`RItools`: Flexible randomization inference in `R`

Mark M. Fredrickson*

September 13, 2012 (Version: 3e7d119)

1 Randomization Inference

"No man crosses the same river twice." Similarly, no unit can be experimented on twice. By applying a treatment, the unit is permanently changed. We cannot roll back the clock and see what would happen if we instead had applied a different treatment, the "fundamental problem of causal inference" (Holland, 1986). Formally, let $\mathbf{Z}$ be a *random* n-vector: $Z_i \in \{0, 1, \ldots, k\}$. In an experimental context, in which the researcher controls $\mathbf{Z}$, we call $\mathbf{Z}$ the *treatment assignment mechanism*. A realized value of $\mathbf{Z} = \mathbf{z}$ we call the *treatment assignment*. After assigning units to a level of treatment, we observe a *fixed* outcome $\mathbf{y_Z}$.[1] Our goal is to compare $\mathbf{y_Z}$ and $\mathbf{y_{Z'}}$. In this paper, we will emphasize comparison of the observed outcome with the outcome in which all units were assigned to level zero: $\mathbf{Z} = \mathbf{0}$, the "uniformity trial" (Bowers et al., 2012; Rosenbaum, 2007). The comparison of $\mathbf{y_Z}$ to $\mathbf{y_0}$ we call the *treatment effect*.

In a single experiment, we do not see $\mathbf{y_0}$. We only see $\mathbf{y_Z}$. In this view, causality is a missing data problem (Rubin, 1980). The task of the researcher is then to fill in the missing values of $\mathbf{y_0}$. By specifying a *model of effects* (hereafter "a model") that relates observed outcomes with unobserved outcomes the researcher can provide statements about what would have occurred if all units had received the same treatment. Formally, a model of effects is a function that maps observed data and a treatment vector to the "uniformity trial" potential outcome. Write $\mathcal{M}$ to represent a specific model, the observed outcomes as $\mathbf{y}$, and the observed vector of treatment assignments as $\mathbf{z}$. Then the adjusted data are the result of applying the model to the data and treatment assignment:

$$\mathbf{y_0} = \mathcal{M}(\mathbf{y}, \mathbf{z})$$

*Ph.D. Student, Dept. of Political Science, University of Illinois @ Urbana-Champaign (`fredric3@illinois.edu`). We greatly appreciate the detailed comments of Roger Koenker.

[1]Readers may be more familiar with individual level notation for potential outcomes: $Y_i(1)$ vs. $Y_i(0)$ (e.g., Holland, 1986). We use sample level notation to allow the possibility of spillover effects between subjects (i.e., treatment assignment to unit i changes the outcome of unit j). See Rosenbaum (2007) and Bowers et al. (2012) for details on how to specify models of treatment effect in the presence of spillover.

Unit	Z	y_z	$y_0 \mid \mathbf{y_0} = \mathbf{y}$	$y_0 \mid \mathbf{y_0} = \mathbf{y} - 2 \cdot \mathbf{z}$
1	1	6	6	4
2	0	2	2	2
3	0	4	4	4
4	1	4	4	2

Table 1: Illustrating the fundamental problem of causality for 4 units in an experiment with two treatment levels.

A very simple model might state that the treatment had no effect on any unit (often called the "sharp null of no effect"): $\mathbf{y_0} = \mathbf{y}$. A slightly more complicated model might state that treatment added two to the response of all treated units: $\mathbf{y_0} = \mathbf{y} - 2 \cdot \mathbf{z}$. With a model, the formerly unknown values can be recovered. Table 1 demonstrates this process for an experiment of 4 units with two treatment levels and two possible models. The column labeled Z indicates the realized assignment to treatment for each unit (either of two possible levels). The column labeled y_z shows the observed data after the experimental manipulation. The final two columns demonstrate the adjustments applied by (i) the sharp null of no effects and (ii) the model that subtracts 2 from every treated unit.

The table demonstrates the core intuition of a good model: a good model is one that makes the actual treated and control groups indistinguishable. Note that in the final column, the distribution of treated and control units is identical. In general, if the model were true, both groups should appear as random samples from a single population such that the treatment variable provides no information on the outcome: $\mathbf{Z} \perp \mathcal{M}(\mathbf{y}, \mathbf{z})$. When the model is the sharp null of no effects, the model implies that we have in fact observed a uniformity trial, and the values of $\mathbf{Z}$ are simply labels without actual effect. This defines our null hypothesis: we can reshuffle the labels at will and we should see no difference in the treatment and control groups. More complex models, by reducing data to the uniformity trial, allow us to use the same null hypothesis with the model adjusted data.[2]

To define alternative hypotheses and to score the similarity of the treatment and control groups, it is necessary to employ a *test statistic*:

$$\mathcal{T}(\mathbf{y}, \mathbf{z}) = t \in \mathbb{R}$$

The statistic must be effect increasing such that a larger test statistic indicates

[2]For simplicity, we treat models and hypotheses as interchangeable in this paper. The generic null hypothesis we use is that the stated model is true. It is also possible to state hypotheses over classes of models. For example, with binary data, we might wish to test the hypothesis, "the treatment made 5 units go from zero to one," which we might call a treatment effect of 5 units. This hypothesis is consistent with any model that switches 5 units that have $y_{\mathbf{0},i} = 0$ to $y_{\mathbf{Z},i} = 1$. We call these *attributable effects hypotheses*, which are perfectly valid under randomization inference (Rosenbaum, 2001; Hansen and Bowers, 2009). We have not yet extended `RItools` to provide this feature. We expect to handle more complex null hypotheses in future versions.

greater discrepancy between the treated and control groups (Rosenbaum, 2002, appendix to ch. 2). Examples of test statistics include the difference of the means or medians of the treated and control groups.

