Randomization inference with Stata: A guide and software

Simon Heß

March 21, 2017

Accepted draft, see stata-journal.com/article.html?article=st0489 for final version

Abstract

Randomization inference or permutation tests are only sporadically used in economics and other social sciencesthis 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: 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 methods most commonly used to study data obtained through RCTs are well-established econometric methods. Most of these methods are geared towards making inference with large samples drawn from infinite populations, relying on asymptotic properties of estimators and generally make strong model assumptions. As Young (2016) with reference to Leamer (2010) points out, data obtained from RCTs at times does not meet the requirements to be able to rely on asymptotic properties. However, the use of randomization in the experimental design enables researchers to make use of the powerful tool of randomization inference, as introduced by Fisher (1935), further developed by Rosenbaum (2002), and recently used in a number of high-quality papers in economics and political science (including but not limited to Bloom *et al.*, 2006; Cohen and Dupas, 2010; Fujiwara and Wantchekon, 2013; Ichino and Schündeln, 2012). In RCTs the experimenter usually knows exactly how the randomization was carried out and randomization inference uses this knowledge of the randomization process to assess whether observed outcomes in a given sample are likely to have been observed by chance even if treatment had no effect. There

exists also a growing literature applying randomization inference to data obtained outside the realm of RCTs. For example, Cattaneo *et al.* (2015, 2017) use randomization inference in the context of regression discontinuity designs and Ganong and Jäger (2015) in the context of regression kink designs.

For a more detailed primer in permutation tests the reader is referred to the books by Rosenbaum (2010) or Imbens and Rubin (2015) who provide detailed introductions. The intuition behind randomization inference is fairly straightforward. Consider a data generating process which involves a random draw from a known distribution, e.g. 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 are known too. These distributions can be numerically obtained through Monte Carlo methods, by computing the desired statistics over and over again 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 the 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, i.e. when basing inference on asymptotic variance estimators, the estimation sample is assumed to consist of random draws from an infinite population about which the researcher wants to learn. This is of course only one of several ways to conceptualize experiments. Under randomization inference the experimental sample is taken as fixed. The only aspect which is assumed to be random is the assignment to either arm of the treatment—or whatever other variable is of interest.

As a first step in conducting randomization inference it is required to define 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. Since randomization inference is not built on and does not assume random sampling from the population, it does not rely on large-sample theory to apply. It merely requires knowledge of the randomization process. Further, while the statistic studied may be based on a model—such as average treatment effect (ATE) estimates from regressions with control variables—the validity of the carried out inference depends in no way 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.

The use of Fisherian randomization inference for TE analysis in the social sciences is still limited to a few select papers and is often banned into footnotes or appendix tables of robustness checks. Young (2016) re-visited results of almost 2000 regressions from more than 50 papers 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 percent of significant tests do not remain significant at the same level. More than 30 percent 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 of the experimental papers are able to maintain the hypothesis that any TE exists at all.

The program accompanying this paper provides an easy to use yet versatile toolkit for randomization inference in Stata. As already pointed out by Bruhn and McKenzie (2009), conducting randomization inference currently requires nontrivial programming skills from the researcher. Where researchers are seen using randomization inference, they often write their own codes, which introduces unnecessary additional risk of coding errors. When .do-files for conducting randomization inference tests are published alongside research papers, 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 this paper serving as a guideline and the provided Stata command will be useful. I argue that more researchers would use randomization inference if standard statistical software would endow them with the necessary means and introduce the Stata program `ritest` for this purpose. Notable in this context are the R-packages `RItools` (Bowers *et al.*, 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 build-in `permute` allows only to permute individual observations, allowing to specify strata, but fails when more complicated treatment assignments—e.g. clustered assignment—are to be modeled. `rdlocrand` by Cattaneo *et al.* (2016) is a comprehensive Stata package, providing the means to conduct randomization inference specifically in the context of regression discontinuity designs

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

under local randomization.

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

The rest of this paper is structured as follows. Section 2 presents the theoretical background for randomization inference in Stata. The command and its syntax is introduced in Section 3. 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 the 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.

Since 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. For the purpose of this exposition I will make no distinction and use these terms interchangeably.

There is flexibility in choosing the test statistic used in randomization inference. In the cases of 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 to study 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 it was shown that basing randomization inference on the t-statistics as opposed to the coefficient estimates can be beneficial. MacKinnon and Webb (2016a) discuss cases where the two approaches yield different results and Young (2016) also compares both methods. Different statistics will differ with regards to against which alternative hypotheses they have 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:

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

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 is different from the null hypothesis of no *average* treatment effect 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, it suffices to compute the test statistic for *each possible permutation* of the treatment vector. Since the set of possible permutations can get excessively large, it is common practice to only use several thousand random draws for the treatment vector instead.

