*. 2nd ed. New York: Springer.

———. 2010. *Design of Observational Studies*. New York: Springer.

Rosenberger, W. F., and J. M. Lachin. 2016. *Randomization in Clinical Trials: Theory and Practice*. 2nd ed. Hoboken, NJ: Wiley.

Young, A. 2016. Channeling Fisher: Randomization tests and the statistical insignificance of seemingly significant experimental results. Working paper, MIT.

About the author

Simon Heß is a PhD student in economics at Goethe University Frankfurt, Germany. His primary research interests are causal inference in the context of development economics and social network analysis. Before starting on his PhD, he studied and worked in Egypt, Ghana, and the UK. His research focuses on Africa and the Middle East.