In his book *The Design of Experiments*, R. A. Fisher challenges researchers to consider if results could be due to chance by defining how else the experiment could have turned out. The only random variable, therefore the only source of chance variation, is $\mathbf{Z}$. Using this fact, we can define evidence against a model using the test statistic to create a p-value:

$$\Pr(\mathbf{y} \mid \mathcal{M}) = \Pr(\mathcal{T}(\mathcal{M}(\mathbf{y}, \mathbf{z}), \mathbf{Z}) > \mathcal{T}(\mathcal{M}(\mathbf{y}, \mathbf{z}), \mathbf{z})) \tag{1}$$

The random quantity in the right hand side of this equation is the distribution of test statistics implied by how $\mathbf{Z}$ can take values.

In an experiment, $\mathbf{Z}$ was generated by the researcher from a known process (for example, flipping a coin for every unit). We can define a *treatment assignment mechanism* to explain the possible draws and their probabilities. The mechanism describes the support of $\mathbf{Z}$, which we notate as Ω, and the probability of any drawing any particular $\mathbf{Z}$: $\Pr(\mathbf{Z} = \mathbf{z})$. For our four unit example, if we fixed all units with $p = 0.5$ probability of receiving treatment:

$$\begin{aligned} \Omega = \{&0000, 1000, 0100, 0010, 0001, 1100, 1010, 1001, \\ &0110, 0101, 0011, 1110, 1011, 1101, 0111, 1111\} \\ \Pr(\mathbf{Z} = \mathbf{z}) &= \frac{1}{|\Omega|} = \frac{1}{2^4} = \frac{1}{16} \end{aligned}$$

Alternatively, if we constrained ourselves to only consider randomizations with two of the units treated and sampled uniformly:

$$\begin{aligned} \Omega &= \{1100, 1010, 1001, 0110, 0101, 0011\} \\ \Pr(\mathbf{Z} = \mathbf{z}) &= \frac{1}{|\Omega|} = \frac{1}{\binom{4}{2}} = \frac{1}{6} \end{aligned}$$

As with the example models of effect, these examples are intentionally simple, but more complex assignment mechanisms can be modeled as long as we understand the possible realizations and the probability of each.

After the researcher defines the treatment assignment mechanism, the model of effects, and the test statistic, the randomization inference algorithm can be written as:

1. Adjust data consistent with the model: $\mathbf{y_0} = \mathcal{M}(\mathbf{y}, \mathbf{z})$
2. Compute $t = \mathcal{T}(\mathbf{y_0}, \mathbf{z})$
3. For each $\mathbf{Z} \in \Omega$ (or a sample) compute $\mathcal{T}(\mathbf{y_0}, \mathbf{Z})$
4. Compute the p-value:

$$p = \sum_{i=1}^{|\Omega|} \mathrm{I}(\mathcal{T}(\mathbf{y_0}, \mathbf{Z}_i) > t) \Pr(\mathbf{Z}_i)$$

This algorithm can be repeated for a series of models, each receiving a p-value. For example, we could consider a group of models, all parameterized by θ where each unique value of θ implies a model (a function of the data and observed values): $\mathcal{M}_{\theta=x}(\mathbf{y}, \mathbf{z})$. A α-level confidence set is the set of models for which $p > (1 - \alpha)$. This formulation has a natural point estimate in the form of the Hodges-Lehmann (HL) point estimate (Hodges and Lehmann, 1963). Formally, we say the HL estimate is the parameter value that makes the expectation of the test statistic equal to the observed test statistic on the adjusted data (Rosenbaum, 2002, ch. 2).[3] In practice, the HL estimate is often taken to be the model (and implied parameter value) with the highest p-value (e.g., Hill and Reiter, 2006), a convention we adopt.

The remainder of this paper explains how the concepts of randomization inference are implemented in the `R` package `RItools`.[4] Throughout, we use as an example an educational field study in which both schools and students were randomly assigned to treatment conditions. Section 2 shows how models, test statistics, and treatment assignment mechanisms are created and used in the randomization inference algorithm. Section 3 provides discussion of computational efficiency, similar software packages, and concluding thoughts. The appendices include additional examples applying `RItools` and a comprehensive list of available models, test statistics, and assignment mechanisms.

2 Demonstration using ERO study

Somers et al. (2010) describe a large scale educational field experiment designed to compare two literacy enhancement programs against standard educational practice. Participating high schools in the study were randomized to one of two new reading curricula: the "Xtreme Reading" program or the "RAAL" program. Within each school, incoming freshman students were assessed for reading proficiency. Students performing one to five years below grade level were randomized to either take an additional reading class (using either the Xtreme or RAAL methods depending on the school level randomization) or were assigned to take a standard freshman elective course. Students were therefore randomized to one of three levels: taking no additional reading class, taking a reading class using "Xtreme Reading" or taking a reading class using "RAAL." Reading performance measures were reapplied at the end of the freshman year for all students.

These data exhibit several interesting characteristics. First, randomization occurred at two levels, combining a cluster randomized study and a individual randomized study. The lower level randomization was conditional on the upper level, such that students in an Xtreme Reading school could never be assigned to

[3]When multiple values of the parameter fulfill this description, the usual behavior is to take the mid-point of the potential values. We relax this condition and allow the HL estimate to be an interval or set of values. This is useful when considering multiple parameters, and researchers should have no problem reducing these intervals to single points if they so desire.