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

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. Tests for this can even be conducted using Stata's `permute`-command. However, more and more lab and field experiments use 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.) simply because there is no other way or in an attempt to contain spillover effects. More sophisticated experimental designs may feature multilevel assignment procedures or multidimensional treatments.

This poses a difficulty for the computation of standard errors, and hence for testing hypotheses. In the case of treatment assignment in clusters, this follows from the fact that regression model errors will not be independent within clusters. The reason for this being that outcome variables typically have non-zero intra-cluster correlation and by assigning treatment jointly to all observations within clusters, 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 effect 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. If a single school class benefits from unobservable teacher quality and this class falls into the treatment group, this unobserved variable will effect 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 *et al.* (2016) similar to Bruhn and McKenzie (2009) show that when inference neglects the stratification, t-tests tend to be overly conservative. This result extends to other covariate balancing methods such as pairwise randomization. In pairwise randomization, 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 sufficiently many clusters were used, clustered treatment assignment can be accounted for by using cluster-robust standard errors (Cameron and Miller, 2015). Similarly, the problems that stratification poses for standard error estimation can be solved by using strata fixed effects (Bruhn and McKenzie, 2009; Bugni *et al.*, 2016; Cameron and Miller, 2015; Duflo *et al.*, 2007). In many of these cases however, the "correct" specification is not straightforward. Besides, what "large enough" samples are still is the subject of ongoing debates, and has been shown to depend 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, 2016a). Similarly, if the randomization protocol involves redrawing treatment until a certain balance criterion was met, the correct parametric specification to use is unclear (Bruhn and McKenzie, 2009).

In contrast, to compute randomization inference p-values it suffices to be able to 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, it is important to note that a resampling test which does not respect the original treatment assignment method will also produce inappropriate results (cf. Lachin, 2016, section 6.4). For example if the original process involved treatment assignment by villages, the resampling/permutation should not be based on individual-level permutations. In this regard `ritest` is an important extension to existing software as it provides the option to specify virtually any form of randomization process.

In cases where 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/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 data set 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 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 (1) no effect of rain, and (2) rainfall not being autocorrelated across years, which may be justifiable in some contexts.

2.2 From distributions of estimates to p-values

In a last step it is necessary to assess whether the sample realization of the chosen test statistic, $\hat{\tau}$, 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]

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

where $\mathring{\tau}_m$ are the M i.i.d. draws from F^{τ}.

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

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

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

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

If the null hypothesis is true, $\{\mathring{\tau}_m\}_{m=1,\dots,M}$ and $\hat{\tau}$ 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 improves means to improve on joint and multiple hypotheses testing methods. In situations where the 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 family-wise 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 further by Young (2016). It is possible to conduct the permutations computing two or more statistics simultaneously, for example TEs on multiple outcomes. Then, the permutations reveal the joint distribution of the statistics under the null hypothesis. Using this information, the power of joint tests can be improved compared to 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 through rerandomization to avoid issues arising from varying distributions across t-statistics. The focus of this guideline is not on the joint tests, but the supplied command allows storing joint estimates to infer joint distributions and conduct these joint tests, as illustrated in the example in Section 4.3.

[3] Using the rank as a p-value is potentially misleading if the distribution is multimodal. For the most common use 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 regards 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.

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, the command's syntax is defined. A more detailed description of all options is found in the accompanying Stata help-file. Section 3.2 explains the usage of available permutation and resampling methods and the syntax to invoke them.

3.1 Syntax

The general structure of a program call 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 to 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 regress testscore treatment age.

A typical example would be:

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

General options

reps(#)	perform # random permutations; default is reps(100).
left—right	compute one-sided p-values; default is two-sided.
eps(#)	numerical tolerance, default is 10^{-7}.
null(outcome value)	specify an alternative null hypothesis. value can either be numeric or the name of a variable. To conduct the test, value is deducted outcome before resampling/permutation.

Automatic resampling (permutation) options

strata(varlist)	permute resampvar within strata.
cluster(varlist)	permute resampvar by clusters.

File-based resampling options

samplingsourcefile(filename)	load permutations of resampvar from filename, containing variables resampvar1, resampvar2, ...
samplingmatchvar(varlist)	use variables varlist to merge samplingsourcefile with the data (1:1 or m:1).