[4]For the software demonstrated in this paper, see the `RItools` repository.

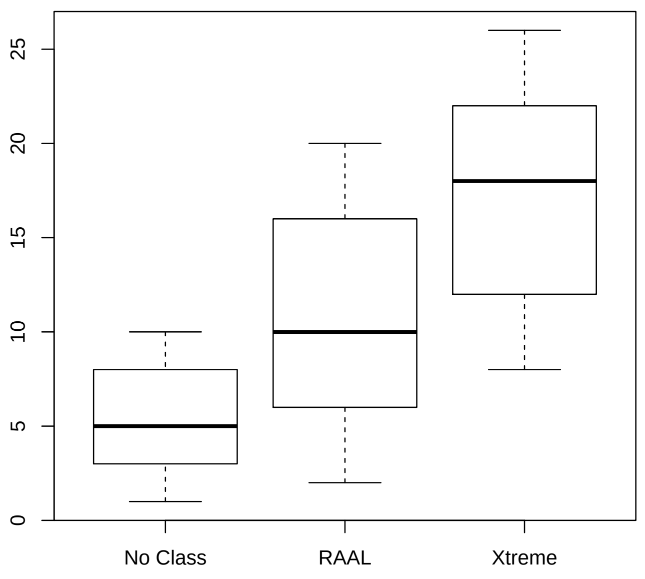

Figure 1: (Simulated) Outcomes for Enhanced Reading Opportunities study. Units are students randomized to either no reading class, a reading class using RAAL or a reading class using Xtreme Reading.

a RAAL class or vice-versa. Second, despite randomizing the reading program type at the school level, the researchers were most interested in student level performance. Finally, while the study population included students reading one year behind grade level, the researchers were most interested in studying the effect on students reading 2 or more years behind grade level.

The following sections consider a simulation of the ERO study.[5] The simulated data contain schools across 2 districts, each with a student population ranging from 80 to 140 students. The simulated outcome can be considered a reading aptitude test carried out at the end of the freshman year. Figure 1 shows a box plot of outcome for each of the three treatment levels (no class, class with RAAL, or class with Xtreme). The data show differences across all three groups, with changes in both location and spread. In the next sections, we consider how to implement the treatment assignment of the ERO study, how to select a model to evaluate against these data, and what test statistic will be most useful in this process. Finally, we combine these choices to analyze ERO

[5]I am working with the study authors and the IES to gain access to the real data.

study for possible treatment effects.

2.1 Assignment Mechanisms

RItools implements assignment mechanisms as functions that take a number of samples and return a sample, without replacement, from the universe of possible randomizations. Additionally, for cases where samples do not have equal weight, the treatment assignment mechanism returns a weight for each draw. The sample may be the entire sample space, which is the case when more samples than unique randomizations are requested. For example, consider the case of randomly assigning 8 units, 3 to treatment and 5 to control (with a total sample space of $\binom{8}{3} = 56$ possible randomizations).

```
> # a little helper function
> expand.z <- function(index) { a <- numeric(8); a[index] <- 1; a }
> assign.3.5 <- function(samples) {
+   if (samples >= choose(8,3)) { # enumerate
+     zs <- apply(combn(8,3), 2, expand.z)
+   } else { # sample
+     zs <- matrix(nrow = 8, ncol = samples)
+
+     for (i in 1:samples) {
+       zs[,i] <- expand.z(sample.int(8, 3))
+     }
+
+   }
+   return(list(weight = 1, samples = zs))
+ }
> dim(assign.3.5(100)$samples)

[1]  8 56

> dim(assign.3.5(10)$samples)

[1]  8 10
```

The weight component of the return value can also be used by importance sampling approaches that target uniformly likely distributions. The weights are used by p-value functions, which we discuss in Section **??**.

Treatment assignment mechanisms may also be used during design of the experiment to generate the treatment assignment that will be used. Simply request a single sample.

2.1.1 ERO Treatment Assignment

The ERO study recruited high schools to participate in the study from several school districts. The researchers blocked high schools within districts and randomly assigned half of each district to the Xtreme condition and half to the RAAL condition. We can therefore describe the school level assignment mechanism using the `simpleRandomSampler` function:

```
districts <- c(rep(1,6), rep(2,8)) # data in district order
total.by.district <- as.numeric(table(districts))
treated.by.districts <- total.by.district/2 # 1/2 of schools in each district
school.sampler <- simpleRandomSampler(total = total.by.district,
                                      treated = treated.by.districts)
```

At the student level, the researchers also used a simple random sample within blocks. 60 students in each school were assigned to attend the enhanced reading course. The remaining students were assigned to a standard freshman elective course.

```
# for each school, I draw a number of students Unif[80,160]
students.by.school <-  sample(x = 80:140, size = length(districts), replace = T)

# the numeric treated at each school is fixed at 60
treated.by.school <- rep(60, length(districts))

student.sampler <- simpleRandomSampler(total = students.by.school,
                                       treated = treated.by.school)
```

This procedure generates a total of 1582 students, 840 of which are assigned to take one of the reading courses.

Discounting any school level effect of being assigned to either the Xtreme or RAAL conditions, we can consider the student level assignment to be a three level ordinal variable: $Z \in \{\text{Xtreme}, \text{RAAL}, \text{none}\}$. It would be incorrect to assign students directly to these conditions as that would allow two students in a single school to be assigned to different reading conditions, which was not possible in the true study design. Instead, we can combine results from the upper and lower level samplers to create a proper treatment assignment:

```
studentSchoolSampler <-

function (n)
{
    schools <- school.sampler(n)$samples
    swidth <- dim(schools)[2]
    if (swidth < n) {
        schools <- cbind(schools, schools[, sample.int(swidth,
            size = n - swidth, replace = T)])
    }
    expanded.schools <- apply(schools, 2, function(col) {
        rep(col, students.by.school)
    })
    students <- student.sampler(n)$samples
    randomizations <- (expanded.schools + 1) * students
    return(list(weight = 1, samples = randomizations))
}
```

2.2 Models of Effects

Models of effects in RItools are implemented as R functions. At the most basic level, *unparameterized models* are functions of precisely two arguments:

```
> model(y, z)
```

Where y is the data and z is the treatment assignment. The function returns an updated copy of the data, adjusting away the effect captured in the model. In Section 1, we saw two models: the sharp null of no effects ($\mathbf{y_0} = \mathbf{y}$) and a simple additive model ($\mathbf{y_0} = \mathbf{y} - 2 \cdot \mathbf{z}$). Neither of these models requires parameters, so the implementation of these models as functions of two arguments is straight forward:

```
> sharp.null <- function(y, z) {
+   y # simply the identity function
+ }
> additive.two <- function(y, z) {
+   y - 2 * z
+ }
```

The second model suggests a class of models, all of which add or subtract a value from the observed data to create the uniformity trail. We might write the constant additive model as:

$$\mathbf{y_0} = \mathbf{y} - \tau \cdot \mathbf{z}$$

Where τ represents a parameter in the model. We can extend our earlier definition of models to *parameterized models* that include additional parameter arguments:

```
> model(y, z, parameter1, parameter2, ...)
```

The additive model can be implemented as:

```
> additive.model <- function(y, z, tau) {
+   y - z * tau
+ }
```

For the most part, RItools will automatically convert parameterized models into unparameterized models as necessary. In fact, most users need only concern themselves with parameterized models. There are few places, however, where unparameterized models are required for more advanced model creation and evaluation. Converting from parameterized models to unparameterized models is simple using the givenParams function:

```
> additive.two <- givenParams(additive.model, tau = 2)
```

2.2.1 Modeling the ERO effects

Within the three level individual level randomization (Xtreme, RAAL, no class), there may be an effect for taking a class and a separate effect for the particular type of class. Moreover, these effects may not combine in simply an additive manner. It seems plausible that either the Xtreme or RAAL class would provide the greatest benefit to the students students who already had the best skill set (i.e. students who would have had high scores if they had been assigned to the "no class" condition). At the same time, one of the two programs might be more effective than the other, leading to a increase in performance across the board.

We can capture these theoretically motivated ideas in the following model. If we write $\mathbf{Z}_{\text{ERO}}$ to indicate randomization to either of the two class conditions and $\mathbf{Z}_{\text{Xtreme}}$ to indicate randomization to the Xtreme class specifically, we can relate observed outcomes to the hypothetical uniformity trial in which all students were assigned to "no class":

$$\mathbf{y_0} = \beta^{-\mathbf{Z}_{\text{ERO}}}(\mathbf{y} - \tau\mathbf{Z}_{\text{Xtreme}}) \tag{2}$$

Where $\beta, \tau \in \mathbb{R}$ and $\beta \neq 0$. This model is captured using the following `R` function:

```
student.model <-

function (y, z, beta, tau)
{
    z_ero <- as.numeric(z != 0)
    z_xtreme <- as.numeric(z == 2)
    (beta^(-z_ero)) * (y - tau * z_xtreme)
}
```

2.3 Test Statistics

Test statistics are also implemented as `R` functions. Test statistics are functions that map observed data y and treatment assignment z to a real value that expresses the difference in treatment levels.[6] The test statistic should be *effect increasing*, so that larger discrepancies in the treated and control distributions are exhibited in larger magnitudes of the test statistic.

As an example, consider a test statistic that compares the difference of means of the treated and control groups, ignoring blocking:

```
> mean.difference <- function(y, z) {
+   mean(y[z == 1]) - mean(y[z == 0])
+ }
> mean.difference(1:4, c(1,0,0,1))
```

[6]Test statistics have the same signature as unparameterized models. There are no parameterized test statistics.

```
[1] 0
```

Instead of looking at the central tendency of the treated and control distributions, we might want a test statistic to be sensitive to changes at the tails. The following statistic compares the differences at the 25th and 75th quantiles of the data and returns the maximum absolute difference:[7]

```
> max.diff.25.75 <- function(y, z) {
+   z1 <- ecdf(y[z == 1])
+   z0 <- ecdf(y[z == 0])
+   max(abs(quantile(z1, 0.25) - quantile(z0, 0.25)), abs(quantile(z1, 0.75) -
+   quantile(z0, 0.75)))
+ }
> max.diff.25.75(1:20, rep(c(1,0,0,1), 5))
```

```
[1] 0.5
```

2.3.1 Test statistic for the ERO model

In summarizing the differences across three distributions, corresponding to the three levels of randomization, we employ a slight variation on the standard median difference statistic. The difference of each group is taken, and the largest magnitude result is returned:

```
max.median.diff <-

function (y, z)
{
    a <- median(y[z == 1])
    b <- median(y[z == 2])
    c <- median(y[z == 0])
    max(abs(a - c), abs(b - c))
}
```

2.4 Analysis

Having selected a treatment assignment mechanism, a model of effects, and a test statistic, the researcher can proceed to analyzing the support for the model based on the data. As discussed in Section 1, the algorithm for computing a p-value for a model involves adjusting the observed data by the hypothesized model and repeatedly computing the test statistic for all (or a sample) of the possible randomizations. `RItools` has both a high level and low level interface to this algorithm. The high level interface `RItest` works with parameterized with models of effect and a set of parameter values to test. The low level interface `randomizatonDistributionEngine`, which provides the backend for the

[7] This statistic may be considered a limited version of the Kolmogorov-Smirnov test statistic that only considers specific point-wise differences.

high level interface, works on unparameterized models and provides a less convenient output format, but is more powerful for advanced users seeking specialized analysis. While we present both interfaces here, we recommend users focus on RItest as their primary method of analyzing data.