Program-based resampling options

samplingprogram(programname)	resample resampvar by calling user-written program programname.
samplingprogramoptions(string)	optionally pass string as options to programname.

Reporting options

level(#)	set confidence level; default is level(95).
verbose	display full table legend.
nodots	suppress replication dots.
noisily	display any output from command.
saving(filename)	save a data set of estimates from the permutations.
kdensityplot	plot the densities of each statistic in exp_list.

3.2 Description of resampling methods

Unlike `permute`, `ritest` allows the specification of more complex resampling structures. To achieve this, there are three different ways: (i) the resampling can be done automatically by permutation if the researcher specifies one or both of the options `cluster(varlist)` and `strata(varlist)`, (ii) alternative assignments of the resampling variable can be read from an external file, or (iii) the name of a program can be supplied which executes the resampling. Below I provide sample syntax for each of these three cases alongside a short description of their functionality. First, I give a brief overview of how the resampling variable and the test statistic(s) can be specified.

Specifying the resampling variable and test statistics

The variable to be permuted is the first argument passed to `ritest`. While in the examples below this is always a binary treatment indicator, 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]` would 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 `clusters`, 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 observations 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 and 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 two strata, of which the first stratum had one of ten clusters treated and the second had nine out of ten clusters treated, this will stay the same in all resampling iterations. On the individual level the distribution of treatment can change however, for example if clusters vary in size (e.g. because a large treated cluster and a small control cluster switch states).

Note that only realizations of the resampling variable which exist in the data are used for the permutation. In the case of continuous variables, for example, this can be a strong restriction on the rerandomization process. If known, one should use the original distribution 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 `samplingsourcefile` based on `samplingmatchvar` (1:1, or if ids are not unique m:1). `command` is executed each time, replacing `resampvar` iteratively with `resampvar1`, `resampvar2`, `resampvar3`, ... `samplingsourcefile` has to 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 the program `samplingprogram`, passing `samplingprogramoptions`

as options to it. This is the most versatile method and allows for a wide range of applications.[5] Where the original randomization of an experiment was carried out with a Stata do-file, the original code can be used to define the `samplingprogram`. Beyond the `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 results are replicated from a paper which already makes consistent use of randomization inference in a data set with few units of treatment assignment. The original analysis was carried out with preexisting software packages. I add to their work by showing how non-approximate p-values can be obtained and how the distributions of ATE-estimators under the null hypothesis can be analyzed. The purpose of this is 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 treatment assignment and varying cluster sizes—a setting where traditional methods have proven unreliable (MacKinnon and Webb, 2016b). Simulated data is used again in Section 4.3 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 papers which make consistent use of randomization inference is Fujiwara and Wantchekon's 2013 study of political clientelism in Benin. Village-level aggregates of survey data from 24 villages are studied, 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 is this quite small. Further, treatment assignment was stratified by district, ssuch that the 24 villages belong to 12 pairs of which always one randomly received treatment. This complicates matters for classical inference and would imply that in order to get the standard errors correct it is necessary to control for pair-fixed effects (Duflo *et al.*, 2007). As a result, 24 observations would be used to estimate the 12 intercepts and a TE. In no way does this setup allow researchers to assume asymptotic results hold and 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`. Their p-values are based on 1000 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 estimates.

Table 1 shows the results of the replication. The upper panel contains the pre-treatment 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 paper 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.

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 which are all equally likely to occur. These can easily be enumerated. Hence, there is no real reason to use an approximation,

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

Table 1: Replication (columns 1-4) and extension (columns 5-6) of Panels A in Tables 1 and 2 from Fujiwara and Wantchekon (2013)

	Control Mean (1)	Treat.-control (2)	Standard error (3)	RI p-value permute (4)	RI p-value ritest (approx.) (5)	RI p-value ritest (precise) (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

aside from `permute` not allowing to do anything but i.i.d. reshuffling of treatment. Column 6 computes the precise p-value of the test by enumerating all 4096 possible permutations. The results do not differ strongly from the approximations in columns 4 and 5. This result is not a surprise given that they sample (with replacement) 1000 of 4096 treatment assignments. In general, computing non-approximate 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 instructive 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 option `kdensityplot` to `ritest`. Figure 1 shows such plots for two of the balance tests in Table 1. For illustrative purposes, one variable which is very balanced (age) and one which is fairly unbalanced (whether the respondent speaks Fon) are used here. The vertical lines represent the parameter estimates in the actual sample.