2.4.1 randomizationDistributionEngine

This function is the workhorse of the analysis phase. It has the following signature:

```
> randomizationDistributionEngine(
+   y,
+   z,
+   models,
+   sampler = simpleRandomSampler(z = z, b = rep(1, length(z))),
+   samples = 5000,
+   p.value = general.two.sided.p.value,
+   include.distribution = FALSE,
+   summaries = list(),
+   type = "exact",
+   ...)
```

Arguments y and z are the same data and treatment indicators we have seen in used in the models of effects and test statistics. The argument models is a list of lists of the form:

```
list(
  list(test.statistic.1, model1, model2, model3, ...)
  list(test.statistic.2, model4, model5, model6, ...)
  ...)
```

Where test.statistic.n is a test statistic function as we've seen before and the models are all *unparameterized* (i.e., they are functions of precisely two arguments: y, z). The models in the lists need not be unique across lists. For example, you could supply two identical sets of models with two different test statistics to assess whether one statistic was better than the other at discriminating between competing models. The engine will automatically prepend the "sharp null of no effects" model (i.e., the identity function with respect to y), so users do not need to supply it. The return value of the function will have the same length as the models with one entry for each test statistic and model list. The sampler argument is a treatment assignment mechanism, and the samples will be passed to the sampler to generate the required draws from the randomization sample space Ω. If samples is larger than $|\Omega|$, then the sample space will be enumerated exactly.

The p.value argument is a function that computes p-values given the observed data and the distribution formed by evaluating the test statistic over the samples draws. For more on the p-value function, see Section ??. By default, the raw distribution of test statistic values is not returned, just the summary formed by the p-value. Setting `include.distribution = TRUE` will include the raw data, though users should be aware this can be quite memory intensive. Another alternative to the raw data is to include a list of functions in the `summaries` argument (for example, to compute the mean of the test statistic distribution). These functions should take a vector representing all the computed values and return a real value.

The final argument of `type` can be set to either `"exact"` or `"asymptotic"`. If the sampler and test statistic are compatible and the `"asymptotic"` option is set, a large sample approximation will be used when possible. Only a very small number of test statistics and samplers are compatible with this flag (at the time of this writing, only `harmonic.mean.difference` and `mann.whitney.u` are compatible with the `simpleRandomSample` sampler). If a large sample approximation is possible, the `samples` argument is ignored.

The return value of the function is a list of objects of class `"RandomizationDistribution"`, which descends from `data.frame`. The object has slots that hold the arguments to the function (for later reference), and the `data.frame` has two columns: `statistic` and `p.value`. The first is the result of applying the test statistic function to the model adjusted data. The second is the proportion of samples with more extreme values than the `statistic` column, as computed by the `p.value` function supplied to the engine. The `data.frame` will have height equal to the number of models test using this test statistic (including the always added sharp null). The results of `summaries` functions will be additional columns.

2.4.2 `RItest`

For users testing parameterized models over a range of parameter values and using a single test statistic, the `RItest` interface will be easier to use and provides a more friendly output format. The function signature looks similar to the engine, but with a few differences:

```
RItest <- function(
  y,
  z,
  test.stat,
  moe = NULL,
  parameters = NULL,
  ...)
```

The `y` and `z` arguments are as before. The `test.stat` argument is a single test statistic function. The optional `moe` argument is a *parameterized* model of effects function (i.e., it takes additional arguments to `y` and `z`). If no argument is supplied, only the sharp null is tested. If `moe` is supplied, then there must also be a `parameters` argument (and vice versa). `parameters` takes the for of a list of vectors:

```
  list(tau = c(1,2,3,4,...), beta = c(-4,0,4), ...)
```

Where `tau` and `beta` are parameter arguments to the `moe` function. Any additional arguments are passed to `randomizationDistributionEngine` (most notably the `sampler` and `samples` arguments).

Behind the scenes, `RItest` creates unparameterized models over the Cartesian product of all the parameter values, and passes these models and the test statistic to the engine to do the computation. The return value of `RItest` is a subclass of `"RandomizationDistribution"`, aptly called `"ParameterizedRandomizationDistribution"`. It is otherwise identical to the parent except that it also maintains the parameters combinations and a copy of the call to the function.

2.5 Analysis

Analysis is performed using the `RItest` interface. After creating the treatment assignment mechanism, the model of effects, and the test statistic, the only remaining task is to select the range of parameters to search and the number of samples from the randomization distribution sample space to draw. For the former, I canvas a small range of values around the true simulation values. For samples, I select 5000.

```
RItest(y = y, z = z, test.stat = max.median.diff, moe = student.model,
    parameters = list(beta = seq(1, 4, by = 0.25), tau = seq(4,
        10, by = 1)), sampler = studentSchoolSampler, samples = 5000)
```

The `summary` method provides evidence against the sharp null of no effects as well as a point estimate for the parameters.

```
> summary(analysis)

Call:  RItest(y = y, z = z, test.stat = max.median.diff, moe = student.model,
          parameters = list(beta = seq(1, 4, by = 0.25), tau = seq(4,
               10, by = 1)), sampler = studentSchoolSampler, samples = 5000)

                        Value    Pr(>x)
Observed Test Statistic    13 < 2.2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Hodges-Lehmann Point Estimate(s):
  beta tau statistic p.value
1 1.75   8 0.7142857       1
2 2.00   8 0.0000000       1
3 2.25   8 0.5555556       1
```

The probability of observing these data if the experiment had no effect is extremely small. The point estimate differs from the true values, though not in sign. The class benefits strong students and all students benefit from the Xtreme

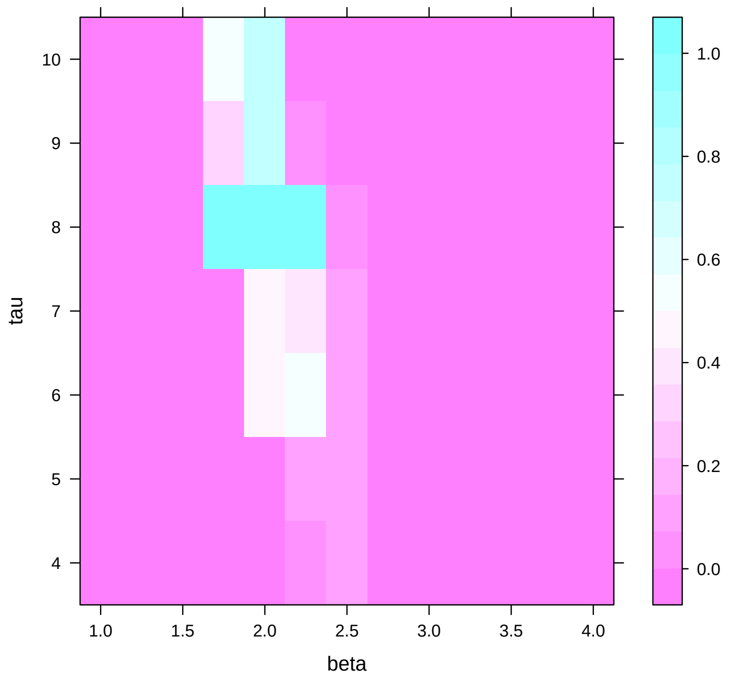

Figure 2: p-value plot for tested values of β and τ on the ERO study data

Reading program more than the RAAL program. Figure 2 shows a plot of the p-values at each evaluated pair of parameters.

3 Discussion

3.1 Computational Efficiency

3.2 Similar Software

3.3 Conclusion

A Simulation Details

Since we use simulated data instead of real data, we provide a few details on how the data are generated. This section will be removed in the final paper.

First, the simulated treatment assignment is generated by running the assignment mechanism once. Then the true uniformity trial is generated from a standard Normal distribution. Finally, using an inverse function of the model, the effect is generated using a true $\beta = 2$ and $\tau = 6$.

```
z <- studentSchoolSampler(1)$samples[,1]

true.beta <- 3
true.tau <- 1

y_unif <- rnorm(n = length(z))
y <- student.model.inverse(y_unif, z, true.beta, true.tau)
```

B Additional Examples

This appendix demonstrates several additional randomized experiments evaluated using `RItools`.

B.1 Simple Simulation Exercise

We start the examples with an extreme simple simulation study on 100 units. The simulation starts by creating a true uniform trial: units vary uniformly on the interval 1 to 10. The observed treatment assignment is also generated immediately, with 50 units assigned to both control and treatment conditions. The actual effect is an increase of 3 units for all treated units. The analysis proceeds using this simulated data and evaluating several hypothesized values of τ in the neighborhood of the true value.

```
simple <- data.frame(uniform = sample.int(10,
                       size = 100, replace = T),
                     z = as.numeric(gl(2, 100)) - 1)
simple$y <- simple$uniform + simple$z * 3

simple.res <- RItest(
                simple$y,
                simple$z,
                mean.difference,
                moe = constant.additive.model,
                parameters = list(
                  tau = seq(2, 4, by = 0.1)),
                samples = 500)
```

The summary method displays the p-value for the sharp null of no effects (using the observed test statistic value), and also attempts to generate a point estimate by finding the parameter combination with the largest observed p-value.

```
> summary(simple.res)
```

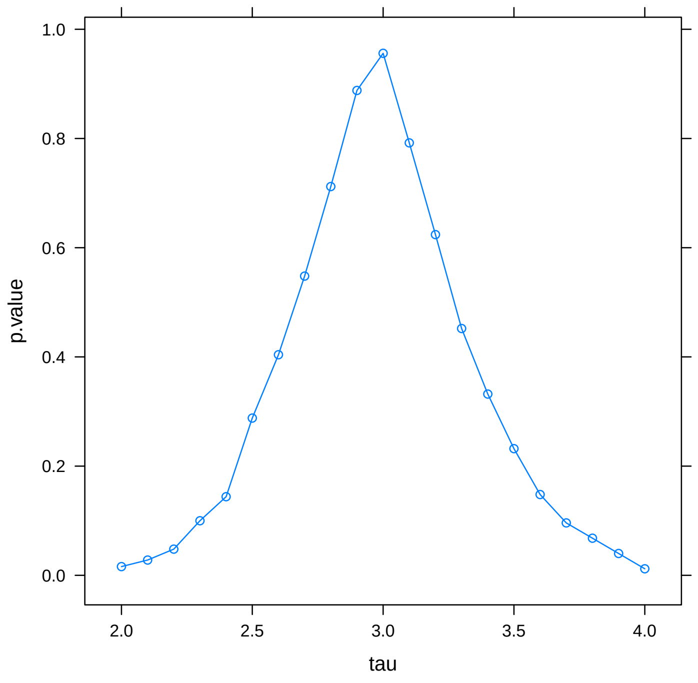

Figure 3: p-value plot for tested values of τ on simulated data

```
Call:  RItest(y = simple$y, z = simple$z, test.stat = mean.difference,
          moe = constant.additive.model, parameters = list(tau = seq(2,
               4, by = 0.1)), samples = 500)

                        Value    Pr(>x)
Observed Test Statistic     3 < 2.2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Hodges-Lehmann Point Estimate(s):
  tau statistic p.value
1   3         0   0.956
```