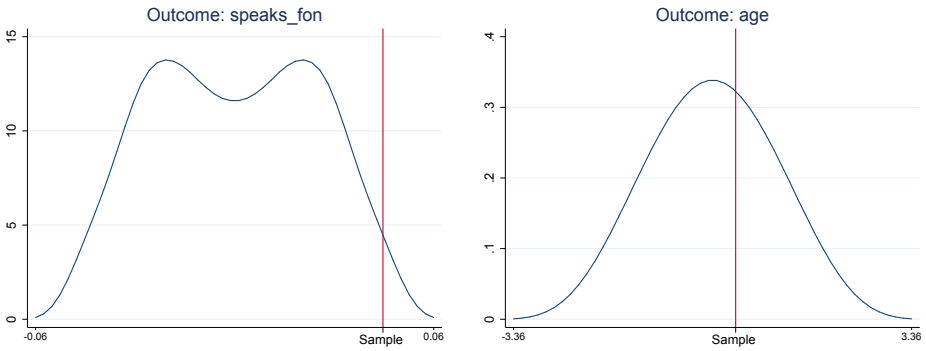

Figure 1: Densities of estimates under the null hypothesis obtained through resampling. The vertical line indicates the location of the estimate under the implemented treatment assignment.

It can be desirable to study the joint distribution of several parameter estimates under the null hypothesis. Young (2016) and Romano and Wolf (2005) propose variations of tests correcting for multiple hypothesis testing if several parameters are tested. To this end the joint distributions of multiple estimates

under the null hypothesis are needed. By passing the `saving(filename)`, `ritest` is instructed to store 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 their clientelism index (excluding vote buying) and their vote buying measure. Considering the p-values in column 6 of Table 1, the former is just marginally significant on a 5% level and 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 is depicted. It is apparent 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.

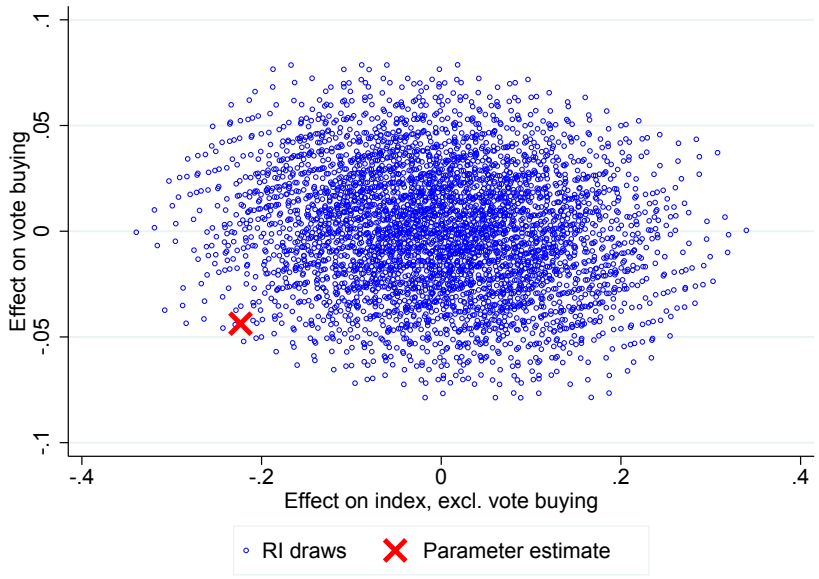

Figure 2: Joint distribution of two TE estimators under the null hypothesis

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 paper (line 6), and the approximate as well as the accurate p-value using `ritest` (line 7 and 8).

```
1  use survey_data_AEJ.dta, clear
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
9          }
```

In line 8, a program is used which has to be defined before. To compute non-approximate p-values, a program is needed which returns a distinct treatment assignment vector for each integer passed inside the argument `run()`. Treatment assignment was stratified by twelve districts, where each district contained two 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 and then uses each digit to determine treatment allocation in the corresponding district. If the i^{th} digit takes on the value zero, stratum i is in state *low*, and hence the second village is assigned to treatment. Conversely if the i^{th} digit takes on the value one, the first village is assigned to treatment, because i got *high*. For example, index=2057 yields assignment 100000001001, 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 0's and 1's where a 1 [0] at position X indicates that
            the Xst 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 (2016b) illustrate how classical cluster-robust inference can fail, even if there are many clusters and the "rule of 42 [clusters]"(Angrist and Pischke, 2008) 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 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, as 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 data set 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—characteristics is very likely to occur. Consequently, 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, half of the class receives treatment, while the other half is assigned to the control group. In the other version only ten classes receive treatment. Two outcome variables are used, one with and one without a TE (as indicated