The simulation finds the true value as the most likely to have generated the observed data, placing a high p-value on the true $\tau = 3$. This can be verified visually using the plot method on `simple.res`, as seen in Figure 3.

B.2 The Lady Tasting Tea

Fisher (1935) describes an experiment to determine the validity of a claim that a person could tell whether tea or milk was first added to a cup, with the other being added second. To test this claim, Fisher proposed a simple experiment. Eight cups would be randomly assigned to either receive milk or tea first. The subject would taste the 8 cups and write down which four she thought had the milk added first. How surprised should we be if the subject was able to correctly determine 3 of the 4 milk-first cups?

To determine the answer to this question, let us first define the possible ways in which the cups could be randomized to the milk and tea conditions. With 4 cups always allocated to both, this randomization scheme can be implemented using `simpleRandomSampler`:

```
lady.sampler <- simpleRandomSampler(8,4)
```

Less assume we drew the following randomization, and the subject labeled the cups as follows, erroneously swapping the last two cups:

```
> actual.cups

[1] 1 0 0 1 1 0 1 0

> lady.guess

[1] 1 0 0 1 1 0 0 1
```

Finally, to summarize the correctness of the guess, compare the two vectors item by item. Since each wrong guess implies *two* differences in the vector, divide the result by 2 to get the number of milk cups properly labeled:

```
number.correct <-

function (y, z)
{
    sum(z == y)/2
}

> number.correct(lady.guess, actual.cups)

[1] 3
```

We can evaluate the probability of seeing 3 or more properly labeled cups using the parameterized interface, even though we are only interested the sharp null of no effects.

```
lady.distribution <- RItest(lady.guess,
                            actual.cups,
                            test.stat = number.correct,
                            p.value = upper.p.value,
                            sampler = lady.sampler,
                            include.distribution = T)
```

Note that this call uses a different p-value function than the default as we are only interested in the alternative of *more* pairs correctly categorized. We also save the entire distribution of possible test statistic values for later.

So how did the lady do? According to our analysis, we might not be very surprised to see the lady properly label 3 cups, even if she was only guessing. Almost 25% of the time, random guessing would result in 3 or more cups properly labeled.

```
> summary(lady.distribution)

Call:  RItest(y = lady.guess, z = actual.cups, test.stat = number.correct,
           p.value = upper.p.value, sampler = lady.sampler, include.distribution = T)

                         Value Pr(>x)
Observed Test Statistic     3 0.2429
```

Figure 4 shows the complete distribution of the test statistic traced over all possible random assignments. With only a 1 out of 70 chance of properly labeling all 4 cups correctly, we would have been very impressed to observe this event.

C Available Models, Test Statistics, and Assignment Mechanisms

C.1 Models

Constant Additive Model The function `additive.model` above is implemented in `RItools` `constant.additive.model`, where the term "constant" is employed to indicate that the additive effect τ does not vary by unit. The parameter name is also `tau`.

Constant Multiplicative Model Similar to an additive effect, we can also consider a multiplicative effect β:

$$\mathbf{y_0} = \beta^{-\mathbf{z}}\mathbf{y}$$

This model is implemented as `constant.multiplicative.model`. The parameter name is `beta`.

Min-Max Meta-Model The "min-max meta-model" is, as the name applies, in not a model itself, but rather a way of creating new models. Given upper bound b, lower bound a, and another model $\mathcal{M}$:

$$\mathbf{y_0} = \min(b, \max(a, \mathcal{M}(\mathbf{y}, \mathbf{z})))$$

This model generator is useful if the potential outcomes are constrained by upper and lower values, such as a 7 point attitude scale on a survey or an

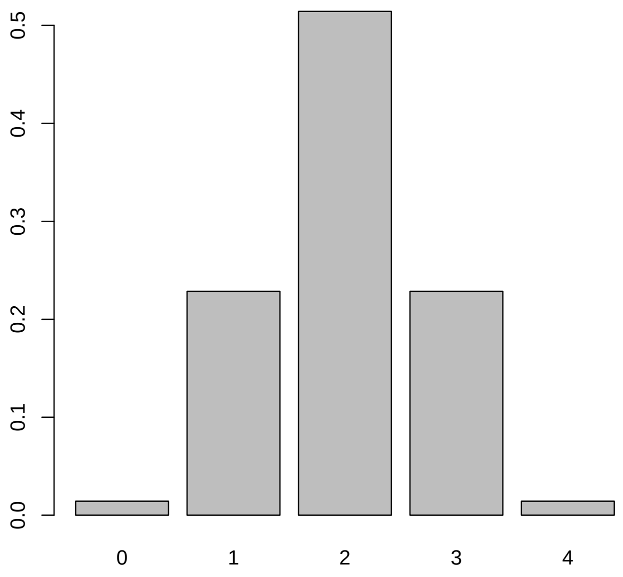

Figure 4: Distribution of correctly labeled cups for null hypothesis of no effect.

instrument that only reports positive values. The function returns a new model. For example, an additive model that is constrained within 0 and 7.

```
> my.model <- min.max.model(model = constant.additive.model, lower = 0, upper = 7)
```

The arguments `lower` and `upper` have default values of `-Infinity` and `Infinity` respectively. The function `my.model` behaves similarly to the `constant.additive.model`, taking a parameter `tau`.

Direct-Indirect Models

C.2 Test Statistics

Mean Difference The mean difference computation above is implemented in `RItools` as `mean.difference`.

Odds Ratio For binary data, the odds ratio is a well known test statistic. It is implemented as `odds.ratio`.

Mann-Whitney "U" The Mann-Whitney "U" statistic will be familiar to users of the Wilcoxon location test. It considers the sum of the ranks of the treated data. It can be found at `mann.whitney.u`.

Quantile Absolute Difference This is a test-statistic generator, i.e. a function that returns a new test statistic. Like the modified KS test statistic above, it compares the differences between treated and control units at a series of quantiles, returning the largest absolute difference. The user supplies `quantiles`, a vector of quantile comparison points, when generating the test statistic:

```
> median.difference <- quantileAbsoluteDifference(0.5)
> quartile.difference <- quantileAbsoluteDifference(c(0, 0.25, 0.5, 0.75, 1))
```

Subset Statistic Another test statistic generator, this time generalizing the process of subgroup analysis in Section **??**. Given an existing test statistic and a logical vector, `subsetStatistic` will only perform the test statistic on the subset of the data indicated in the `subset` logical. The example above could be written more simply as:

```
> female.mean.difference <- subsetStatistic(mean.difference, gender == 0)
```

C.3 Assignment Mechanisms

As with models and test statistics, we provide several generator functions to create appropriate assignment mechanisms. These generators take a set of arguments and generate a function that accepts a single argument, the number of samples, and returns the appropriate randomizations.

Simple Random Samples The example above demonstrates a simple random sample of size 3 from 8 possible units. The function `simpleRandomSampler` generalizes this pattern for any number of treated units less than the total number of units. The above example could be recreated as:

```
> assign.3.5 <- simpleRandomSampler(total = 8, treated = 3)
```

If we wish to add blocks, both `total` and `treated` can be passed as vectors of the counts of block total units and block treated units.

```
> assign.blocked <- simpleRandomSampler(total = c(8,6), treated = c(3,2))
```

Data should be in block order when using this function (i.e., all the observations in block 1 should come before observations in block 2 in `y` arguments to models, test statistics, and the analysis functions). If data are not in block order and you have an existing treatment assignment and blocking factor, you can use the alternate interface:

```
> assign.blocked.outoforder <- simpleRandomSampler(z = c(1,0,0,1), b =
+ c(1,2,1,2)) # two blocks, two treated units each
```

You are also allowed to pass both `total` and `z` or `treated` and `b`. Any other combinations are not allowed.

Independent Probability Assignment For reasons of experimental design, the units may each be assigned to treatment and control independently of each other (e.g., each unit is assigned as the result of a coin flip). This assignment mechanism is implemented in the function `independentProbabilitySampler`. The function takes two arguments: `n`, the number of units in the experiment; `p`, an optional vector of the probability of each unit receiving treatment. By default, `p = 0.5` for all units.

When `p` is not simply 0.5 for all units, the `weight` component of the return value will be the probability of seeing each treatment assignment in the sample:

$$\prod_{i=1}^{n} z_i p_i + (1 - z_i)(1 - p_i) \tag{3}$$

References

Bowers, J., Fredrickson, M. M., and Panagopoulos, C. (2012). Reasoning about interefernce between units. Working paper.

Fisher, R. A. (1935). *The Design of Experiments.* Oliver and Boyd, Edinburgh.

Hansen, B. B. and Bowers, J. (2009). Attributing effects to a cluster-randomized get-out-the-vote campaign. *Journal of the American Statistical Association*, 104(487):873 – 885.

Hill, J. and Reiter, J. P. (2006). Interval estimation for treatment effects using propensity score matching. *Statistics in Medicine*, 25(13):2230–2256.

Hodges, J. L., J. and Lehmann, E. L. (1963). Estimates of location based on rank tests. *The Annals of Mathematical Statistics*, 34(2):pp. 598–611.

Holland, P. W. (1986). Statistics and causal inference. *Journal of the American Statistical Association*, 81(396):945–960.

Rosenbaum, P. R. (2001). Effects attributable to treatment: Inference in experiments and observational studies with a discrete pivot. *Biometrika*, 88(2):219 – 231.

Rosenbaum, P. R. (2002). *Observational Studies.* Springer, 2^{nd} edition.

Rosenbaum, P. R. (2007). Interference between units in randomized experiments. *Journal of the American Statistical Association*, 102(477):191 – 200.

Rubin, D. B. (1980). Randomization analysis of experimental data: The fisher randomization test comment. *Journal of the American Statistical Association*, 75(371):591–593.

Somers, M.-A., Corrin, W., Sepanik, S., Salinger, T., Levin, J., Zmach, C., Wong, E., Strasberg, P., and Silverberg, M. (2010). The enhanced reading opportunities study final report: The impact of supplemental literacy courses for struggling ninth-grade readers. Technical Report NCEE 2010-4021, National Center for Educational Evaluation and Regional Assistance, US Dept. of Education, Alexandria, VA.