[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 data set as described in this subsection.

by the superscript), to be able to assess size as well as power. The two outcomes are distributed as:

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

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

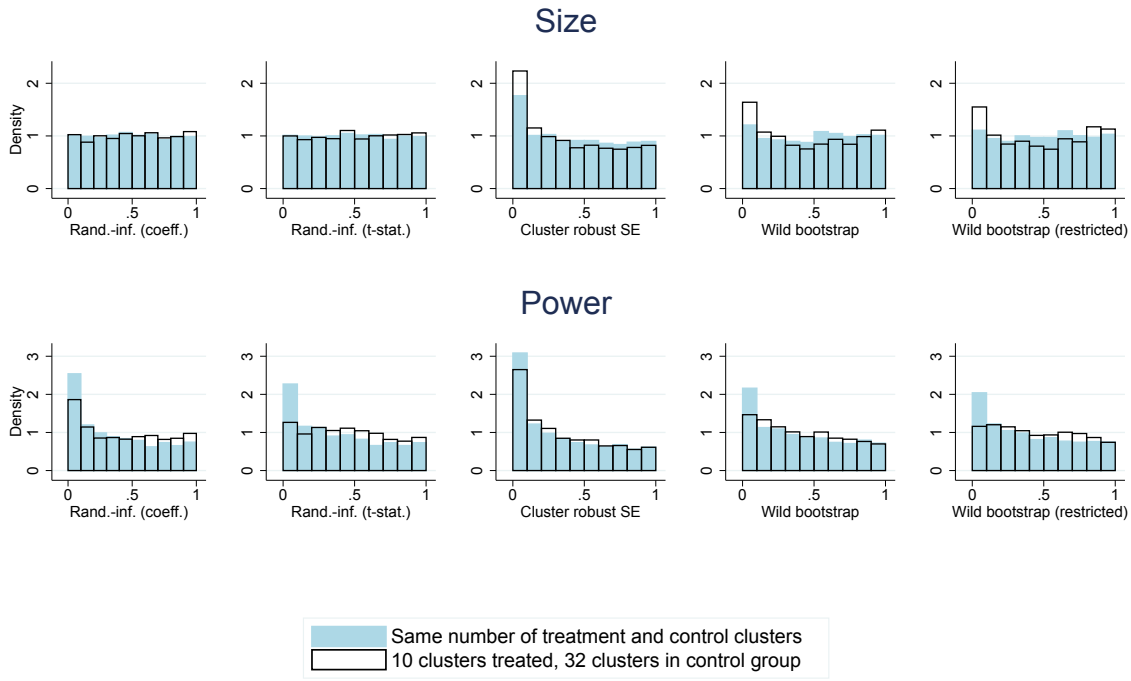

Figure 3: Distribution of p-values based on different testing methods.

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 blue bars correspond to the situation where treatment and control groups contain the same number of clusters, 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, 2016b). Yet, if treated clusters are few the wild bootstrap also over-rejects.

To illustrate statistical power, the lower half of Figure 3 shows the same tests, but uses the outcome where 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, in spite of the bootstraps being oversized.

4.3 Simulation with two strongly related tests

Assume TEs on two related outcomes are studied. It can happen that it is not possible to detect a significant TE estimate 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 ATE. Both outcomes are related, meaning they are affected by the same unobservable noise term.

The data is 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 estimating the TE for either outcome, detecting treatment is difficult as power is low. Figure 4 shows the distribution of estimates under the null hypothesis in contrast with the actual estimate in one 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.

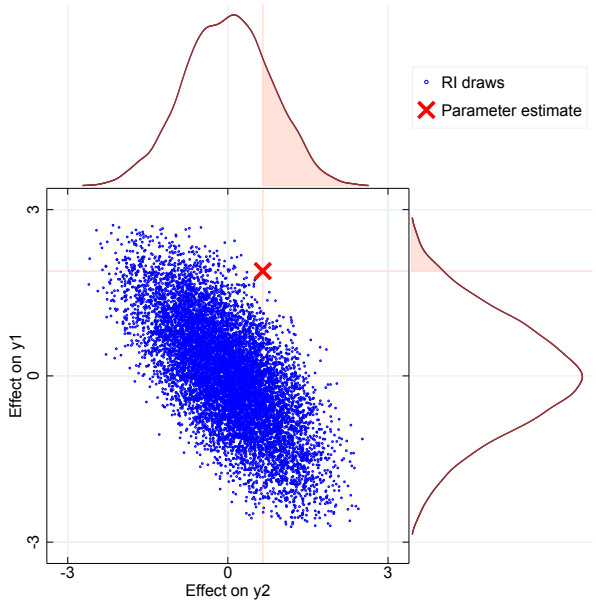

Figure 4: Joint distribution of two TE estimators under the null hypothesis vs. the actual estimate

In this example randomization inference provides the researcher with the correlation structure of the two tests. This information can be used to derive further tests. Either by approximating and parametricizing 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 which incorporates information from both dimensions. This could even be done without/before observing the actual realization where data mining is a concern. These can than 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

Since `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

This paper illustrates the usefulness of randomization inference for TE analysis. Using examples with real and simulated data, various cases in which randomization inference facilitates consistent analysis or provides additional insights are presented and discussed. The examples cover a range of data generating processes commonly encountered in experimental setups.

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

References

Angrist, J. D. and Pischke, J.-S. (2008). *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton university press.

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

Bloom, E., Bhushan, I., Clingingsmith, D., Hong, R., King, E., Kremer, M., Loevinsohn, B. and Schwartz, J. B. (2006). Contracting for health: Evidence from Cambodia. *Brookings Institution*.

Bowers, J., Fredrickson, M. and Hansen, B. (2016). RItools: Randomization inference tools. https://cran.r-project.org/package=RItools.

Bruhn, M. and McKenzie, D. (2009). In pursuit of balance: Randomization in practice in development field experiments. *American Economic Journal: Applied Economics*, **1** (4), 200–232.

Bugni, F., Canay, I. and Shaikh, A. (2016). *Inference under covariate-adaptive randomization*. Tech. rep., Working paper.

Cameron, A. C. and Miller, D. L. (2015). A practitioners guide to cluster-robust inference. *Journal of Human Resources*, **50** (2), 317–372.

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

—, Titiunik, R. and Vazquez-Bare, G. (2016). Inference in regression discontinuity designs under local randomization. *Stata Journal*, **16** (2), 331–367.

—, — and — (2017). 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 Dupas, P. (2010). Free distribution or cost-sharing? Evidence from a randomized malaria prevention experiment. *The Quarterly Journal of Economics*, **125** (1), 1–45.

Duflo, E., Glennerster, R. and Kremer, M. (2007). Using randomization in development economics research: A toolkit. *Handbook of Development Economics*, **4**, 3895–3962.

Fisher, S. R. A. (1935). *The Design of Experiments*. Macmillan.

Fujiwara, T. and Wantchekon, L. (2013). Can informed public deliberation overcome clientelism? Experimental evidence from Benin. *American Economic Journal: Applied Economics*, **5** (4), 241–255.

Ganong, P. and Jäger, S. (2015). *A Permutation Test for the Regression Kink Design*. Tech. rep., Working paper.

Ichino, N. and Schündeln, M. (2012). Deterring or displacing electoral irregularities? Spillover effects of observers in a randomized field experiment in Ghana. *The Journal of Politics*, **74** (01), 292–307.

Imbens, G. W. and Rubin, D. B. (2015). *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press.

Lachin, W. F., John M.; Rosenberger (2016). *Randomization in clinical trials : Theory and practice*. Wiley series in probability and statistics, John Wiley & Sons, 2nd edn.

Leamer, E. E. (2010). Tantalus on the road to Asymptopia. *The Journal of Economic Perspectives*, **24** (2), 31–46.

MacKinnon, J. G. and Webb, M. D. (2016a). *Randomization Inference for Difference-in-Differences with Few Treated Clusters*. Tech. rep., Carleton University, Department of Economics.

— and — (2016b). Wild bootstrap inference for wildly different cluster sizes. *Journal of Applied Econometrics*.

Romano, J. P. and Wolf, M. (2005). Exact and approximate stepdown methods for multiple hypothesis testing. *Journal of the American Statistical Association*, **100** (469), 94–108.

Rosenbaum, P. (2002). *Observational Studies*. Springer.

— (2010). *Design of Observational Studies*. Springer Series in Statistics.

Young, A. (2016). *Channeling Fisher: Randomization Tests and the Statistical Insignificance of Seemingly Significant Experimental Results*. Tech. rep., Working paper.